ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Bachelor Thesis BSc^2 Econometrics/Economics (FEB24100)

The profitability of drones in the vehicle routing problem with stochastic drone delivery

Name student: Jimme van der Leij
Student ID number: 513394
Supervisor: Ymro Hoogendoorn
Second assessor: Dr. Oguzhan Vicil
Date: 03-07-2022

Abstract

Using unmanned aerial vehicles, also known as drones, to deliver packages could revolutionize the logistics industry. Incorporating drones into delivery systems could save costs, minimize delivery time and decrease pollution. For this reason, many delivery companies have started to experiment with using drones. There are, on the other hand, also great concerns about the safety of drones. Drones have a risk to collide with objects or other drones in mid-air and, as a consequence, fall down potentially hurting people on the ground. From an operations research perspective, extensive literature already exists on using drones in logistics systems. However, incorporating a risk of failed drone delivery by, for example, a collision into a mathematical formulation has not yet been implemented. Therefore, a variation on the Vehicle Routing Problem with Drones, by Sacramento, Pisinger, and Ropke (2019), will be studied in this paper to incorporate a failure probability for drone delivery. Our formulation will be called the Vehicle Routing Problem with Stochastic Drone Delivery. This formulation considers a fleet of trucks all accompanied by a single drone, which needs to serve a set of customers with a service level guarantee. While delivery by truck is considered to be always successful, drone delivery will have a chance to fail. This paper will then assess the effect of this failure probability on the profitability of drones in delivery systems. To get good solutions for our model in reasonable running times an Adaptive Linear Neighbourhood Search metaheuristic will be used from Sacramento et al. (2019), which was adapted to fit our formulation.

Contents

1 Introduction

In 2013 Jeff Bezos, CEO of Amazon, announced for the first time that it was developing an unmanned aerial vehicle (UAV) for parcel delivery (Rose, 2013). These UAVs, commonly known as drones, showed great promise to revolutionize the logistics industry, due to the fact that they are, in contrast to usual delivery vehicles, not bound to the road network but have the ability to travel via the air. In addition, drones do not need the guidance of personnel and, as a result, drones could save significant time and operating costs for a logistics operation.

An initial cost-benefit analysis into Amazon's *Prime Air* operations confirmed this potential (Welch, 2015). Welch (2015) concluded that not only do the revenues from using UAVs in the logistics system outweigh the start-up and operational costs, but also that the incorporation of UAVs would lead to a competitive advantage due to faster delivery and greater customer satisfaction. In recent years, however, Amazon's *Prime Air* operation has had some difficulties in getting its drone deliveries off the ground and it is speculated that it will not be able to deliver on its initial promise (Kersley, 2021).

Other companies on the other hand, such as DHL and Alphabet, are making great progress in drone delivery (Dell'Amico, Montemanni, & Novellani, 2021). While both Alphabet and DHL have begun testing, DHL's Parcelcopter has already completed a 3-month successful trial of delivering medical and urgent goods to a German Island (DHL, 2016).

Commercial use of drones is hardly limited to parcel delivery. Manna, a company founded in 2019, had, as of August 2021, an average of around 2000 successful food delivery flights by UAVs daily with an average service time below three minutes (Koetier, 2021). After successfully launching in Dublin, Manna is now planning to expand their operations to six other European countries (Keane, 2022). Other domains, where drones are currently used, include drug deliveries, agriculture, entertainment, and surveillance (Rao, Gopi, & Maione, 2016). While many of these objectives are open to academic research, the scope of this thesis is limited to the use of drones for parcel deliveries.

Using drones for deliveries also has potential benefits for society. Drones are electric and, as a result, do not directly pollute the air or emit greenhouse gasses while flying. However, assessing the environmental impact of drones is very complex and depends on the deployment of drones (Kellermann, Biehle, & Fischer, 2020). Goodchild and Toy (2018) suggest a blended approach of drones and trucks delivering parcels: trucks should deliver to further addresses, while drones can be used for nearby customers since drones have a small payload capacity and short-range because of limited battery capacity.

Using the same reasoning, Murray and Chu (2015) suggest combining a UAV on a truck, as this ensures the advantages of both modes of transport. The truck will function both as a delivery vehicle, as well as a mobile depot from which the UAV can leave to deliver to nearby customers. This approach uses the ability of the truck to cover large distances and the ability of the drone to quickly deliver parcels to nearby customers. For

this reason, it is expected that this approach will decrease the environmental impact of the truck. However, solid scientific evidence is, to the best of our knowledge, still lacking.

There are, on the other hand, also great concerns about the expansion of commercial drones. UAVs have a risk of air collisions, malfunctions or misuse by criminal or terrorist organizations and can, as a consequence, do severe harm to individuals (Smith, 2015; Stöcker, Bennett, Nex, Gerke, & Zevenbergen, 2017). To minimize these risks, European legislation is strict but still evolving. More importantly, from a commercial perspective legislation is moving in the right direction (Alamouri, Lampert, & Gerke, 2021). The legislation now provides detailed documentation and standardized procedures to acquire flight permission, which makes investing in drones more reliable.

Due to these advancements in technology and legislation, it becomes interesting to research the incorporation of drones in logistic systems. While extensive literature and several mathematical formulations already exist on UAVs and trucks cooperating in a logistics system, so far a probability of failure for drone delivery, due to for example a collision or malfunction, has not yet been incorporated into a model. Since drone failures are likely to occur and can have large consequences on the viability of using drones in logistics systems, this thesis will try to answer the following research question:

How does the probability of a drone failure impact the profitability of drones in logistics systems?

To answer this question we will consider the delivery of drones as stochastic, meaning it is uncertain whether a planned drone delivery will be successful.

In Section 2 a literature review is provided to assess the current scientific knowledge. Next, in Section 3 the problem at hand and its parameters are defined and explained. To obtain good solutions in reasonable time an Adaptive Large Neighbourhood Search (ALNS) algorithm is used and discussed in Section 4 based on the paper from Sacramento et al. (2019). Thereafter, in Section 5 the results from our implementation of the ALNS algorithm will be discussed. Lastly, we will conclude what the effects of the probability of destruction are on the viability of using drones in logistics systems in Section 6.

2 Literature review

The well-known traveling salesman problem (TSP), explained by, for example, Applegate, Bixby, Chvatal, and Cook (2011), is not very applicable to UAVs. In the TSP a vehicle or a salesman has to serve as a set of customers on a single route, while minimizing, for example, the costs of the route. For a UAV, however, this formulation does not seem practical. Due to the limited battery and payload capacity of the drone, it is in many cases infeasible to let the drone visit several customers in a row.

For this reason, Murray and Chu (2015) were the first to create a variant on the TSP that allows for the incorporation of a drone. They presented the flying sidekick traveling salesman problem (FTSP): a TSP where

the delivery truck is accompanied by a UAV that can leave the truck to deliver a package to a customer. After delivering a package, the drone returns to the truck at a different location. This formulation is a more realistic scenario for drones because the truck driver is able to change the batteries of the UAV after every delivery.

Secondly, they provided another formulation where a fleet of drones and one truck work independently from the same depot. In this problem, which was named the parallel drone TSP (PDTSP), the truck follows a TSP route and in the meantime, the drones deliver packages to other customers from the base depot. Their paper started a new branch of operations research, where drones are incorporated into traditional operations research formulations.

A logical extension to the FTSP was provided by Sacramento et al. (2019). They kept the same scenario: a single truck is accompanied by a drone. However, they extended the formulation to account for more than one truck. They defined this problem as a vehicle routing problem with drones (VRP-D). A number of trucks all accompanied by a single drone needs to serve a set of customers. Every truck has a single route starting and ending at the depot. Drones can leave from their respective truck to deliver a package to a single customer, whereafter it is retrieved by its a truck at a different node. Because of the realistic applicability of their formulation, we will use their formulation as the basis for our research.

Other interesting formulations, though not considered in this paper, are for example the multi-traveling salesmen problem with drones (mTSPD) presented by Kitjacharoenchai et al. (2019) or the hybrid vehicle routing problem with drones (HVRPD) presented by Karak and Abdelghany (2019). Kitjacharoenchai et al. (2019) present a variation of a multi-traveling salesman problem (mTSP), where a number of trucks has to deliver to a set of customers, while simultaneously a fleet of drones delivers packages to customers. These drones can be launched and retrieved by either a truck or the home depot. Moreover, drones are not assigned to any truck but can return to a different truck than they have departed from.

In the HRVPD a single truck carries a fleet of drones and packages (Karak & Abdelghany, 2019). The truck acts as a mobile depot from which drones can be launched to deliver parcels to customers. In Karak and Abdelghany's (2019) formulation drones can first deliver packages and then to pick up packages at other nodes that need to be returned. While these different formulations all have their advantages and disadvantages, it is yet to be seen which design is the most practical.

Stochasticity of delivery has, to our knowledge, not yet been implemented into a problem formulation with drones. However, stochasticity has been widely included in formulations of the VRP (Gendreau, Laporte, & Séguin, 1996). Common examples of including stochasticity are: stochasticity of demand or stochasticity of travel time. Another type of stochasticity, described by Bertsimas (1988), considers the set of customers itself as stochastic, customers can appear on the route with a certain probability p_i for every customer i. Bertsimas (1988) termed this formulation the probabilistic VRP. Later, this formulation was commonly referred to as the VRP with stochastic customers (VRPSC) (Gendreau et al., 1996).

This latter stochastic formulation shows similarities to the formulation considered in this paper. However, instead of customers existing or spawning with a certain probability, delivery of a customer by drone is stochastic, i.e., a customer is successfully served by a drone with a certain probability p. As a consequence of the stochastic element, it is for both formulations uncertain, whether a customer can be served successfully.

Solving the FTSP, VRP-D or VRPSC to optimality with exact methods in reasonable running times can only be done for small instances (Sacramento et al., 2019). Therefore, heuristic solutions are used to get a relatively good solution in a reasonable running time. In this paper, we will use an extension of the large neighborhood search (LNS), which is called an adaptive large neighborhood search (ALNS)

The LNS algorithm, first proposed by Shaw (1998), destroys and repairs a current solution to find new local minima and improve the solution (Pisinger & Ropke, 2019). Since an LNS algorithm searches a large neighborhood the algorithm is relatively time-consuming per iteration. However, it is also able to find better solutions at every iteration.

The ALNS algorithm, first defined by Ropke and Pisinger (2006), extends the LNS algorithm by allowing several destroy en repair methods to be chosen (Pisinger & Ropke, 2019). The methods are dynamically assigned weights based on their performance and methods are chosen based on their relative weights. As a result, methods that helped to improve the solution will have a higher probability to be selected again by the algorithm.

3 Vehicle routing problem with stochastic drone delivery

In this section we will define and discuss the vehicle routing problem with stochastic drone delivery. First, we will define and discuss the problem at hand in Section 3.1. Then, we will discuss the values of the parameters that we use in Section 3.2.

3.1 Problem description

In the VRP-D a set of homogenous trucks, all equipped with a single drone , have to deliver packages to a set of customers C. In the original problem, defined by Sacramento et al. (2019), each customer has to be served exactly once by either a truck or a drone. In this formulation, every customer has to be served with a probability of at least δ, which will be called the individual service requirement. Additionally, on average every customer has to be successfully served with a probability $\gamma \geq \delta$, which is the average service requirement. Each truck starts and ends its route at the home depot. The drone can leave its respective truck along the route to serve a customer, whereafter it has to return to its truck. Every move of a vehicle V, which can be a drone D or a truck T, from node i to j has a travel time $\tau^V_{i,j} = \frac{d_{i,j}}{v^V}$, where $d_{i,j}$, v^V represent the distance and travel speed respectively.

A truck will always serve a customer successfully with a probability equal to one. A drone, on the other hand,

has a probability $p \leq 1$ for each independent visit to successfully deliver and return. For this reason, we call this formulation the vehicle routing problem with stochastic drone delivery (VRPSDD). As discussed earlier, there are many possible reasons for a drone to fail at delivering its package. Therefore in this formulation, this probability is included.

Boeing (2021) concludes in their summary report that only around ten percent of all airplane accidents happen while cruising and that the other ninety percent occur, while a plain is taking off, ascending, descending, or landing. For UAVs statistics on when accidents occur are still missing. However, we believe that drone accidents will similarly occur in these stages, as drones are then close to obstacles, like buildings or trees. In the air, accidents are less likely to occur since there are fewer obstacles. As a consequence, we assume that the probability of a failure $\bar{p} = 1 - p$ and p is constant and does not depend on the time flown or distance traveled.

A truck is only equipped with one drone. Therefore, as soon as a drone failure occurs, the drone will not be able to visit any customers later on the route. The probability that the customer of the i-th drone visit successfully gets his package is then equal to p^i. If a drone delivery fails, it is assumed that the package will be lost and therefore, that the truck will not be able to deliver a package to that customer. Thereafter, the truck will take over the planned drone sorties. This will be done in the following way. At every subsequent drone launch position i of drone sortie (i, j, k), where i represents the launch node, j the customer's node, and k the recovery node, instead of the drone visiting the customer j, the truck travels to j, whereafter it returns to its initial route. This will be called a route adjustment and this will be considered the simple recourse policy. This adjustment might not be optimal but practical to implement in reality for every drone failure. Moreover, it is possible to calculate the expected costs of this adjustment. It is important to note that the route after the adjustment could exceed the maximum route time T_{max}. This, however, can only happen after a failure and will be seen as an exceptional situation. In a later section, Section 4.5, however, we will implement a more advanced recourse policy, which uses a route reoptimization to minimize the adjustment costs when an actual failure occurs.

Every drone visit has to respect both the payload and battery capacity e of the drone. Because drones can mostly hold one package, they can only serve one customer at a time. It is allowed, however, for the drone to depart and return several times along the truck route. There are two ways a drone can deliver a package to its customer. It can either drop off packages by lowering them from the drone or it can land at a platform of a customer and detach the package there. Similar to other papers that focus on the technical aspects of drone delivery, for example, Miranda et al. (2022), we assume that all customers have a landing platform on which a drone lands to deliver a package. As a consequence, we can assume that drones are able to land at the recovery position k of a sortie and can wait there until a truck picks them up. Therefore, the total travel and service time of the truck route between node i and k does not have to satisfy the drone's endurance. Only the total travel, service s^D, launch s^L and recovery s^R time of the drone need to be smaller or equal to the drone's endurance. The following, thus, needs to hold for a feasible sortie $s = (i, j, k)$:

$$s^L + \tau^D_{i,j} + s^D + \tau^D_{j,k} + s^R < e \tag{1}$$

The order of operations of the truck at a launch or recovery node i determines the time at node i in the route. We assume that the truck driver will always first serve customer i before launching the drone if node i represents a launch node and will always first recover the drone before serving customer i if node i represents a recovery node, this ensures that the time the truck driver is away from the drone is minimized. However, this might come at the cost of having a larger total route time.

There is a cost $c^V_{i,j}$ associated for every move dependent on the distance from node i to j of the truck T or drone D. Denoting mc a mile to kilometer converter, fc fuel consumption per kilometer, and fp the fuel price per liter. The travel costs in euro of the truck are equal to $c^T_{i,j} = d_{i,j} \cdot mc \cdot fc \cdot fp$. The travel costs of the drone will then be set to a factor $a < 1$ times the truck's travel costs. This factor will be smaller than one since we assume that drones will be have lower travel costs than drones.

Trucks have a limited total travel time for their route since delivery drivers have a limited working period T_{max}. When drones are recovered by the truck, they are transported by the truck and do not incur any extra costs. The objective is to minimize the total costs of serving all customers, the total costs consist of the total truck's traveling costs and the sortie costs SC. Sorties incur the following costs depending on whether the drone delivery has been successful. Let P be the set containing all feasible sorties then the costs associated with a successful delivery for the n-th drone visit are equal to:

$$S_n = c^D_{i,j} + c^D_{j,k} \tag{2}$$

The probability that the delivery of the n-th sortie is successful, is equal to p^n. Secondly, if the drone has a failure on the n-th delivery we incur the fixed costs F for the repair costs of the drone and additional delivery costs. These additional delivery costs are set to the traveling costs of a truck from the depot to a customer and back $c^T_{0,j}$, where node 0 represents the depot and j the customer of the sortie. In reality, the costs of delivering a package to that customer will be lower, since the customer could be incorporated into a route. However, this will provide us with an upper bound on the actual costs. The total fixed costs of the n-th drone visit are equal to:

$$F_n = F + 2c^T_{0,j} \tag{3}$$

The chance of having a failure at the n-th drone sortie will be equal to $p^{n-1}(1-p)$. Lastly, drone sorties incur adjustment costs if the drone fails before it is set to serve the customer at that sortie. These adjustment costs are equal to: the truck's traveling cost of moving from the launch position i to the customer j; plus the traveling costs of moving from customer j to the next node in the route of the truck $i+1$; minus the traveling costs of moving the truck from node i to k, since the truck does not travel from i to k anymore. The

adjustment costs become:

$$A_n = c^T_{i,j} + c^T_{j,i+1} - c^T_{i,i+1} \tag{4}$$

Note that $i+1$ represents the node after i in the truck route, which not necessarily has to be the recovery node. The probability of incurring adjustment costs at the n-th sortie is equal to $(1-p^{n-1})$. Taking into account all these costs, the expected total sortie costs of the i-th sortie are equal to:

$$SC_i = p^i S_i + p^{i-1}(1-p)F_i + (1-p^{i-1})A_i \tag{5}$$

Due to the fact that the sortie costs are asymmetric for $p < 1$, the total costs would become also asymmetric for $p < 1$. To prevent asymmetric costs the total costs of a truck route are determined by the traveling costs of the truck, which are symmetric, plus the minimum of both orientations of the sortie costs. This ensures that the costs are set to the minimum of both route orientations and thus, the route costs become symmetric again.

3.2 Parameters

There are many parameters related to the abilities of the drone and truck relevant for the VRP-D. To be able to compare this implementation of the VRP-D with Sacramento et al. (2019)'s formulation, most of the values of the parameters will be selected from their paper. They have tried to select parameters that are close to reality. However, due to the fact that drone delivery is not widely used yet and is still likely to improve, the parameters might be inaccurate. An overview of the parameters can be found in Table 1.

In addition, we set the average service requirement γ and the individual service requirement δ to 0.999 and 0.99 respectively. These are reasonably low service level guarantees, to ensure that it is still feasible to let customers be served by drones.

The production costs of an octocopter drone similar to Amazon's, the DJI spreading wings SJ100, are between 1000-3000 US dollars. Including a software cost estimation of around 2000 US dollars, we estimate the total production cost of a drone to be 4000 US dollars (Keeney, 2021). Since the damage after a drone failure differs, we set the estimated repair costs to halve of the production cost, which is 2000 US dollars. To get a nice round number, we assume a favorable exchange rate of one-to-one and set the fixed repair costs F of the drone to 2000 euros.

.

4 Methodology

Heuristics are used to find good solutions in reasonable running times. In this case, we will use an ALNS algorithm based on Sacramento et al. (2019). In section 4.1 the ALNS algorithm is described. Next, we

Table 1: Parameters of VRP-D

Parameter	Notation	Value
Individual service requirement	δ	0.99
Average service requirement	γ	0.999
Launch time	s^L	1 min
Recovery time	s^R	1 min
Service time truck	s^T	2 min
Service time drone	s^D	1 min
Truck speed	v^T	35 mph
Drone speed	v^D	50 mph
Drone endurance	e	30 min
Truck capacity	Q	1400 kg
Drone capacity	Q^D	5 kg
Fuel price	fp	1.13 €/l
Fuel consumption	fc	0.07 l/km
Miles converter	mc	1.61 km/mi
Drone factor cost	α	0.1
Maximum route duration	T_{max}	8 h
Fixed costs drone failure	F	2000 €

describe how to obtain an initial solution for the ALNS algorithm in Section 4.2. In the following section, Section 4.3, the random and cluster destroy methods are explained. Next, in section 4.4 three repair methods are discussed. At the end of this section, we will discuss a recourse policy to optimize the route further after a failure has occurred, see Section 4.5.

4.1 Adaptive large neighbourhood search

In an ALNS algorithm destroy and repair methods are iteratively and statistically chosen to improve the current solution. Destroy methods destroy part of the current solution, while repair methods rebuild the partial solution to a new feasible solution. At every iteration a destroy and repair method is statistically selected based on their relative performance, i.e., methods that improved the current solution more often will have a higher probability to be selected again.

Table 2: Scores for destroy and repair methods

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

We define Ω^- and Ω^+ as the set of respectively the destroy and repair methods. At each iteration a repair and destroy method is chosen probabilistically based on their respective weights. Normally, the weights of both the repair methods and desroy methods are updated based on their relative performance. In this algorithm, however, only the repair methods' weights are updated according to their performance. Let w_{ij} be the weight of method i after iteration j. Initially, the weights are initialized with equal probability. The weight of repair method i used in iteration j is then updated by their score Ψ, defined in Table 2, relative to the current

solution and reaction factor $\rho \in \{0, 1\}$ according to the following formula (Sacramento et al., 2019):

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

Note that if a method is not used in iteration j its weight remains the same for the next iteration. A destroy, repair method i will then be selected in iteration j with a probability equal to $\frac{w_{i,j}}{\sum_{k \in \Omega} w_{k,j}}$, where Ω equals Ω^-, Ω^+ respectively. To prevent solutions are randomly updated, new solutions have to be accepted based on an acceptance criterion. In this case a solution s^t with a better objective value than the current solution s will always be accepted. However, new solutions with an objective value larger than the current solution are only accepted with the following probability (Sacramento et al., 2019):

$$e^{\frac{f(s)-f(s^t)}{T}}$$

Here $f(s)$ denotes the objective value of solution s and T denotes a temperature parameter that is used to decrease the acceptance probability over time. We initialize $T = T_{st}$ and then decrease T linearly over time, converging to zero at t^{max}, the time the algorithm is terminated. This ensures that at the start of the algorithm a wider range of possible solutions are accepted, while at the end worse solutions are only accepted with a small probability. The temperature parameter is then updated with the following formula (Sacramento et al., 2019):

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

Here t^{elap} is the elapsed time of the ALNS algorithm. The algorithm is terminated as soon as $t^{elap} \geq t^{max}$. Lastly, different from other ALNS algorithms, this algorithm resets the current solution to the best solution found thus far if there has not been made an improvement on the best solution after a certain number of iterations (Sacramento et al., 2019). The pseudocode of the ALNS algorithm is given in Algorithm 1.

4.2 Initial solution

To use the ALNS algorithm we first have to find an initial solution. The initial solution will be constructed by using heuristics with the following three steps. Firstly, a Nearest Neighbour heuristic will be used for the trucks to make sure all customers are visited. A truck route is initialized and iteratively the customer closest to the last position in the current route is added to the truck's route until either, the capacity or time constraint of the truck becomes restrictive, or all customers have been served. In case of the former, a new truck will be added to the solution and the same procedure is repeated until all customers are served.

In the second step, the current VRP solution is improved by an iterated local search algorithm (ILS). At

Algorithm 1: Pseudo-Code for the ALNS algorithm

```
input : Initial Temperature: T_st,
        Max iterations without improvement: noImpvMax,
        Time limit: t^max
s ← InitialSolution();
s* ← s;
noImpv ← 0;
while t^elap < t^max do
    Select a destroy method d() and a repair method r() from Ω⁻ and Ω⁺;
    s^t ← r(d(s)) ;
    T = T_st(1 − t^elap/t^max) ;
    if Random(0,1) < exp((f(s)−f(s^t))/T) then
        s ← s^t
    if f(s) < f(s*) then
        s* ← s;
        noImpv ← 0;
    else
        noImpv ← noImpv + 1;
        if noImpv > noImpvMax then
            s ← s*;
            noImpv ← 0;
    Update scores of Ω⁻ and Ω⁺ based on acceptance criteria
return s*;
```

every iteration, this ILS considers all the intra- and inter-route two-opt moves of the current solution and implements the two-opt move with the largest cost saving. This is repeated for a total of one thousand iterations.

Thirdly, the drones are added to the current solution by the following procedure. First, the set D of possible drone customers on the route of a truck is identified. Secondly, for all customers $c \in D$, iteratively the customer gets removed from the truck route and is possibly added as a drone customer to a route. The set of all possible sorties P_c for customer c is identified and Algorithm 2, adapted from Sacramento et al. (2019), finds the best feasible sortie. A feasible sortie can of course not coincide with other sorties and has to respect both the individual and average service requirement. This sortie is then added to the solution if the total costs of the route including the sortie are smaller than the threshold η. The threshold is initialized by the total costs of the route before customer c was removed from the route and is updated, every time a better sortie is found, by the current total costs. If it is no longer feasible to let a customer in D be visited by the drone, the customer is added again to the truck route by the `BestTruckInsertion(c, s)` method. This is repeated until all customers in D have been considered.

4.3 Destroy methods

Two different destroy methods will be used to destroy part of the current solution. A destroy method will at least remove β customers from the current solution. Let δ represent the fraction of the total customers that is preferably removed, and let c_{low}, c_{lim} present a lower and upper bound for β respectively. Then β is

Algorithm 2: *FindSortie*(c, s, η) function that finds 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
BestSortie = ∅;
for Each Route in s do
    if # drone sorties of route + 1 ≤ MaxDroneSorties then
        if Capacity(Route) + q_c < Q then
            for Pair Positions(i, k) in Route where i < k do
                Construct sortie p = (i, c, k) with launch-position i, delivery-position c and recovery-position k;
                Check feasibility for sortie p;
                if averageServiceLevel(s ∪ p) < γ then
                    if s^L + s^R + τ^D_ic + τ^D_ck + Se^D_c < e AND f(s) + CostSortie(p) < η then
                        BestSortie ← p;
                        Update η;
return BestSortie;
```

determined by the following formula (Sacramento et al., 2019):

$$\beta = \min(\max(c_{low}, \delta|C|), c_{lim}) \tag{6}$$

In most cases δ determines the number of customers to be removed. However, for very large or small instances the lower and upper bound ensure that the values of β remain sensible. We set the parameter c_{low} to a random integer between 1 and 3, and the upperbound c_{lim} to 40.

The two destroy methods described below will be consistently chosen with equal probability. Therefore, the adaptive part of the ALNS algorithm will only be used for the repair methods.

4.3.1 Random destroy

The first destroy method, adapted from Sacramento et al. (2019), will remove customers randomly from the current solution until at least β customers are removed. If a removed node represented a launch or recovery node for the drone, the drone customer will also be removed. Thus, it is possible that more than β customers are removed. However, at most $\beta + 2$ customers can be removed since a node can only be a launch position for one sortie and a recovery position for one sortie. Lastly, if at the end of the algorithm the average service requirement is not satisfied anymore, iteratively a random drone customer is selected and changed into a truck customer. The drone sortie is removed from the route and the customer is inserted by the `BestTruckInsertion(c, s)` method that inserts customer c into solution s by a truck visit with the lowest additional costs. As a service requirement was not present in Sacramento et al. (2019), this last part has been added to the original algorithm to ensure a feasible solution.

Algorithm 3: Cluster Removal of Customers

input : Current Solution: s,
number of customers to remove: β

$c_1 \leftarrow$ `RandomCustomer(`s`)` ;
remove c_1 from s;
$removed \leftarrow c_1$;
while $removed < \beta$ **do**
  $c \leftarrow$ `RandomCloseCustomer(`c_1, s`)` ;
  if c *is Launch and/or Recovery Position* **then**
    remove drone customers associated with c from s
  remove c from s ;
  update $removed$;
while *averageServiceProbability(s)* $> \gamma$ **do**
  Select random drone customer c;
  Remove customer c and sortie, and insert the customer by `BestTruckInsertion(c, s)`
return s;

4.3.2 Cluster destroy

The cluster destroy method, adapted from Sacramento et al. (2019), removes customers in a cluster around a randomly chosen focal point. Let customer c_1 be the randomly chosen focal point, then, the next customer to be removed is randomly chosen from the subset of the two closest customers to c_1. This randomness is added to the selection procedure to avoid that the same partial solution will be reached again. This is repeated until at least β customers are removed, see Algorithm 3. Again, if the average service requirement γ is not satisfied after removing β customers, random drone customers are changed into truck customers by the `BestTruckInsertion(c, s)` method until this requirement is satisfied.

4.4 Repair methods

After a solution has been destroyed, repair methods are used to rebuild the partial solution. Let the set X denote the set of customers that have been removed, then the repair methods add these customers iteratively to truck routes. Repair methods ensure that customers are feasibly added. If customers can not feasibly be inserted into existing routes a new truck route is added, such that a repair method is always able to create a feasible solution. We will present three repair methods that were adapted from Sacramento et al. (2019) to fit our problem.

4.4.1 Greedy truck-first sortie-second repair method

The first repair method to be discussed, is the greedy truck-first sortie-second heuristic, which consists of two steps. A random customer from the set X is first added to a truck route with the `BestTruckInsertion(`c, s`)` function. This function finds the cheapest insertion for the customer into a truck route. This is repeated until all customers from X are added to a truck route.

In the second phase, a random customer is selected from the set C. If this customer is currently served by a truck the `FindSortie(`c, s, η`)` function is used to find the best feasible sortie for that customer. Only if the total costs are decreased by serving that customer by a drone, the drone sortie is added to the route.

Algorithm 4: Greedy truck-first sortie-second repair method

```
input : Partial Solution: s,
        set of removed customers: X
while X ≠ ∅ do
    c ← RandomCustomer(X);
    X = X \ {c} ;
    BestTruckInsertions(c, s);
C = AllCustomers(s);
while C ≠ ∅ do
    c ← RandomCustomer(C) ;
    C = C \ {c} ;
    if q_c ≤ Q^D AND Type(c) = Truck then
        s' ← s ;
        η = f(s') ;
        s ← s \ {c} ;
        p ← FindSortie(c, s, η);
        if p ≠ ∅ then
            s ← s ∪ {p}
        else
            s ← s';
return s;
```

Otherwise, the route is not changed and the procedure repeats until all customers have been considered.

4.4.2 Nearby-area truck-first sortie-second repair method

A variation on the previous heuristic is the nearby-area truck-first sortie-second repair method, which was adapted from Sacramento et al. (2019). This method works similarly but, instead of inserting a customer into the route with lowest cost, the customer is inserted after a random position within a five mile radius. While Sacramento et al. (2019) did not specify how to insert a customer c, if the set of nodes within a five mile radius is empty, we will insert those customers by the `BestTruckInsertion(c, s)` method. In the second phase of the algorithm, iteratively a random truck-customer from the set C is selected and all possible sorties are identified. In this algorithm the truck customer is inserted by a random sortie that is selected out of the set of feasible sorties that does not increase the cost of the partial solution with more than ten percent. Due to the fact that this method does not necessarily selects the lowest cost insertions, it can be seen as a weaker version of the previous algorithm. However, it does find a wider range of solutions, which can help escaping local minima.

4.4.3 Closest insertion repair method

The third repair method, adapted from Sacramento et al. (2019), tries to insert customers in the largest cost saving way considering both truck and drone visits. First, the route that lies closest to a randomly selected customer from X is identified. Then the `AttemptBestInsertion(`c, s`)` function is used to insert the customer into that route. This function finds the largest cost saving insertion into that specific route considering both truck and drone insertions. If the function is unable to feasibly add the customer into the route, the customer is added to the set X_N. When all customers from X have been considered, the customers in set X_N that

could not be feasibly added, are added by the greedy truck-first sortie-second algorithm defined earlier, see Algorithm 4.

Algorithm 5: Closest insertion repair method

```
input  : Partial solution: s,
         set of removed costomers: X
X_N = ∅;
while X ≠ ∅ do
    c ← RandomCustomer(X);
    X = X \ {c};
    c' ← NearestCustomer(c, s);
    r ← RouteOf(c');
    if AttemptBestInsertion(c, s) = false then
        X_N = X_N ∪ {c};
if X_N ≠ ∅ then
    s ← RepairTruckFirstSortieSecond(X_N, s) (Algortihm 4);
return s;
```

4.5 Recourse policy

So far we have considered that the truck takes over the planned drone customers with a route adjustment after a failure occurs. As discussed in Section 3.1, this would be done in the following way for the all still to be completed drone sorties $s = (i, j, k)$: as soon as the truck has visited the customer at the launch position i, it will visit the customer at node j itself, since the drone can not be launched anymore. Thereafter it visits the customer at node $i + 1$, which is the node planned after node i in its initial route. The truck then continues its initial route until it reaches a new launch position i of a scheduled sortie and the procedure is repeated.

This route adjustment, however, could be far from optimal but is very simple to implement for a truck driver. For this reason, this will be considered the simple recourse policy. In this section we describe a more advanced recourse policy that reoptimizes the route with a cheapest insertion algorithm, whenever a failure occurs.

When a drone crashes, the truck has to take over the planned drone customers from that point. We assume that the truck will notice the drone failure at the recovery position k of the drone sortie that has failed. Therefore, from that point onwards the problem reduces to a TSP, where the start and end node of the route are not equal. To reoptimize this route, we will use a variation on the well-known cheapest insertion algorithm, described by for example Rosenkrantz, Stearns, and Lewis (1977). We start with the route from k to the depot and first insert all the drone customers by the cheapest insertion algorithm in the current route. Let C^l denote all customers present in the route after k, which are all customers still to be visited after the drone failure. Next, the route is improved by removing and reinserting a customer $j \in C^l$ by the `findCheapestInsertion(`R, j`)` function. This function considers all insertions between the start node k and the depot 0, and inserts customer j with the cheapest cost. This is repeated for all customers until no improvements can be made, see Algorithm 6.

Since the cost savings of the advanced recourse policy depend on the sortie at which a failure occurs. We

Algorithm 6: Route improvement algorithm for the recourse policy

```
input : Remainder of the route R after recovery position k of the failed drone sortie,
        Set S of the remainder of sorties still to be visited
for Every sortie s = (i, j, k) ∈ S do
    Insert customer j into route R to findCheapestInsertion(R, j);
smallestCost = totalCost(R);
Determine the set of customers C^l in the route after k;
while currentCost < smallestCost do
    smallestCost = currentCost;
    for Every customer j ∈ C^l do
        Insert customer j in route R to findCheapestInsertion(R, j);
    currentCost = totalCost(R);
return new route R;
```

will calculate for several solutions the cost savings S_s at each sortie s of the advanced recourse policy in comparison to the simple recourse policy discussed earlier. These savings will be used to assess the performance of our recourse policy. Remember that the probability that a failure occurs at the i-th sortie is equal to $\mathbb{P}(F_i) = (1-p)p^{i-1}$. We can calculate the expected cost savings TS_r of each route $r \in R$, denoting the event of a failure at sortie s as F_s, with the following formula:

$$\mathbb{E}[TS_r] = \sum_{i}^{n_r}(S_i \cdot \mathbb{P}(F_i)) = \sum_{i}^{n_r}(S_i \cdot (1-p)p^{i-1})$$

Note that $S_i = 0$ if there is no failure at sortie i and that n_r is the total amount of sorties in route r. Then the probability that a failure occurs in a route is equal to one minus the probability that no failures occur in the route: $\mathbb{P}(F_r) = 1 - p^{n_r}$. The cost savings given that a failure F_r has occurred in route r can be calculated with the following formula:

$$\mathbb{E}[TS_r|F_r] = \sum_{i}^{n_r}\left(S_i \cdot \mathbb{P}\left(F_i|F_r\right)\right) = \sum_{i}^{n_r}\left(S_i \cdot \frac{(1-p)p^{i-1}}{1-p^{n_r}}\right) \tag{7}$$

Let R denote the set of all routes in a solution. The total expected cost savings of the routes combined can be calculated with the following formula:

$$\mathbb{E}[\sum_{r\in R} TS_r] = \sum_{r\in R}\mathbb{E}[TS_r]$$

Let now denote F the event of at least one failure in all combined routes of the solution and let $F_r^{\complement}$ the event of not having a failure at route r. The probability to not have any failures in R is given by $\mathbb{P}(F_R^{\complement}) = \prod_{r\in R}\mathbb{P}(F_r) = \prod_{r\in R} p^{n_r}$. We partition the event F in the disjoint events F_r and $F_r^{\complement}$, so that we can calculate the total expected cost savings of solution conditional on that a failure has occurred with the

following formula:

$$\mathbb{E}[\sum_{r\in R} TS_r|F] = \sum_{r\in R}\mathbb{E}[TS_r|F] = \sum_{r\in R}\Big(\mathbb{E}[TS_r|F\cap F_r]\cdot\mathbb{P}(F_r|F) + \mathbb{E}[TS_r|F\cap F_r^{\complement}]\cdot\mathbb{P}(F_r^{\complement}|F)\Big) = \sum_{r\in R}\left(\mathbb{E}[TS_r|F_r]\cdot\frac{\mathbb{P}(F_r)}{\mathbb{P}(F)}\right) = \sum_{r\in R}\left(\mathbb{E}[TS_r|F_r]\cdot\frac{(1-p^{n_r})}{1-\prod_{u\in R}p^{n_u}}\right)$$

Again note that $\mathbb{E}[TS_r|F_r^{\complement}] = 0$ because total savings of route r are equal to zero if there is no failure in route r. Both the expected savings and expected savings conditional on a failure will be used to assess the effectiveness of the advanced recourse policy. Additionally, we will calculate the expected route costs RC of a solution given a failure. We define RC as the total incurred costs of the combined routes minus the fixed failure costs of a sortie F_s. As a result, RC represents only the traveling costs of a solution. Let RC_r be the route costs of route r, then $\mathbb{E}[RC_r|F]$ can be calculated using Equation (7), by replacing TS_r by RC_r and S_s by the costs of a route when sortie s fails. Then the expected route costs of a solution given a failure can be calculated with the following formula:

$$\mathbb{E}[\sum_{r\in R} RC_r|F] = \sum_{r\in R}\mathbb{E}[RC_r|F] = \sum_{r\in R}\Big(\mathbb{E}[RC_r|F\cap F_r]\cdot\mathbb{P}(F_r|F) + \mathbb{E}[RC_r|F\cap F_r^{\complement}]\cdot\mathbb{P}(F_r^{\complement}|F)\Big) = \sum_{r\in R}\left(\mathbb{E}[RC_r|F_r]\cdot\frac{\mathbb{P}(F_r)}{\mathbb{P}(F)} + \mathbb{E}[RC_r|F_r^{\complement}]\cdot\frac{\mathbb{P}(F_r^{\complement})\cdot\mathbb{P}(F_{R\setminus r})}{\mathbb{P}(F)}\right) = \sum_{r\in R}\left(\mathbb{E}[RC_r|F_r]\cdot\frac{(1-p^{n_r})}{1-\prod_{u\in R}p^{n_u}} + \mathbb{E}[RC_r|F_r^{\complement}]\cdot\frac{p^{n_r}(1-\prod_{t\in R\setminus r}p^{n_t})}{1-\prod_{u\in R}p^{n_u}}\right)$$

Note that in this case, if there is no failure at route r, there are still travel costs in route r and therefore, the second term does not disappear. These expected route costs given a failure will be used to assess the effectiveness of the advanced recourse policy by inspecting the expected savings given a failure relative to the expected route costs given a failure.

5 Experimental results

In this section, we will present the results of our VRPSDD. First, we will discuss the instances used for our methods in Section 5.1. We will then start comparing our results, when there is a guaranteed successful delivery, i.e., $p = 1$, with those of Sacramento et al. (2019) in Section 5.2. Next, we will discuss the effect of the failure probability $\bar{p} = 1 - p$ on the profitability of drones in Section 5.3. First, we will discuss the instances used for the VRPSDD, Section 5.3.1, then we will present and discuss the results of the VRPSDD, Section 5.3.2, and lastly, we will assess the performance of our solutions when the probability used in the model does not equal the true failure probability, Section 5.3.3. In the last results subsection, Section 5.4, we discuss the performance of the advanced recourse policy. All of the results are obtained by an Intel(R)

Core(TM) i5-8265U processor.

5.1 Instances

We will use the instances generated by Sacramento et al. (2019) for a good comparison of our results. They initialized a number of customers n on a grid with a length and width of $2g$. for every customer they generated a random x and y coordinate between $\pm g$. The depot is always set in the middle of the grid at coordinate $(0, 0)$.

Secondly, they initialized the demand q_i of customer i in the following way. With a probability equal to 0.86 a customer is initialized with a demand from the uniform distribution between 0 and 2.27 kg. Otherwise, the demand is initialized from the uniform distribution between 2.27 and 68 kg. This is because Amazon claims that 86% of their package deliveries weigh less than 2.27 kg (Allain, 2013) and UPS reports that the maximum allowed weight of a package on a truck is 68 kg (UPS, 2017). All instances have a code $n.g.i$ where n represents the amount of customers, g the grid dimension and i the number of the instance with that grid and customer combination. All used instances can be found in Table 3.

5.2 Performance ALNS algorithm

We first assess the performance of our implementation of the ALNS algorithm. To do so we will compare our results with $p = 1$ to the results of Sacramento et al. (2019). Since no failure probability was included in Sacramento et al. (2019), we set the success probability to one to allow for a fair comparison.

To accurately compare our results, we set the parameters of the ALNS algorithm equal to the values of Sacramento et al. (2019). After a parameter tuning experiment described in Sacramento (2017), they determined the following values to be effective: initial temperature factor $T^*_{st} = 0.004$, factor of destruction $\delta = 0.15$ and non-improvement parameter $noImpvMax = 1000$ iterations. The initial temperature T_{st} is set to the objective value of the initial solution times T^*_{st}. This ensures that the initial temperature adjusts to the size of the instance.

The remaining parameters of the adaptive part of the algorithm ρ and the scores $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ are set to 0.9 and $\{33, 9, 13, 0\}$ respectively and we initialize the weights w_i of every repair method i to 9. Lastly, we run the ALNS algorithm for a maximum of 5 minutes and therefore, set t_{max} to 5 minutes. The results of our implementation can be found in Table 3. Each instance will be run ten times for different seeds.

Before comparing the results of both implementations of the ALNS algorithm, let us discuss the small differences in implementation. First of all, Sacramento et al.'s (2019) implementation assumes the truck has to arrive at the recovery position before the drone. While in our implementation the drone can land on a platform at the customer and wait for the truck to arrive. This allows the truck to visit more customers before it has to pick up the drone. The following constraint has, thus, been relaxed. Let $t^T_{i,k}$ denote the total

travel time of the truck route from i to k, including travel time and service time in between nodes, then every drone sortie (i, j, k) in the route has to satisfy the following constraint:

$$t_{i,k}^{T} \leq e \tag{8}$$

Secondly, while our implementation has three different repair methods, Sacramento et al. (2019) utilize a fourth repair method called heavy insertion. Heavy insertion first inserts all customers with a demand greater than the drone capacity with the best truck insertion method described in Section 4.4.1 and then inserts the rest of the customers with the closest insertion algorithm, see Algorithm 5.

Thirdly, Sacramento et al. (2019) improve the initial VRP solution by an ILS algorithm with three methods. These methods include an intra- or inter-route two-opt, relocation, or exchange move. We, on the other hand, only use a two-opt move. Additionally, they improve their initial VRP-D solution, by a string relocation algorithm, while our initial solution is immediately used in the ALNS algorithm.

Lastly, there are some small differences in the parameters of the ALNS algorithm. Sacramento et al. (2019) increase the initial temperature T_{st} for small instances, which they did not define, by ten percent to allow for a larger variety of solutions. Also, the values of the initial weights for the methods of the ALNS algorithm were unspecified. This, however, will likely not have a large effect, if the amount of iterations is large enough, as then the weights can adapt.

Let us now compare the results of Table 3 of both implementations. First of all, we see that both implementations are able to find the same objective value for small instances, $|C| < 20$, except for instances 12.5.2 and 12.20.4, which both have a higher value for our implementation. However, for larger instances, the differences in the smallest objective value are larger and in most cases, our smallest objective value is higher.

Table 3: Comparison of the performance of the ALNS algorithm

Instance	z^*		μ		Difference	σ		Iterations		$z^{initial}$	
6.5.1	1.0982	1.0982	1.0982	1.0982	0.00	0.000	0.000	10739814	41959813	1.3822	1.3368
6.5.2	0.8422	0.8422	0.8422	0.8422	0.00	0.000	0.000	9235810	42802018	1.0571	1.1084
6.5.3	1.2114	1.2114	1.2114	1.2114	0.00	0.000	0.000	9506438	44620683	1.3026	1.3372
6.5.4	0.9460	0.9460	0.9460	0.9460	0.00	0.000	0.000	12726278	44129451	1.0198	1.0198
6.10.1	2.4061	2.4061	2.4184	2.4061	0.51	0.039	0.000	12295129	38949492	2.8796	2.8796
6.10.2	1.6793	1.6793	1.6793	1.6793	0.00	0.000	0.000	12937848	37191908	2.6913	1.7702
6.10.3	1.3255	1.3255	1.3255	1.3255	0.00	0.000	0.000	14066338	38426647	1.9033	1.9033
6.10.4	1.4431	1.4431	1.4431	1.4431	0.00	0.000	0.000	12263820	40046320	2.5417	1.7345
6.20.1	2.6776	2.6776	2.6776	2.6776	0.00	0.000	0.000	12823071	39483597	4.4296	3.6704
6.20.2	4.3196	4.3196	4.3196	4.3196	0.00	0.000	0.000	10982472	42665143	5.4408	5.5610
6.20.3	3.8248	3.8248	3.8248	3.8248	0.00	0.000	0.000	11335483	43387965	5.6434	5.0516
6.20.4	3.6787	3.6787	3.6787	3.6787	0.00	0.000	0.000	12209879	38621176	5.6617	4.5477
10.5.1	1.6556	1.6556	1.6556	1.6556	0.00	0.000	0.000	4062999	25746784	1.6638	1.6638
10.5.2	1.4519	1.4519	1.4519	1.4519	0.00	0.000	0.000	5793572	26450623	1.5151	1.5151
10.5.3	1.4736	1.4736	1.4736	1.4736	0.00	0.000	0.000	5808815	27265176	1.8619	1.8619
10.5.4	1.2849	1.2849	1.2849	1.2849	0.00	0.000	0.000	4508835	25627890	1.8450	1.8032
10.10.1	2.3265	2.3265	2.3347	2.3265	0.35	0.011	0.000	5728418	28019615	3.3204	2.6737
10.10.2	3.1586	3.1586	3.1586	3.1586	0.00	0.000	0.000	5924160	26639115	3.7888	3.5087
10.10.3	2.5527	2.5527	2.5527	2.5527	0.00	0.000	0.000	3357279	29000966	3.5457	3.5457

10.10.4	2.5393	2.5393	2.5393	2.5393	0.00	0.000	0.000	2924422	26113164	3.2579	2.8711
10.20.1	4.4524	4.4524	4.4524	4.4524	0.00	0.000	0.000	5826215	26201747	4.5351	4.5351
10.20.2	6.1678	6.1678	6.1678	6.1678	0.00	0.000	0.000	7379404	26938008	7.0386	6.7845
10.20.3	4.5463	4.5463	4.5463	4.5463	0.00	0.000	0.000	7190908	27736599	5.8839	5.1654
10.20.4	6.1536	6.1536	6.1601	6.1536	0.11	0.009	0.000	7469917	28239850	6.8651	6.7417
12.5.1	1.3738	1.3738	1.3738	1.3738	0.00	0.000	0.000	5606612	22287186	1.5301	1.5301
12.5.2	1.0690	1.0590	1.0798	1.0590	1.97	0.020	0.000	5852701	22431403	1.8254	1.7826
12.5.3	1.4477	1.4477	1.4477	1.4477	0.00	0.000	0.000	5600286	23479327	1.6556	1.6293
12.5.4	1.5810	1.5810	1.5810	1.5810	0.00	0.000	0.000	5605303	23118928	1.8868	1.7541
12.10.1	2.6810	2.6810	2.6810	2.6810	0.00	0.000	0.000	6814475	23554627	3.7695	3.7708
12.10.2	2.6842	2.6842	2.6842	2.6842	0.00	0.000	0.000	5083628	21947382	3.3080	3.3080
12.10.3	2.8805	2.8805	2.8805	2.8805	0.00	0.000	0.000	4891834	21958882	3.7680	3.6837
12.10.4	2.3142	2.3142	2.3142	2.3142	0.00	0.000	0.000	5546401	22823268	4.3579	3.1713
12.20.1	5.7776	5.7776	5.7986	5.7776	0.36	0.022	0.000	6792508	22746779	7.7530	7.0187
12.20.2	8.2725	8.2725	8.2725	8.2725	0.00	0.000	0.000	3311492	20178441	8.9144	8.2733
12.20.3	4.1669	4.1669	4.1669	4.1669	0.00	0.000	0.000	4335987	21708160	4.3857	5.4396
12.20.4	6.4099	6.0886	6.4099	6.0886	5.28	0.000	0.000	5231333	24023052	7.7145	7.7145
20.5.1	1.7935	1.7935	1.7935	1.7935	0.00	0.000	0.000	1704737	11301575	2.1394	2.0346
20.5.2	1.9540	1.9540	1.9540	1.9540	0.00	0.000	0.000	1667146	10631574	2.2816	2.0477
20.5.3	1.4866	1.4866	1.4866	1.4866	0.00	0.000	0.000	1946247	11114470	1.9227	1.9191
20.5.4	1.3789	1.3789	1.3789	1.3789	0.00	0.000	0.000	2171819	12616178	1.8572	1.8444
20.10.1	3.2525	3.2525	3.4044	3.2525	4.67	0.131	0.000	1781196	13287214	4.4137	3.9230
20.10.2	3.0894	3.0894	3.0894	3.0894	0.00	0.000	0.000	2041085	12565887	4.6780	4.7589
20.10.3	3.7023	3.7023	3.7023	3.7258	-0.63	0.000	0.050	1996986	11718401	4.6208	4.6310
20.10.4	3.1966	3.3089	3.1966	3.3137	-3.53	0.000	0.015	2153710	12508345	4.6358	4.3647
20.20.1	7.3445	7.3445	7.3445	7.3512	-0.09	0.000	0.021	862654	11638755	8.5470	8.0480
20.20.2	7.9220	7.5489	7.9291	7.5489	5.04	0.006	0.000	1780965	11942478	8.8392	8.5893
20.20.3	7.5968	7.4610	7.5968	7.4746	1.63	0.000	0.043	2011912	10418184	10.0839	8.5274
20.20.4	7.0133	7.0133	7.0133	7.0133	0.00	0.000	0.000	1609199	11091582	10.3536	8.6699
50.10.1	5.8613	5.8613	5.8615	5.8613	0.00	0.001	0.000	82945	1110439	7.3846	6.2551
50.10.2	5.5849	5.5849	5.5855	5.6210	-0.63	0.001	0.076	105767	1351994	6.7108	6.4094
50.10.3	5.5939	5.4224	5.6085	5.4255	3.37	0.015	0.001	121030	1453177	6.9004	7.1079
50.10.4	5.1411	5.2083	5.3586	5.3526	0.11	0.113	0.109	103371	1762387	6.6976	6.8049
50.20.1	10.4390	10.4553	10.4567	10.4564	0.00	0.009	0.001	99472	1313010	12.6861	13.2090
50.20.2	10.0561	10.0561	10.0601	10.0561	0.04	0.006	0.000	109380	1396076	12.2855	12.6846
50.20.3	10.5018	10.5425	10.5282	10.6570	-1.21	0.021	0.060	108180	1337178	14.0583	14.3447
50.20.4	10.6642	10.6642	10.8033	11.0008	-1.80	0.218	0.187	94844	1299549	12.7175	12.7835
50.30.1	15.7714	15.8179	16.1633	15.8179	2.18	0.380	0.000	99198	1509428	20.1345	19.8709
50.30.2	15.3549	15.0148	15.3588	15.4636	-0.68	0.007	0.473	105189	1427745	18.7605	20.0485
50.30.3	16.3874	16.7690	16.3928	16.7713	-2.26	0.002	0.003	115615	1340989	20.4367	21.1009
50.30.4	18.8994	18.2875	18.9022	18.2875	3.36	0.005	0.000	81872	1138204	22.2569	22.0994
50.40.1	20.1210	20.3751	20.1367	21.1771	-4.91	0.025	0.551	86087	1230243	24.8783	25.1103
50.40.2	20.6253	20.6262	20.6253	20.6262	0.00	0.000	0.000	91673	1277543	22.7771	23.1038
50.40.3	23.1217	22.6452	23.1391	22.7053	1.91	0.026	0.190	80220	1132225	29.4871	27.0804
50.40.4	22.6857	22.3371	22.6990	22.7891	-0.40	0.017	0.195	78042	1222262	25.4232	28.0744
100.10.1	6.9951	6.8574	7.0907	6.8902	2.91	0.070	0.027	9717	202026	8.6596	8.9208
100.10.2	7.7357	7.5851	7.8463	7.6781	2.19	0.086	0.081	8552	165091	9.4893	9.2321
100.10.3	7.3089	7.1835	7.4950	7.3055	2.59	0.079	0.092	9183	184846	9.4422	8.8057
100.10.4	7.4428	7.4568	7.4689	7.5459	-1.02	0.021	0.064	7178	165521	8.4320	8.9807
100.20.1	13.9022	13.6067	14.0588	13.7946	1.92	0.203	0.114	4822	123349	17.0567	16.3388
100.20.2	14.4535	14.1340	14.5394	14.5375	0.01	0.047	0.145	4784	144852	18.3780	17.1472
100.20.3	14.3902	13.7099	14.5326	13.7672	5.56	0.072	0.065	4801	169247	17.3086	17.4572
100.20.4	14.4173	13.8494	14.5326	14.1976	2.36	0.058	0.245	24536	158025	19.2880	18.5038
100.30.1	22.6443	22.5882	22.6922	23.6364	-3.99	0.030	0.546	43055	288148	27.1382	28.1576
100.30.2	22.5750	22.3143	22.6717	22.3846	1.28	0.102	0.102	41149	373621	25.9523	26.1837
100.30.3	23.7731	23.7195	24.0282	23.9094	0.50	0.102	0.114	21365	327256	28.5819	28.9919
100.30.4	23.3552	22.3701	23.5186	22.6585	3.80	0.130	0.149	30540	436002	27.5232	26.6007
100.40.1	29.1029	29.1397	29.4098	30.1807	-2.55	0.379	1.109	37298	517089	42.5995	37.9838
100.40.2	29.6915	30.9900	31.7017	31.2092	1.58	0.724	0.177	33154	413419	39.6568	39.4368
100.40.3	28.6277	29.0248	30.1031	29.6653	1.48	0.688	0.309	27243	448001	37.4194	37.5153
100.40.4	29.9389	28.9735	30.0684	29.2049	2.96	0.068	0.160	32030	420258	37.1351	37.1676
150.10.1	10.0642	8.7903	10.2281	8.9351	14.47	0.151	0.057	732	57857	11.5928	11.5984
150.10.2	8.8374	8.2591	9.0005	8.4160	6.94	0.110	0.113	905	53929	10.0959	10.7638
150.10.3	10.8713	8.4960	11.3192	9.0207	25.48	0.241	0.215	281	47887	11.6823	12.0059
150.10.4	9.7879	8.8373	10.0067	9.0398	10.70	0.113	0.129	760	47692	11.1653	11.2467

150.20.1	18.2182	17.3194	18.3254	17.5964	4.14	0.068	0.372	6310	65612	22.0390	21.4186
150.20.2	18.1021	16.6341	18.3218	17.4507	4.99	0.095	0.610	5893	134443	21.7041	23.7137
150.20.3	18.2992	17.4058	18.4847	18.3447	0.76	0.162	0.512	9188	108024	22.3543	23.4890
150.20.4	17.7624	16.8752	18.0900	17.4774	3.51	0.177	0.389	9065	137596	22.8426	23.6055
150.30.1	26.4783	25.9854	27.5699	26.5488	3.85	0.439	0.333	4460	126169	33.0837	31.6512
150.30.2	27.4671	26.2055	27.7652	26.7411	3.83	0.209	0.258	4667	129856	32.5653	33.7452
150.30.3	26.4816	25.3164	27.0563	26.1137	3.61	0.324	0.456	4614	133104	33.1217	33.0490
150.30.4	27.6283	26.1027	28.0634	27.2923	2.83	0.266	0.900	4621	128121	33.3930	32.4017
150.40.1	39.5856	34.0121	41.1699	35.4534	16.12	1.269	1.059	2107	125867	45.0352	46.7403
150.40.2	38.4930	36.5616	38.8538	38.2965	1.46	0.312	0.682	15880	162347	47.6441	47.7825
150.40.3	39.6415	36.6574	40.0308	38.2955	4.53	0.229	0.896	4441	162242	48.7182	48.1543
150.40.4	38.6243	35.0156	40.3236	36.0662	11.80	0.943	1.054	5016	157272	46.3604	46.5532
200.10.1	10.3993	10.0945	10.6550	10.4050	2.40	0.194	0.163	1970	24243	12.2130	12.5477
200.10.2	10.3971	10.4226	10.5110	10.6149	-0.98	0.082	0.105	1141	24546	11.7903	13.5972
200.10.3	10.6835	9.7990	10.8623	9.9235	9.46	0.142	0.057	1496	75474	13.0375	12.8025
200.10.4	11.2794	10.3553	11.5699	10.6400	8.74	0.242	0.202	474	41039	13.0216	13.2637
200.20.1	22.1915	21.2151	22.9432	21.4601	6.91	0.322	0.259	910	55098	26.1477	26.5177
200.20.2	22.3884	21.4585	22.6663	22.0461	2.81	0.261	0.627	1673	46692	27.0371	26.9291
200.20.3	21.1136	20.8522	21.5369	21.0604	2.26	0.297	0.131	1514	49846	24.6286	24.9398
200.20.4	20.8433	19.2350	22.2340	20.1804	10.18	0.842	0.436	534	52022	24.7949	25.6556
200.30.1	32.5144	30.3602	32.9217	31.7826	3.58	0.380	0.764	1534	82206	37.7211	37.8968
200.30.2	33.1647	32.8128	33.4313	33.2164	0.65	0.260	0.307	4098	33975	36.6478	38.3242
200.30.3	33.4331	32.2535	33.9373	32.7373	3.67	0.413	0.358	1940	29877	39.5920	40.0349
200.30.4	33.7482	32.0931	34.0936	32.7638	4.06	0.228	0.450	3406	59025	38.9176	40.5048
200.40.1	43.0172	41.4980	43.9489	42.3048	3.89	0.886	0.556	3766	95321	52.8401	57.0652
200.40.2	45.3591	43.2502	45.9687	44.2211	3.95	0.498	0.476	3523	93067	53.0603	55.1417
200.40.3	45.3140	43.3375	46.3483	44.2613	4.72	0.891	0.642	1982	100623	53.4029	54.0758
200.40.4	45.6621	42.0579	47.5789	43.3370	9.79	1.195	0.850	1831	98119	52.4187	53.3860

Note. the left column shows the values given by our implementation, the right column those found in Sacramento et al. (2019), z^* represents the lowest found objective value, μ represents the mean objective value, Difference shows the difference between the mean objective values in (%), σ shows the standard deviation, Iterations the average amount of iterations and $z_{initial}$ the initial objective value.

More interesting are the mean objective values of both implementations. Again, we see zero to few differences with small instances and the largest difference in the implementations is found in instance 12.20.4, where our implementation has 5.28% higher mean objective value. For the larger instances, we see a gradual increase in the difference in objective value as the amount of customers rises. The largest difference we find for instance 150.10.3, where our implementation has 25.48% higher objective value. However, we also find fifteen mean objective values lower than Sacramento et al. (2019). For instance 50.40.1, for example, we find a mean objective value that is 4.91% lower.

The standard deviations of both implementations, although not similar for the instances, show similar values. The standard deviations logically increase with grid size and the number of customers. For both implementations, all standard deviations are smaller than one, except for instance 200.40.4 and 150.40.4 of respectively our and Sacramento et al.'s (2019) implementation. This indicates that for different random values the ALNS algorithm can find consistently similar values.

Our implementation does have a considerably lower amount of iterations, which gets progressively worse for larger instances. For smaller instances, Sacramento et al. (2019)'s implementation has around four times more iterations, while for larger instances the difference can be more than 10.000 times, such as for instance 150.10.3. Interestingly though, the amount of iterations for a given number of customers seems to decrease

with grid size for our implementation. This might be because the number of feasible sorties decreases with an increase in grid size since fewer sorties still satisfy the endurance constraint (1).

Lastly, we compare the initial solutions of both implementations. Although our implementation uses fewer methods to improve the initial solution, the values are quite comparable. We find both better and worse initial solutions.

While it might seem counter-intuitive that our implementation finds better mean objective values with fewer iterations, this can be explained by the difference in formulation. As explained earlier, we relaxed the assumption that trucks need to arrive before drones at the recovery position (8). As a result, there are more feasible sorties, which can explain the better objective values. This relaxation can also explain the paradoxical result that our implementation finds better initial solutions with fewer improvement methods.

It is, however, hard to say how large the effect of relaxing this assumption is on the mean objective values since there is an opposing effect that increases the objective values. Our optimal objective values should be relatively worse because we have fewer iterations for every instance, which decreases the chance of improving the solution. This also explains why many of our mean objective values are larger for larger instances. While for the smaller instances, where we can perform more iterations and need relatively fewer iterations to get good results, we find similar mean objective values. Additionally, we have not implemented the heavy insertion repair method, which according to Sacramento et al. (2019) is quite successful in improving the current solution. As a consequence, we have probably found worse objective values.

Still, it seems that the effect of relaxing assumption (8) is rather small since we find a similar result for most of the smaller instances, where we have sufficient iterations. The explanation why relaxing this assumption has quite a small impact is intuitive. It is, namely, in most cases beneficial for the truck driver to as quickly retrieve the drone as possible since it can then launch the drone again, which saves costs. As a result, the truck driver already arrives before the drone at the recovery location in most cases.

We identified two reasons why our implementation has a significantly lower amount of iterations. First of all, our implementation is likely to have been less efficiently programmed and, as a result, the ALNS algorithm can perform fewer iterations in the same time frame. Secondly, we believe the ALNS algorithm's running time depends strongly on the amount of feasible sorties, since the ALNS algorithm has to perform a lot of feasibility checks and calculate the new total costs for every feasible sortie. Due to the relaxation of Assumption (8), the set of feasible sorties has increased and therefore, the running time of the ALNS algorithm has increased.

5.3 Effect of the failure probability on profitability of drones

In this subsection we will discuss and assess the effect of the failure probability on the profitability of drones. We will start by describing the test instances. Then, we will provide our results of the ALNS algorithm for the VRPSDD. Lastly, we will discuss the performance of our solutions when the failure probability used in

the model does not equal the true failure probability.

5.3.1 Test instances

Actual data on the prevalence of autonomous drone accidents is still lacking since autonomous drones are not widely used yet. Additionally, The probability $\bar{p} = 1 - p$ of a drone accident is likely to change with time, as more advanced technology will reduce the likelihood of an accident. On the other hand, drone traffic will likely increase, which would also increase the average probability of drone accidents due to collisions. For this reason, the odds of having a drone accident will be set to a range of values from one in a thousand to one in a million, and zero, ($\bar{p} \in \{0.001, 0.0001, 0.00001, 0.000001, 0\}$). For airplanes the rate of accidents is approximately one in 1.2 million (Ropeik & Gray, 2002) and we expect drones to have a similar or larger failure rate.

The smaller instances do not seem realistic in practice, due to the fact that very few customers are served and mainly one truck is used. Therefore, we will only consider the instances with 150 or 200 customers. Since our implementation has few iterations for these instances, we increase the maximum running time of the algorithm to one hour. We will use only one seed for every instance, since the difference in objective values for different seeds was quite small, if enough iterations were performed. To be able to compare the values between different grid sizes, we take the mean of all four different instances with the same grid size and number of customers.

5.3.2 Results

In Table 4 we find the average results of the VRPSDD. First of all, we see that the number of drones used increases as $\bar{p}$ decreases, as we would expect. Moreover, we find that it is not profitable to use drones at all for $\bar{p} = 0.001$. For a rate of failure of one in hundred thousand, i.e., $\bar{p} = 0.0001$, we find that it is optimal to use only very few drones. We then find a sharp increase in the number of drone sorties used for $\bar{p} = 0.00001$. The critical value of the failure probability for the profitability of using drones seems to be between one in ten thousand and one in hundred thousand

Due to the fact that drones are not used for values of $\bar{p} = 0.001$ the average service level, which is equal to one minus the average rate of failure, does not come near the average service requirement limit of 0.999. As a result, this constraint is never restrictive. The same holds for the individual service requirement of 0.99. The maximum amount of drone sorties due to this requirement for respectively $\bar{p} = 0.001, 0.0001$ are approximately ten and a hundred. As we can see, the number of actual drone sorties used is not near this amount. The reason why both constraints are not restrictive is because of the high expected sortie costs. The drone sorties become unprofitable for larger failure probabilities, where the service requirements might become restrictive, since the expected sortie costs increase because there is a higher probability to incur the relatively large failure costs. For larger success probabilities usage of drones becomes profitable but then the service requirements will not be restrictive anymore. The service constraints are, thus, already satisfied by minimizing the costs.

Table 4: Results of the VRPSDD

	$\bar{p}$	150.1	150.2	150.3	150.4	200.1	200.2	200.3	200.4
	0.001	12.2834	24.4742	35.9306	50.7879	14.1891	28.1889	42.8582	57.9038
	0.0001	12.2543	24.0621	34.1998	47.1133	14.137	28.015	41.6127	55.3359
$\bar{z}$	0.00001	10.4480	19.2245	28.3749	39.6452	11.8588	22.6375	34.4411	45.4581
	0.000001	9.5981	18.2724	27.2779	38.4296	10.6428	21.5161	33.0462	44.4336
	0	9.6973	18.1164	27.0979	38.9560	10.3541	21.3242	33.0088	44.7081
	0.001	0	0	0	0	0	0	0	0
	0.0001	0.25	8.25	16.5	25.5	0.75	4.25	11.75	29.5
#D	0.00001	45.25	57.75	57.75	55.5	71.75	75	77	83
	0.000001	54	64	63.25	60.25	82.25	82.75	84.5	82
	0	52.5	65	63	58.5	86.25	83	84	81.25
$\lvert C^D \rvert$	N.A.	128	130.25	127.5	128.75	174.75	173.5	172.5	173.25
	0.001	1	2	2	2.5	2	2	3	3
	0.0001	1	2	2	2.5	2	2	3	3
#T	0.00001	1	2	2	2.75	2	2	3	3
	0.000001	1	2	2	2.5	2	2	3	3
	0	1	2	2	2.75	2	2	3	3
	0.001	0	0	0	0	0	0	0	0
	0.0001	1.67E-07	1.55E-05	6.35E-05	0.000109	0	3.63E-06	1.77E-05	9.05E-05
ARF	0.00001	7.03E-05	5.94E-05	5.86E-05	4.49E-05	6.94E-05	7.26E-05	5.41E-05	6.01E-05
	0.000001	9.95E-06	7.39E-06	7.00E-06	5.44E-06	9.53E-06	8.78E-06	6.54E-06	5.85E-06
	0	0	0	0	0	0	0	0	0
	0.001	341.75	1535.5	1853.75	3634	218	375	882	781
	0.0001	480.75	5187.25	8285	20746	281	675	3415	4073
#Iterations	0.00001	5016	51421	42124	77658.5	11492	12640	29134	26985
	0.000001	5957	51294.5	38165	76302.75	17767	17228	38176	30476
	0	4436.75	50551.75	55682	86187.25	20413	16897	31728	25965

Note. $\bar{z}$ is the mean objective value in €, $\#D$ represents the amount of drone sorties used in the solution, $|C^D|$ is the total amount of customers with demand smaller than Q^D, $\#T$ the amounts of trucks, ARF is the average rate of failure, i.e., the average chance for a customer to not receive a package and #Iterations the average amount of iterations .

Interestingly, we still find a very low amount of iterations for the ALNS algorithm with a higher probability of failure. For example, instance 200.10 with $p = 0.001$ has only 296 iterations on average. While instances with a smaller failure probability can have around thirty thousand iterations. This is likely due to the fact that our algorithm's running times are mainly determined by the set of feasible sorties. In these cases, drones are not used and therefore, the set of feasible sorties is very large since new sorties can not coincide anymore with sorties already present in the route because there are none. For every possible drone sortie the total costs of the route need to be determined and, as a result, the running time of an iteration increases. This is not an unsurprising result since the problem reduces to a capacitated VRP if drone sorties are unprofitable, for which this ALNS algorithm was not designed to be efficient.

If we compare the number of drone sorties for different grid sizes, we find that for larger grid sizes drones are more profitable. This might seem counter-intuitive since the sortie costs SC increase for larger distances because of the second term in the fixed costs F_s for sortie s, see Equation (3), where we assumed that in the future a truck has to travel from the depot to the customer to deliver the lost package. However, the cost savings also increase for larger distances, as drones are cheaper per mile. Additionally, the fixed repair costs F account for the largest part of the SC and these are constant at 2000 euros. Therefore, large distances

decrease the repair costs per mile and make drones more profitable.

Also from a time-saving perspective can drones have a larger impact on larger grids as then the time saved per mile increases, since the launch and recovery time are constant. These time savings can prevent another truck is needed to serve the leftover customers, which can save costs. For these instances, however, this does not seem to be relevant as the number of trucks for different p's does not change. It is, however, important to note that we only have four samples for each grid and therefore, we are not able to give conclusive answers on the effect of grid size, as these might be caused by random differences.

5.3.3 Solution performance under uncertainty

In practice it might be hard to correctly estimate the true value of the failure probability $\bar{p}$. Additionally, it is interesting to see if the lower objective values discussed previously, are due to better-found solutions by the algorithm or because of the lower failure probability $\bar{p}$. Therefore, we will now consider the objective values of our solutions when the true value of $\bar{p}$ does not equal the value of the estimated parameter used in the model $\tilde{\bar{p}}$.

Note that if the ALNS algorithm was able to find better solutions for different $\tilde{\bar{p}}$, then for each column in Table 5 the value on the diagonal, when $\tilde{\bar{p}} = \bar{p}$, should be the minimum of that column. We find that the objective values for $\bar{p}$, $\tilde{\bar{p}} \in (0.00001, 0.000001, 0)$ are quite similar and that for some cases the diagonal value does not represent the minimum of those columns. This is, for example, the case for instance 200.10, where both $\tilde{\bar{p}} = 0.00001$ and $\tilde{\bar{p}} = 0$ are able to find on average a better solution for $\bar{p} = 0.000001$ then $\tilde{\bar{p}} = 0.000001$. This means that the ALNS algorithm does not necessarily find better solutions for smaller failure probabilities, i.e., $\bar{p} \leq 0.00001$

The reason why the algorithm does not always find the minimum objective value for $\tilde{\bar{p}} = \bar{p} \leq 0.00001$ is hard to pinpoint. The most logical explanation is that for $\tilde{\bar{p}} \leq 0.00001$ the expected costs of failure become smaller than the cost savings of using a sortie and therefore, for $\tilde{\bar{p}} \leq 0.00001$ the objective values for the same true $\bar{p}$ are not notably different. Due to randomness then, a solution with $\tilde{\bar{p}} \neq \bar{p}$ can possibly find a better objective value.

Another explanation could be the amount of iterations. However, for the instances, where $\tilde{\bar{p}} = 0.000001$ has the most iterations (200.20, 200.30, 200.40, see Table 5.3.2), we do not the find the best solutions for $\tilde{\bar{p}} = 0.000001$. Thus, in this case the amount of iterations is not the determining factor for finding better solutions.

Comparing the objective values for $\bar{p} \geq 0.00001$ we do find the minimum objective value when $\tilde{\bar{p}} = \bar{p}$. So for a failure probability $\bar{p} \in (0.001, 0.0001, 0.00001)$ the algorithm does find better solutions when the model value equals the actual value and therefore, we do find that the profitability of drone sorties does substantially differ for those values. Also note that for all instances the objective value of $\tilde{\bar{p}} = 0.001$ is the same for all $\bar{p}$, because

Table 5: Solution performance for $\tilde{\bar{p}} \neq \bar{p}$

(a) 150.10

$\tilde{\bar{p}}$ \ $\bar{p}$	0.001	0.0001	0.00001	0.000001	0
0.001	12.283	12.283	12.283	12.283	12.283
0.0001	12.705	12.254	12.209	12.205	12.204
0.00001	98.259	18.592	10.448	9.632	9.541
0.000001	114.915	20.287	10.572	9.598	9.490
0	112.268	20.194	10.749	9.803	9.697

(b) 150.20

$\tilde{\bar{p}}$ \ $\bar{p}$	0.001	0.0001	0.00001	0.000001	0
0.001	24.474	24.474	24.474	24.474	24.474
0.0001	38.906	24.062	22.575	22.426	22.410
0.00001	132.304	29.638	19.225	18.182	18.066
0.000001	144.508	30.967	19.428	18.272	18.144
0	146.456	31.141	19.421	18.247	18.116

(c) 150.30

$\tilde{\bar{p}}$ \ $\bar{p}$	0.001	0.0001	0.00001	0.000001	0
0.001	35.931	35.931	35.931	35.931	35.931
0.0001	63.822	34.200	31.224	30.926	30.893
0.00001	141.596	38.800	28.375	27.331	27.215
0.000001	152.273	39.839	28.421	27.278	27.151
0	151.717	39.734	28.363	27.224	27.098

(d) 150.40

$\tilde{\bar{p}}$ \ $\bar{p}$	0.001	0.0001	0.00001	0.000001	0
0.001	50.788	50.788	50.788	50.788	50.788
0.0001	92.910	47.113	42.509	42.048	41.997
0.00001	148.875	49.675	39.645	38.641	38.530
0.000001	157.932	50.406	39.520	38.430	38.308
0	155.130	50.701	40.132	39.074	38.956

(e) 200.10

$\tilde{\bar{p}}$ \ $\bar{p}$	0.001	0.0001	0.00001	0.000001	0
0.001	14.189	14.189	14.189	14.189	14.189
0.0001	14.137	14.137	14.137	14.137	14.137
0.00001	147.106	24.365	11.859	10.606	10.467
0.000001	171.715	26.925	12.126	10.643	10.478
0	178.885	27.551	12.077	10.526	10.354

(f) 200.20

$\tilde{\bar{p}}$ \ $\bar{p}$	0.001	0.0001	0.00001	0.000001	0
0.001	28.189	28.189	28.189	28.189	28.189
0.0001	35.669	28.015	27.249	27.172	27.164
0.00001	168.920	36.157	22.638	21.283	21.133
0.000001	184.137	37.926	23.011	21.516	21.350
0	184.604	37.952	22.990	21.491	21.324

(g) 200.30

$\tilde{\bar{p}}$ \ $\bar{p}$	0.001	0.0001	0.00001	0.000001	0
0.001	42.858	42.858	42.858	42.858	42.858
0.0001	62.758	41.613	39.494	39.282	39.258
0.00001	185.505	48.336	34.441	33.050	32.895
0.000001	200.189	49.826	34.574	33.046	32.876
0	199.337	49.855	34.696	33.178	33.009

(h) 200.40

$\tilde{\bar{p}}$ \ $\bar{p}$	0.001	0.0001	0.00001	0.000001	0
0.001	57.904	57.904	57.904	57.904	57.904
0.0001	108.343	55.336	50.008	49.475	49.416
0.00001	208.439	60.455	45.458	43.956	43.790
0.000001	206.982	60.735	45.917	44.434	44.269
0	205.949	61.023	46.342	44.871	44.708

Note. $\tilde{\bar{p}}$ represents the failure probability used in the model, $\bar{p}$ represents the true failure probability, values shown are the average costs of a solution for the true failure probability $\bar{p}$

drones are unprofitable for $\tilde{\bar{p}} = 0.001$ and, as a result, the objective value does not depend on $\bar{p}$.

5.4 Recourse policy

We will now discuss the performance of the advanced recourse policy relative to the simple recourse policy, both described in Section 4.5. Since for $\bar{p} = 0.001$ no drones are used and for $\bar{p} = 0$ there are no failed deliveries, our recourse policy is only relevant for $\bar{p} \in (0.0001, 0.00001, 0.000001)$. The expected savings and expected savings given that a failure has occurred of the advanced recourse policy can be found below in Table 6.

First of all, we find that for most instances the recourse policy has the largest expected savings when $\bar{p} = 0.00001$. Only for the largest grid, $g = 40$, the advanced recourse policy has the largest expected savings for $\bar{p} = 0.0001$. On the other hand, given that a failure has occurred the advanced recourse policy is the most

effective for a failure probability equal to 0.000001.

Table 6: Performance of the advanced recourse policy

(a) 150.10

$\bar{p}$	$\mathbb{E}[S]$	$\mathbb{E}[C]$	$\mathbb{E}[S\|F]$	$\mathbb{E}[RC\|F]$
0.0001	3.37E-08	12.254	0.001	12.128
0.00001	9.14E-04	10.448	2.007	13.174
0.000001	1.15E-04	9.598	2.133	13.219

(b) 150.20

$\bar{p}$	$\mathbb{E}[S]$	$\mathbb{E}[C]$	$\mathbb{E}[S\|F]$	$\mathbb{E}[RC\|F]$
0.0001	1.16E-04	28.015	0.118	22.970
0.00001	1.64E-03	22.638	2.800	22.801
0.000001	1.81E-04	21.516	2.805	22.895

(c) 150.30

$\bar{p}$	$\mathbb{E}[S]$	$\mathbb{E}[C]$	$\mathbb{E}[S\|F]$	$\mathbb{E}[RC\|F]$
0.0001	9.63E-04	34.200	0.573	32.929
0.00001	1.90E-03	28.375	3.277	33.029
0.000001	2.28E-04	27.278	3.600	33.285

(d) 150.40

$\bar{p}$	$\mathbb{E}[S]$	$\mathbb{E}[C]$	$\mathbb{E}[S\|F]$	$\mathbb{E}[RC\|F]$
0.0001	2.67E-03	47.113	0.964	44.992
0.00001	1.81E-03	39.645	3.238	44.818
0.000001	1.93E-04	38.430	3.171	44.697

(e) 200.10

$\bar{p}$	$\mathbb{E}[S]$	$\mathbb{E}[C]$	$\mathbb{E}[S\|F]$	$\mathbb{E}[RC\|F]$
0.0001	0	14.137	N.A.	N.A.
0.00001	9.26E-04	11.859	1.335	12.983
0.000001	1.55E-04	10.643	1.869	13.517

(f) 200.20

$\bar{p}$	$\mathbb{E}[S]$	$\mathbb{E}[C]$	$\mathbb{E}[S\|F]$	$\mathbb{E}[RC\|F]$
0.0001	8.67E-06	28.015	0.016	27.289
0.00001	2.14E-03	22.638	2.854	26.023
0.000001	2.80E-04	21.516	3.403	26.670

(g) 200.30

$\bar{p}$	$\mathbb{E}[S]$	$\mathbb{E}[C]$	$\mathbb{E}[S\|F]$	$\mathbb{E}[RC\|F]$
0.0001	1.92E-04	41.613	0.158	39.953
0.00001	2.16E-03	34.441	2.801	37.796
0.000001	3.23E-04	33.046	3.802	38.680

(h) 200.40

$\bar{p}$	$\mathbb{E}[S]$	$\mathbb{E}[C]$	$\mathbb{E}[S\|F]$	$\mathbb{E}[RC\|F]$
0.0001	4.77E-03	55.336	1.634	52.646
0.00001	3.22E-03	45.458	3.884	50.278
0.000001	3.33E-04	44.434	4.042	50.892

Note. $\mathbb{E}[S]$ represent the expected savings of the advanced recourse policy, $\mathbb{E}[C]$ the expected costs of the simple recourse policy, $\mathbb{E}[S|F]$ the expected savings given that a failure has occurred of the advanced recourse policy and $\mathbb{E}[RC|F]$ the expected route costs of the simple recourse policy given that a failure has occurred

This is of course directly related to the number of drones used in those solutions. For a smaller failure probability, more drones are used and, as a result, there are more route adjustment costs in the simple recourse policy. The advanced recourse policy minimizes these route adjustment costs by reoptimizing the route. Therefore, given that a failure has occurred the largest cost savings are obtained for smaller failure probabilities. However, the expected cost savings for smaller $\bar{p}$ decline at some point, since the probability of having a failure decreases, while the number of drones used remains about the same. This is what we find for $\bar{p} = 0.00001$ and already for $\bar{p} = 0.0001$ for the largest grid.

Logically we find that for larger grids the cost savings increase. However, the cost savings seem to increase less than proportionate to the expected route costs given a failure. For smaller grids, these expected cost savings given a failure are, for example, around ten percent or more relative to the expected route costs given a failure has occurred for $\bar{p} = 0.00001$. While for the largest grid, $g = 40$, these same cost savings are only around 7.5% of the expected route costs given failure. This is interesting as we do see that the expected route costs given a failure are proportionate to grid size. Therefore, it seems that the advanced recourse policy is relatively less effective for larger grids.

In general we find that the expected savings of the advanced recourse policy are very low. They are in the order of $10^{-3}, 10^{-4}$ euro for $p = 0.00001, 0.000001$ respectively. However, for large-scale logistics companies with frequent deliveries, this recourse policy might still save significant costs. Moreover, the algorithm of

the advanced recourse policy runs within a second and is relatively easy to implement, making it a practical method to reduce costs.

6 Conclusion

In this paper we assessed the effect of the probability of successful drone delivery on the profitability of drones. First, we assessed the performance of our ALNS algorithm in comparison to Sacramento et al.'s (2019) implementation. We found that our implementation performed similarly for small instances, despite having fewer repair methods and fewer initial improvement methods. For most larger instances, on the other hand, our implementation relatively underperformed. This is likely due to the fact that we have a significantly lower amount of iterations for larger instances and, as a consequence, our implementation finds worse solutions in the same running time. The difference in running times is likely to be the result of a less efficient implementation of the algorithm.

Still, we are also able to find a few better solutions, which are due to the fact that we have relaxed the truck endurance assumption, which assumes that the total travel and service time of a truck between the launch and recovery location of a drone has to be smaller than the endurance of a drone. We assumed that drones could land at the recovery position and can wait at the truck there. We conclude, however, that letting drones land at the recovery position does not save a lot of costs, since it is more profitable for trucks to quickly recover drones so that the drone can visit new customers.

Secondly, we assessed the effect of drone failure probability $\bar{p}$ on the profitability of drones. We found that drones are only profitable for a $\bar{p} \leq 0.0001$. So drone failures should occur with a rate of less than one in ten thousand visits to be profitable. Otherwise, the expected value of incurring repair and adjustment costs is too high for drones to be profitable.

Additionally, we found that drones become more profitable if customers are spread between larger distances. Since the probability of failure is constant over distance, the expected fixed repair costs, that contribute the most to the costs of a drone, remain constant over distance, while the expected cost savings of drones increase per mile. This makes drones more profitable for routes that need to cover larger distances.

We also noticed that the ALNS algorithm is very inefficient for probabilities of failure $\bar{p} \geq 0.001$ since in that case drones are unprofitable and the problem reduces to a capacitated VRP. The ALNS algorithm was not designed for this problem and therefore, other optimizing methods would be more appropriate.

Next, we checked how our found solutions performed when the failure probability used by the ALNS algorithm does not equal the actual failure probability. We found that for the model's failure probability $\tilde{\bar{p}} \in (0.00001, 0.000001, 0)$ there were no substantial differences in costs when they had the same actual failure probability. Therefore, we can conclude that the ALNS algorithm is not necessarily able to find better solutions for those probabilities. This is likely caused by the fact that for those probabilities the expected cost savings

are always larger than the expected failure costs of a drone sortie, which means that drone visits have the same relative profitability compared to truck visits for these probabilities. As a result, the ALNS algorithm does not find solutions with significantly different objective values for the same actual failure probability.

For $\tilde{p} \in (0.001, 0.0001, 0.00001)$ we did find large cost differences when the solutions had the same actual failure probability. This is logical since there were also large differences in drone profitability for those failure probabilities. As a result, the ALNS algorithm did find solutions with considerably different costs for those instances.

Lastly, we discussed the performance of our advanced recourse policy relative to the simple recourse policy. We found that the expected savings of the policy were in the order of 10^{-3} euro for a failure probability equal to one in hundred thousand. These small expected savings were the result of a small failure probability. The expected savings given a failure, however, could account for approximately ten percent of the expected route costs given a failure. Which showed that the advanced recourse policy is effective in minimizing the travel costs given a failure has occurred.

While the expected savings from the advanced recourse policy seem small, large logistics companies with frequent delivery could still save significant costs by implementing the advanced recourse policy. Especially, since the cheapest insertion algorithm is relatively easy to implement and reoptimizes the route almost instantly.

A limitation of this paper was the sample size of instances used for a certain grid size and number of customers. Only four instances were generated and used, which is too few to give reliable results. Therefore, the results given in this paper are vulnerable to randomness. Additionally, no statistical tests have been performed to test if the differences in the results were statically significant. Therefore, no statistically significant differences can be reported in this paper, which is necessary to give reliable conclusions. Even though, many of the results in this paper contained notable and consistent differences, statistical tests should have been performed.

In this paper, we assumed a constant failure probability for each drone visit, independent of distance or time traveled, because we expect most drone accidents to occur shortly after taking off and shortly before landing. However, it might also be realistic to assume that a drone failure is dependent on the distance traveled. Therefore, for future research it is interesting to test how a probability that does depend on the number of miles traveled affects the profitability of drones. For example, assuming a constant failure rate per mile traveled or even more interestingly a non-homogeneous failure rate to account for a higher failure probability at the start and end of the delivery.

Lastly, it might be interesting to let the failure probability be dependent on the number of drones in the airspace to see what the effect of air traffic is on the profitability of drones. Air traffic is going to increase due to drone delivery, which could have a negative external effect on the profitability of additional drones in the air space.

Since this is the first paper that incorporates stochastic drone delivery into a problem formulation with drones, it might be interesting to also incorporate stochastic drone delivery into formulations other than the VRP-D. For example, the formulation provided by Kitjacharoenchai et al. (2019), considers a fleet of drones flying independently from the trucks. In that formulation, including a failure probability might have less effect as trucks could potentially send out another drone to serve the remainder of drone customers.

Another possibility for future research is to see the effect of different cost functions. In this paper, for example, we set the repair costs to fifty percent of total production costs, which resulted in quite a large amount. As a result, the other failure costs became negligible. Adjusting the amount of repair costs might give interesting new results as the other costs become relevant again.

Additionally, we have not yet used any costs related to the loss of the package. These costs could be included in the fixed costs of a failure, but could also be made dependent on the demand of a customer in kilo. Even more interesting, would be to include a new parameter that models the value of each package demanded by a customer, to see if it is even profitable and efficient for drones to deliver higher-value packages.

References

Alamouri, A., Lampert, A., & Gerke, M. (2021). An exploratory investigation of uas regulations in europe and the impact on effective use and economic potential. *Drones*, *5*(3), 63. doi: https://doi.org/10.3390/drones5030063

Allain, R. (2013). *Physics of the amazon octocopter drone.* Retrieved from `https://www.wired.com/2013/12/physics-of-the-amazon-prime-air-drone/`

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2011). *The traveling salesman problem : a computational study.* Princeton University Press. doi: https://doi.org/10.1515/9781400841103

Bertsimas, D. (1988). *Probabilistic combinatorial optimization problems* (Unpublished doctoral dissertation). Massachusetts Institute of Technology.

Boeing. (2021). *Statistical summary of commercial jet airplane accidents* (Tech. Rep.). Arlington, United States.

Dell'Amico, M., Montemanni, R., & Novellani, S. (2021). Drone-assisted deliveries: New formulations for the flying sidekick traveling salesman problem. *Optimization Letters*, *15*(5), 1617–1648.

DHL. (2016). *Successful trial integration of dhl parcelcopter into logistics chain.* Retrieved from `https://www.dpdhl.com/en/media-relations/press-releases/2016/successful-trial-integration-dhl-parcelcopter-logistics-chain.html`

Gendreau, M., Laporte, G., & Séguin, R. (1996). Stochastic vehicle routing. *European Journal of Operational Research*, *88*(1), 3–12.

Goodchild, A., & Toy, J. (2018). Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing co2 emissions in the delivery service industry. *Transportation Research Part D: Transport and Environment*, *61*, 58–67. doi: https://doi.org/10.1016/j.trd.2017.02.017

Karak, A., & Abdelghany, K. (2019). The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transportation Research Part C: Emerging Technologies*, *102*, 427–449.

Keane, J. (2022). *Drone delivery player manna eyes european launches next year.* Retrieved from `https://www.forbes.com/sites/jonathankeane/2022/04/26/drone-delivery-player-manna-eyes-european-launches-next-year/?sh=3915dc443238`

Keeney, T. (2021). *Drone delivery: How can amazon charge $1 for drone delivery?* Retrieved from `https://ark-invest.com/articles/analyst-research/drone-delivery-amazon/`

Kellermann, R., Biehle, T., & Fischer, L. (2020). Drones for parcel and passenger transportation: A literature review. *Transportation Research Interdisciplinary Perspectives*, *4*, 100088.

Kersley, A. (2021). *The slow collapse of amazon's drone delivery dream.* Retrieved from `https://www.wired.co.uk/article/amazon-drone-delivery-prime-air`

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.

Koetier, J. (2021). *Drone delivery is live today, and it's 90% cheaper than car-based services.* Retrieved from https://www.forbes.com/sites/johnkoetsier/2021/08/18/drone-delivery-is-live-today-and-its-90-cheaper-than-car-based-services/?sh=61b30c164d02

Miranda, V. R., Rezende, A., Rocha, T. L., Azpúrua, H., Pimenta, L. C., & Freitas, G. M. (2022). Autonomous navigation system for a delivery drone. *Journal of Control, Automation and Electrical Systems*, *33*(1), 141–155.

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

Pisinger, D., & Ropke, S. (2019). Large neighborhood search. In *Handbook of metaheuristics* (pp. 99–127). Springer.

Rao, B., Gopi, A. G., & Maione, R. (2016). The societal impact of commercial drones. *Technology in society*, *45*, 83–90.

Ropeik, D., & Gray, G. M. (2002). *Risk: A practical guide for deciding what's really safe and what's dangerous in the world around you.* Houghton Mifflin Harcourt.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, *40*(4), 455–472.

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

Rosenkrantz, D. J., Stearns, R. E., & Lewis, P. M., II. (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, *6*(3), 563–581.

Sacramento, D. (2017). *Heuristics for solving the drone-vehicle routing problem.* Technical University of Denmark.

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. doi: https://doi.org/10.1016/j.trc.2019.02.018

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and practice of constraint programming* (pp. 417–431). Springer.

Smith, K. W. (2015). Drone technology: Benefits, risks, and legal considerations. *Seattle J. Envtl. L.*, *5:1*, 291–302.

Stöcker, C., Bennett, R., Nex, F., Gerke, M., & Zevenbergen, J. (2017). Review of the current state of uav regulations. *Remote sensing*, *9*(5), 459. doi: https://doi.org/10.3390/rs9050459

UPS. (2017). *What are the weight and size limits for shipping using ups.* Retrieved from https://www.ups.com/pr/en/help-center/sri/sze2.page

Welch, A. (2015). *A cost-benefit analysis of amazon prime air* (Unpublished doctoral dissertation). University of Tennessee at Chattanooga.

A Appendix

A.1 Explanation of the programming files

In this section we will explain how to execute the programming files in the folder "Code and instances.zip" to obtain the results in this thesis.

First of all, the folder instances contains the instances necessary to run the code. Make sure to specify the directory of these instances in the main files to run the code. Similarly the folder solutions contains the solutions obtained by our VRPSDD, which are necessary to run the recourse policy and the uncertainty analysis.

Run Main.java to obtain the replications results of the VRP-D. Make sure to specify in the for loops which instances you want to run, i.e., the combination of customers, grid, instance and seed. Also specify a directory to a csv file, where you want to write your results to. Main.java will then run the ALNS algorithm for the VRP-D programmed in the ALNS.java file. ALNS.java contains all methods and algorithms necessary to run the ALNS algorithm. Additionally, it contains the parameters of the VRP-D.

In ExtensionMain.java run the `runP()` method to obtain the results of the VRPSDD. Again make sure to specify which instances you want to run and the directory of the file for the results to be written to. The ExtensionMain.java will then run the ALNS algorithm for the VRPSDD specified in ALNSE.java. ALNSE.java contains all methods and algorithms for the VRPSDD. Additionally it contains the uncertainty analysis method and recourse policy methods. You need to run the `runUncertainty()` method to run the uncertainty analysis and you need to run the `runRP()` method to run the recourse policy.

The Solution.java file models a solution object that can be used for either the VRP-D or VRPSDD. The solution object contains a list of truck routes. Additionally, the file contains methods to change the solution.

The TruckRoute.java file models a truck route object in the VRP-D or VRPSDD and includes methods to change or get information from the truck route.

The Customer.java file models a customer with its location and demand.

Lastly, the Sortie.java file models a drones sortie with its launch, customer and recovery node. Also, it contains methods to alter the drone sortie.