ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Bachelor Thesis in Econometrics and Operations Research

Drone-assisted parcel delivery

The flying sidekick travelling salesman problem with payload-dependent flight endurance and warm starts

Ryan de Reus (457818)

Supervisor: Y.N. Hoogendoorn
Second Assessor: D. Huisman

23 June, 2019

Abstract

Parcel delivery is becoming more and more popular as people buy a lot of products online. Innovative companies are therefore looking into other ways for their last-mile deliveries. In this paper we looked into the flying sidekick travelling salesman problem, where a drone can deliver parcels to customers apart from the truck. To give a more realistic approach, we incorporated the parcel weight into the flight endurance of the drone. This papers provides exact and heuristic approaches to solve the mixed integer linear programming (MILP) formulation of our problem. Furthermore, we will combine these two approaches to come up with a Warm Start Algorithm that gives us better solutions. Overall, we see that the use of drones is a valuable contribution to a delivery process, with an average reduction of 10.3%.

Contents

1 Introduction

In the last decade, the number of online purchases has risen across the entire world (Clement, 2019). International companies are therefore trying to optimise their delivery processes. In 2013, Amazon introduced its Prime Air parcel-delivery drone, which could deliver packages up to 5 lbs (approximately 2.3 kg) within 30 minutes. Although lots of people thought it was a huge publicity stunt (Carlson, 2013), Amazon began testing their Prime Air in the UK in 2016 (Perlow, 2016). Figure 1 shows a picture of such a delivery drone.

Figure 1: Amazon's Prime Air (source: amazon.com)

Another company with a delivery drone is DHL, which started a test phase in 2013 in which their 'Parcelcopter' crossed the Rhine to deliver medicines. One year later, it became the first company that uses commercial drones to deliver packages by transporting medicines from the German mainland to an island in the North Sea (Hern, 2014). Two years after that, in 2016, the company took a more challenging step and delivered in the Alps, a mountainous region with geographical and meteorological challenges. Last year, DHL made a new development by transporting medicines to an island in Lake Victoria (East Africa). This new 'Parcelcopter' flew a distance of 65 km, with a payload up to 4 kg and an airspeed of 130 km/h (Deutsche Post AG, 2019). One of the most remarkable achievements came from New Zealand, where Domino's (a pizza company) managed to offer a drone-delivery service for customers in the town Whangaparaoa (Murhpy, 2016).

Nevertheless, there are still some barriers to overcome. Each drone, also known as an unmanned aerial vehicle (UAV), has a flight endurance which is affected by the flight speed, payload and battery capacity. Since the UAVs are quite small, the battery capacity of this small vehicle is also limited. So companies have to decide which characteristics (speed, payload, size etc.) they want to prioritise (Murray and Chu, 2015). For the sake of safety, the drone might need some redundant systems (sensors, localisation software, etc.), which also have an impact on battery life (Jeong et al., 2019).

If companies are going to use drones for their last-mile deliveries, we need to optimise the entire delivery process. Therefore, this paper will look into solving some of these delivery problems. The main idea of delivery is that a truck departs from the central depot, visits all the customers and returns to the depot. The mathematical formulation of such a problem is known as the Travelling Salesman Problem (TSP). The origins of the TSP are not very clear, but we know that around 1930 it was first considered mathematically (Lawler, 1985). The TSP is one of the most intensively studied problems in optimisation and is very useful as a benchmark in optimisation methods. In this paper, we incorporate UAVs in the delivery process, where they may be launched from the truck, deliver the package to the customer and return to the truck. This problem is known as the Flying Sidekick Travelling Salesman Problem (FSTSP), as introduced by Murray and Chu (2015).

In this paper, we will solve the FSTSP. We will do this in two different ways. First, we use a mathematical formulation for a mixed integer linear programming (MILP) model based on Murray and Chu (2015), which we will solve using an exact method. Earlier research (mentioned in Section 2) showed that we can only find an exact solution if we keep our instances small enough. Therefore, we will also solve this problem using heuristic methods as proposed by Murray and Chu (2015). Furthermore, we will change the assumption of the constant flight endurance to a flight endurance depending on the specifications of the UAV and its package. Therefore this paper will contribute to a shorter total delivery time (compared to truck-only deliveries), a more realistic model of the FSTSP by incorporating a variable flight endurance and a clear comparison between the heuristic and exact solving method on smaller instances.

The remainder of this paper is structured as follows. In Section 2, we will provide some literature related to this problem. After that, in Section 3, we will give a formal definition and the MILP model to solve this FSTSP using exact methods. In Section 4, we will present a global overview of the heuristics we will use to solve instances. Section 5 will introduce the payload-dependent flight endurance and explain the changes required for its implementation. In Section 6, we present our Warm Start Algorithm. Section 7 gives us the results of the methods proposed earlier sections. Finally, Section 8 will give a conclusion based on our results.

2 Related Literature

The problem we want to solve in this paper is an extension of the basic Travelling Salesman Problem (TSP). Similar to this is the Vehicle Routing Problem (VRP), where they use multiple vehicles to visit the customers. Although lots of literature can be found on all kinds of VRPs and TSPs, almost none of them is directly applicable to our FSTSP. According to Drexl (2012), the FSTSP belongs to the category *movement synchronisation en route.* This is because the UAV may join or separate from the truck while it is on its route.

If we look at earlier articles where drones are not involved yet, the work presented by Lin (2011) might be the closest related to ours. It presents a pickup and delivery problem in which heavy resources (e.g. trucks or vans) may transport both deliverables and other lighter vehicles (here they refer to scooters and foot couriers). However, there are some differences. First of all, they use a VRP, which means that there are multiple trucks, where we use a TSP with only one truck. Furthermore, we use a UAV as the ‘light vehicle’, where they used foot couriers and scooters. Also, the UAV is restricted to visit only one customer, where the foot couriers and scooters could visit multiple customers. The drone may also join/separate from the truck multiple times, where in Lin (2011) this was limited to only one time. Due to constraints on the payload the drone, some packages may be too heavy and have to be transported by the truck. Lastly, our UAV has a limited flight endurance, which we have to respect.

Murray and Chu (2015) are the first involving UAVs in the optimisation of a delivery process. It is also the basis for many other works, including this paper. Ferrandez et al. (2016), for example, investigated the same problem and created some heuristics based on k-means and a genetic algorithm. Ponza (2016) wrote a master thesis based on the work of Murray and Chu and solved the problem for different types of drones using Simulated Annealing, a metaheuristic which is good in real-world applications. De Freitas and Penna (2018) used Variable Neighbourhood Search to come up with a good heuristic for the FSTSP.

A slightly different problem is the Travelling Salesman Problem with Drones (TSP-D), where the launching and recovering point of the UAV are allowed to be the same, and the drones follow the same road network as the truck. Agatz et al. (2018) created an integer linear programming model for the TSP-D and used a route-first cluster-second heuristic based on local search and dynamic programming. Bouman et al. (2018) proposed an exact method based on dynamic programming which can solve larger instances (with 20 nodes) than any exact methods presented earlier. Ha et al. (2018) used a MILP-formulation like the one in the FSTSP, but used it on the TSP-D, where they developed two heuristic algorithms (*cluster first route second* and a *route first cluster second*) for instances of 10, 50 and 100 customers.

More recently, Schermer et al. (2019) proposes an extension of the VRP, where they implement drones and en route operations (VRPDERO). Here, the drones may not only be launched and recovered at customer locations but also at discrete points in-between two customers. For solving large-scale instances, they use the concepts of Variable Neighbourhood Search (VNS) together with Tabu Search. Karak and Abdelghany (2019) presented a hybrid vehicle-drone routing problem (HVDRP) for pickup and delivery services, which minimises the total routing costs, respecting the constraints for the UAV flight range and weight capacity. They benchmarked the performance of their heuristic against a vehicle-driven routing heuristic and a drone-driven routing heuristic. Lastly, Jeong et al. (2019) proposed another MILP model for the FSTSP whereby energy consumption and time-dependent no-fly zones are incorporated. This gives us a more realistic view, especially since no-fly zones (e.g. for weather conditions or sensitive facilities) is a crucial regulation set by the Federal Aviation Administration (FAA).

3 The Flying Sidekick Travelling Salesman Problem

In this section, we will give the formal definition and mathematical formulation of our FSTSP. The model is based on Murray and Chu (2015). In the FSTSP one truck and one UAV travel from the depot to serve all customers and return to the depot while minimising this last arrival time. Customers should be visited exactly once, and neither the drone or the truck may revisit any customers. At each stop (either a customer or the depot), the UAV can be launched or recovered, where the launching and retrieving points are not the same. The UAV is restricted to serve only one customer at a time. After a sortie, we will replace the battery of the UAV (which we incorporate in the launch/recover service time). The flight endurance may not be exceeded and is assumed to be constant (this will be revised in Section 5). First, we introduce the notation we will use in our formulations.

Sets	
$C = \{1, ..., c\}$	The set of customers
$C' \subseteq C$	Customers eligible for UAV-service
$N = \{0, 1, ..., c+1\}$	The set of all nodes, where 0 and $c+1$ are the depot
$N_d = \{0, 1, ..., c\} \subseteq N$	The set of departing nodes
$N_a = \{1, 2, ..., c+1\} \subseteq N$	The set of arriving nodes
P	The set of tuples, where a tuple is denoted as $\langle i, j, k \rangle$ with the condition that $i \in N_d$ is the launch point, $j \in \{C' : j \neq i\}$ the UAV-eligible customer that will be delivered by the drone and $k \in \{N_a : k \neq i, k \neq j, \tau'_{ij} + \tau'_{jk} \leq e\}$ the rendezvous point
Parameters	
τ_{ij}	Travel time from $i \in N_d$ to $j \in N_a$ by truck
τ'_{ij}	Travel time from $i \in N_d$ to $j \in N_a$ by UAV
s_L	Time needed to launch the UAV from the truck
s_R	Time needed to recover the UAV back on the truck
e	Flight endurance of the UAV
M	Very large positive constant
Variables	
$x_{ij} \in \{0, 1\}$	Equal to one if the truck travels from $i \in N_d$ to $j \in \{N_a : i \neq j\}$
$y_{ijk} \in \{0, 1\}$	Equal to one if the UAV is launched at $i \in N_d$, delivers a package at customer $j \in \{C' : j \neq i\}$, and returns to the truck at $k \in \{N_a : \langle i, j, k \rangle \in P, k \neq i, k \neq j\}$
t_j	Time when the truck arrives at $j \in N_a$
t'_j	Time when the UAV arrives at $j \in N_a$
p_{ij}	Equal to one if the truck visits $i \in C$ before $j \in C$, where $i \neq j$
u_i	Position of $i \in N_a$ in the truck's path

With the notation introduces above, we will formulate a mixed integer linear program (MILP) formulation to model our flying sidekick travelling salesman problem using (1) - (32).

$$\min \quad t_{c+1} \tag{1}$$

$$\text{s.t.} \sum_{j \in N_a} x_{0j} = 1 \tag{2}$$

$$\sum_{i \in N_d} x_{i,c+1} = 1 \tag{3}$$

$$\sum_{\substack{i \in N_d \\ i \neq j}} x_{ij} + \sum_{\substack{i \in N_d \\ i \neq j}} \sum_{\substack{k \in N_a \\ \langle i,j,k \rangle \in P}} y_{ijk} = 1 \qquad \forall j \in C \tag{4}$$

$$\sum_{\substack{i \in N_d \\ i \neq j}} x_{ij} = \sum_{\substack{k \in N_a \\ k \neq j}} x_{jk} \qquad \forall j \in C \tag{5}$$

$$u_i - u_j + 1 \leq (c+2)(1 - x_{ij}) \qquad \forall i \in C, j \in \{N_a : j \neq i\} \tag{6}$$

$$u_k - u_i \geq 1 - (c+2)\left(1 - \sum_{\substack{j \in C \\ \langle i,j,k \rangle \in P}} y_{ijk}\right) \qquad \forall i \in C, k \in \{N_a : k \neq i\} \tag{7}$$

$$\sum_{\substack{j \in C \\ j \neq i}} \sum_{\substack{k \in N_a \\ \langle i,j,k \rangle \in P}} y_{ijk} \leq 1 \qquad \forall i \in N_d \tag{8}$$

$$\sum_{\substack{i \in N_d \\ i \neq k}} \sum_{\substack{j \in C \\ \langle i,j,k \rangle \in P}} y_{ijk} \leq 1 \qquad \forall k \in N_a \tag{9}$$

$$2y_{ijk} \leq \sum_{\substack{h \in N_d \\ h \neq i}} x_{hi} + \sum_{\substack{l \in C \\ l \neq k}} x_{lk} \qquad \forall i \in C, j \in \{C : j \neq i\}, k \in \{N_a : \langle i,j,k \rangle \in P\} \tag{10}$$

$$y_{0jk} \leq \sum_{\substack{h \in N_d \\ h \neq k}} x_{hk} \qquad \forall j \in C, k \in \{N_a : \langle 0,j,k \rangle \in P\} \tag{11}$$

$$t'_i \geq t_i - M \left(1 - \sum_{\substack{j \in C \\ j \neq i}} \sum_{\substack{k \in N_a \\ \langle i,j,k \rangle \in P}} y_{ijk} \right) \qquad \forall i \in C \qquad (12)$$

$$t'_i \leq t_i + M \left(1 - \sum_{\substack{j \in C \\ j \neq i}} \sum_{\substack{k \in N_a \\ \langle i,j,k \rangle \in P}} y_{ijk} \right) \qquad \forall i \in C \qquad (13)$$

$$t'_k \geq t_k - M \left(1 - \sum_{\substack{i \in N_d \\ i \neq k}} \sum_{\substack{j \in C \\ \langle i,j,k \in P \rangle}} y_{ijk} \right) \qquad \forall k \in N_a \qquad (14)$$

$$t'_k \leq t_k + M \left(1 - \sum_{\substack{i \in N_d \\ i \neq k}} \sum_{\substack{j \in C \\ \langle i,j,k \in P \rangle}} y_{ijk} \right) \qquad \forall k \in N_a \qquad (15)$$

$$t_k \geq t_h + \tau_{hk} + s_L \left(\sum_{\substack{l \in C \\ l \neq k}} \sum_{\substack{m \in N_a \\ \langle k,l,m \rangle \in P}} y_{klm} \right) + s_R \left(\sum_{\substack{i \in N_d \\ i \neq k}} \sum_{\substack{j \in C \\ \langle i,j,k \rangle \in P}} y_{ijk} \right) - M\left(1 - x_{hk}\right) \qquad \forall h \in N_d, k \in \{N_a : k \neq h\} \qquad (16)$$

$$t'_j \geq t'_i + \tau'_{ij} + s_L - M \left(1 - \sum_{\substack{k \in N_a \\ \langle i,j,k \rangle \in P}} y_{ijk} \right) \qquad \forall j \in C', i \in \{N_d : i \neq j\} \qquad (17)$$

$$t'_k \geq t'_j + \tau'_{jk} + s_R - M \left(1 - \sum_{\substack{i \in N_d \\ \langle i,j,k \rangle \in P}} y_{ijk} \right) \qquad \forall j \in C', k \in \{N_a : k \neq j\} \qquad (18)$$

$$t'_k - \left(t'_j - \tau'_{ij}\right) \leq e + M\left(1 - y_{ijk}\right) \qquad \forall k \in N_a, j \in \{C : j \neq k\}, i \in \{N_d : \langle i,j,k \rangle \in P\} \qquad (19)$$

$$u_i - u_j \geq 1 - (c+2)p_{ij} \qquad \forall i \in C, j \in \{C : j \neq i\} \qquad (20)$$

$$u_i - u_j \leq -1 + (c+2)\left(1 - p_{ij}\right)$$

$$\forall i \in C, j \in \{C : j \neq i\} \quad (21)$$

$$p_{ij} + p_{ji} = 1$$

$$\forall i \in C, j \in \{C : j \neq i\} \quad (22)$$

$$p_{0j} = 1$$

$$\forall j \in C \quad (23)$$

$$t'_l \geq t'_k - M \left(3 - \sum_{\substack{j \in C \\ \langle i,j,k \rangle \in P \\ j \neq l}} y_{ijk} - \sum_{\substack{m \in C \\ m \neq i \\ m \neq k \\ m \neq l}} \sum_{\substack{n \in N_a \\ \langle l,m,n \rangle \in P \\ n \neq i \\ n \neq k}} y_{lmn} - p_{il} \right)$$

$$\forall i \in N_d, k \in \{N_a : k \neq i\}, l \in \{C : l \neq i, l \neq k\} \quad (24)$$

$$t_o = 0 \quad (25)$$

$$t'_0 = 0 \quad (26)$$

$$x_{ij} \in \{0, 1\}$$

$$\forall i \in N_d, j \in \{N_a : j \neq i\} \quad (27)$$

$$y_{ijk} \in \{0, 1\}$$

$$\forall i \in N_d, j \in \{C : j \neq i\}, k \in \{N_a : \langle i, j, k \rangle \in P\} \quad (28)$$

$$1 \leq u_i \leq c + 2$$

$$\forall i \in N_a \quad (29)$$

$$t_i \geq 0$$

$$\forall i \in N \quad (30)$$

$$t'_i \geq 0$$

$$\forall i \in N \quad (31)$$

$$p_{ij} \in \{0, 1\}$$

$$\forall i \in N_d, j \in \{C : j \neq i\} \quad (32)$$

The objective function (1) minimises the latest arrival time at the depot. The first Constraints (2) and (3) make sure that at the depot exactly one truck departs and returns respectively. Constraint (4) ensures that each customer is visited exactly once by either the truck or the UAV. If the truck arrives at a customer, Constraint (5) makes sure it also departs from it. Constraints (6) and (7) eliminate any subtours for the truck and drone respectively. Constraints (8) and (9) ensure that the UAV may launch/recover at each departing/arriving node respectively, and that this operation will happen at most once per node. If the drone is launched at i and retrieved at k, then Constraint (10) makes sure that the truck is assigned to both i and k. In case the

launching node is the depot, we will use Constraint (11) instead.
Constraints (12) and (13) ensure that the truck and drone are arrive at the same time for the launch at customer i. In a similar way, Constraints (14) and (15) will do this for the rendezvous at arriving node k. Constraints (12) - (15) make the assumption that the launching and retrieving point cannot be the same and a launch/rendezvous may only happen once per node. Constraint (16) calculates the effective arrival time of the truck, including the the travel time and launch/recover service times (if applicable). Similarly, Constraint (17) will calculate the arrival time of the UAV at customer j, incorporating the UAV's travel time to this customer. Constraint (18) makes sure that time required for the recovery of the UAV cannot be started earlier than the rendezvous of the truck and drone.
Furthermore, the UAV cannot fly to a customer and back to another node if this tour exceeds the battery capacity of the UAV. Therefore we have Constraint (19) that checks the flight endurance for this tour. Constraints (20) - (23) ensure the right order in the sequence of nodes. For example, a truck can only travel to customers that are not visited yet. Constraint (24) prevents the drone from launching at a departure node when the drone is already flying to another customer. Constraints (25) and (26) initialise the departure time of the truck and drone at the depot to 0. Finally, our last Constraints (27) - (32) specify the definition of the decision variables and their boundaries.

4 Heuristic

The exact model as described in Section 3 will solve the FSTSP to optimality. However, as Murray and Chu (2015) shows, the optimal value of the exact model for instances with only ten customers cannot be found within 30 minutes. It may take several hours to come up with a provable optimal solution (Jeong et al., 2019). Therefore, we clearly need heuristic approaches to solve (large-scale) instances in a time-efficient way. The heuristic we use is based on Murray and Chu (2015). The general idea is a local search and saving heuristic. With this, we will route the customers first and then reassign them to the UAV or in other places in the truck route. These procedures will then continue until we can found no further improvement. Our heuristic method consists of multiple algorithms.

For each algorithm, we provide the pseudocode and explain what we will do in this algorithm. First we start with our `main` function, which is Algorithm 1. Since our problem is an extension of the travelling salesman problem (TSP), we will use this to initialise our heuristic. Appendix A.1 contains the mathematical formulation of our basic TSP model. We will use `solveTSP(C)` to call the sequence of nodes visited by the truck and the arrival time of each of these nodes from the TSP results. As we start our mode, we will set the maximum savings to 0, and create the set *Cprime*, which is a copy of all UAV-eligible customers.

Algorithm 1 Main

```
1: Initialise:
        $Cprime \leftarrow$ C' % Make a copy of the set of UAV-eligible customers
        $[truckRoute, t] \leftarrow$ solveTSP(C)
        $truckSubRoutes \leftarrow \{truckRoute\}$
        $maxSavings \leftarrow 0$
2: repeat
3:     for all $j \in Cprime$ do
4:         Calculate calcSavings($j, t$)
5:         for all $subroute$ in $truckSubRoutes$ do
6:             if $subroute$ has an UAV-sortie then
7:                 Calculate calcCostTruck($j, t, subroute$)
8:             else
9:                 Calculate calcCostUAV($j, t, subroute$)
10:            end if
11:        end for
12:    end for
13:    if $maxSavings > 0$ then
14:        Call performUpdate
15:        $maxSavings \leftarrow 0$
16:    else
17:        STOP
18:    end if
19: until Stop
```

For each customer, we will calculate the savings of removing it from the truck route in line 4. We will do this using the `calcSavings` function in Algorithm 2. Each subroute in our heuristic is a sequence of nodes that is part of the bigger truck route. For example, if we have a UAV sortie, then a subroute includes the launch and rendezvous point and all the truck nodes in-between them. For each subroute, we will check in line 7 if it is already using the drone for delivery. If so, we will try to insert the removed customer into this subroute, where it will be delivered by the truck. The cost of this operation will be determined by the `calcCostTruck` function (Algorithm 3). If the drone is not used in this subroute, we will try to deliver the removed customer by drone. The cost for this operation is calculated by the `calcCostUAV` function (Algorithm 4). If our savings are more than our costs, we will update the solution using `performUpdate` in Algorithm 5. After this update, we will reset the maximum savings in line 15 en repeat the procedure until we cannot find any further improvement.

In Algorithm 2 we will start by finding the nodes directly before and after customer j (the node

Algorithm 2 calcSavings

```
Require: $j$(customer assigned to the truck) and $t$(truck's arrival time at each node)
 1: Find $i$, the node visited directly before $j$ in the truck's route
 2: Find $k$ the node visited directly after $j$ in the truck's route
 3: $savings \leftarrow \tau_{ij} + \tau_{jk} - \tau_{ik}$
 4: Find $subroute$ in $truckSubRoutes$ which contains customer $j$
 5: if $subroute$ has an UAV-sortie then
 6:     Find $a$, the first node in $subroute$ (launching point)
 7:     Find $c$, the last node in $subroute$ (rendezvous point)
 8:     Find $b$, the customer visited by the UAV that launches at $a$ and returns to $b$
 9:     Calculate $t^\star_c$, the truck's arrival time at $c$ if $j$ is removed from the truck's route
10:     $savings \leftarrow \min\{savings, t^\star_c - (t_a + \tau'_{ab} + \tau'_{bc} + s_R)\}$
11: end if
```

that we try to reassign), we will call them node i and k respectively. Then, we will calculate the savings of removing customer j, using the formula in line 3. After that, we will check if the subroute which contains customer j has a UAV-sortie. If it has not, this algorithm is finished. Otherwise, we will find the first and last node in this subroute and the customer visited by the drone (line 6-8). We will call these nodes a and c and b respectively. Then we will calculate $t^\star_c$, the arrival time at c if we remove customer j. The total savings will be the minimum of the one found before and the difference between the old and new arrival time at c (as can be seen in line 10).

Focusing on Algorithm 3, we try to insert customer j at another place in the truck's route and calculate the costs for this operation. We look for all adjacent nodes in the subroute and calculate the cost of inserting customer j between these two adjacent nodes (line 4). If the costs are less than the savings, we will check in line 6 whether the drone's flight assigned to the subroute is still feasible. If this is true and the savings minus the costs are greater than the maximum saving thus far, we will save this change as lines 8-10 show.

Instead of inserting customer j at another place in the route, Algorithm 4 will calculate the costs associated with serving this customer by UAV. We will look at each pair of nodes in the subroutes (line 1) and check if an UAV-sortie to customer j does not exceed the flight endurance (restriction in line 2). If it does, there will be no change and the algorithm ends. Otherwise, we continue and calculate the truck's arrival time if j is removed from the truck's route to the UAV (line 3). Then we can calculate the cost of this operation using the formula in line 4. If the savings minus the costs are higher than the maximum savings thus far and the effective flight endurance of the drone (including service and wait times) is not exceeded (line 5), line 6-8 will save the change.

Algorithm 3 calcCostTruck

```
Require: $j, t, subroute$
1: Find $a$, the first node in $subroute$
2: Find $c$, the last node in $subroute$
3: for all adjacent $i$ and $k$ in $subroute$ do
4:     $cost \leftarrow \tau_{ij} + \tau_{jk} - \tau_{ik}$
5:     if $cost < savings$ then
6:         if $t_c - t_a + cost \leq e$ then
7:             if $savings - cost > maxSavings$ then
8:                 $servedByUAV \leftarrow false$
9:                 $i^\star \leftarrow i$
                   $j^\star \leftarrow j$
                   $k^\star \leftarrow k$
10:                $maxSavings \leftarrow savings - cost$
11:            end if
12:        end if
13:    end if
14: end for
```

Algorithm 4 calcCostUAV

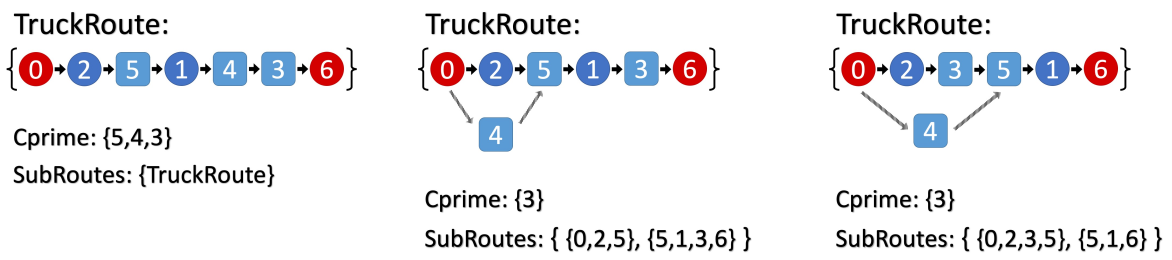
```
Require: $j, t, subroute$
1: for all $i$ and $k$ in $subroute$, where $i$ is visited before $k$ do
2:     if $\tau'_{ij} + \tau'_{jk} + s_L + s_R \leq e$ then
3:         Calculate $t^\star_k$, the truck's arrival time at $k$ if $j$ is removed from the truck's route
4:         $cost \leftarrow \max\left\{0, \max\left\{(t^\star_k - t_i) + s_L + s_R, \tau'_{ij} + \tau'_{jk} + s_L + s_R\right\} - (t^\star_k - t_i)\right\}$
5:         if $savings - cost > maxSavings$ and $t^\star_k - t_i + s_L + s_R \leq e$ then
6:             $servedByUAV \leftarrow true$
7:             $i^\star \leftarrow i$
               $j^\star \leftarrow j$
               $k^\star \leftarrow k$
8:             $maxSavings \leftarrow savings - cost$
9:         end if
10:    end if
11: end for
```

Algorithm 5 performUpdate

```
Require: $servedByUAV, i^\star, j^\star, k^\star$
 1: if $servedByUAV == true$ then
 2:     Assign the UAV to $i^\star \to j^\star \to k^\star$
 3:     Remove $j^\star$ from $truckRoute$ and all subroutes in $truckSubRoutes$
 4:     Create a new truck subroute that starts at $i^\star$ and end at $k^\star$ and add it to $truckSubRoutes$
 5:     Remove $i^\star, j^\star$ and $k^\star$ from $Cprime$
 6:     Update $t$, the vector of truck arrival times to each node
 7: else
 8:     Remove $j^\star$ from its current truck subroute
 9:     Insert $j^\star$ between $i^\star$ and $k^\star$ in the new truck subroute
10:     Update $truckRoute$ to reflect the new sequence of nodes visited
11:     Update $t$
12: end if
```

Algorithm 5 is the final part of our heuristic. Here we will perform the update suggested by the `calcSavings` and `calcCostUAV/calcCostTruck` functions. Boolean *servedByUAV* and nodes $i^\star, j^\star, k^\star$ are assumed to be the results of Algorithm 1 - 4. We consider two procedures. First, we look at the case were *servedByUAV* was set to *true*, which means that Algorithm 4 caused the change with the least costs. We will start by assigning the UAV to the new sortie ($i^\star \to j^\star \to k^\star$). After that, we will remove UAV-served customer $j^\star$ from the truck's route and all its subroutes. We then create a new subroute from $i^\star$ to $k^\star$ with the UAV-sortie to $j^\star$. Nodes $i^\star, j^\star, k^\star$ need to be removed from *Cprime* as they cannot be delivered by UAV anymore. Finally, we update the arrival time for each node. In the second procedure, the least-cost change was a reorder in the truck's route by Algorithm 3. Here we start with removing customer $j^\star$ from its current subroute and inserting it between $i^\star$ and $k^\star$. We then update the truck's route and the arrival times for each node.

Figure 2 is an example that gives a clear view of what Algorithms 1 - 5 will do. Initially, we start with the TSP solution, where all customers are assigned to the truck, as can be seen in 2a. Figure 2b shows us the situation after we executed `calcCostUAV`, where we assigned a UAV sortie to $i^\star = 0, j^\star = 4, k^\star = 5$. Since node 4 is served by the drone, and node 5 is a rendezvous point, we cannot assign these nodes to a UAV anymore. Therefore, we update *Cprime*. We also see that the list of subroutes is partitioned because of the UAV-sortie. Lastly, we apply `calcCostTruck` function, which brings us to the situation in Figure 2c. Customer $j^\star = 3$ has been removed from its original position and is now inserted between $i^\star = 2$ and $k^\star = 5$.



(a) TSP solution, before the heuristic, all the customers are assigned to the truck

(b) FSTSP solution after `calcCostUAV`, customer 4 is assigned to the UAV now

(c) FSTSP solution after `calcCostTruck`, customer 3 has been moved in the truck's route

Figure 2: An example to demonstrate the FSTSP heuristic. The red nodes (0 and 7) represent the depot. The boxed (light blue) nodes are UAV-eligible customers and the circular (darker blue) nodes are UAV-ineligible customers

5 Payload-Dependent Flight Endurance

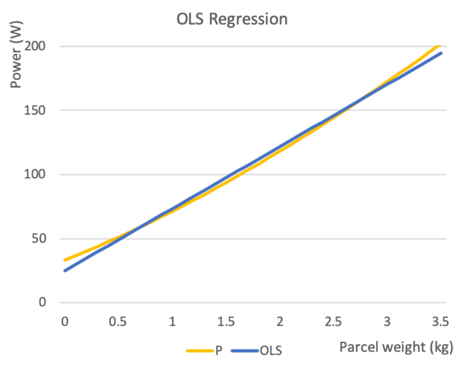
In Section 3 and 4 we assumed that our flight endurance was constant. In this section we will make the flight endurance variable by incorporating the power consumption of the drone. If a package is heavy it will take more power than a package that is light. According to Dorling et al. (2016) the power P (in Watts) of a drone can be calculated with Equation (33), where we use the frame weight W (in kg), parcel weight w (in kg), gravity g (in Newton), fluid density of air ρ (in kg/m^3), the area of the proppeler ς (in m^2) and the number of rotors n.

$$P = (W + w)^{3/2} \sqrt{\frac{g^3}{2\rho\varsigma n}} \tag{33}$$

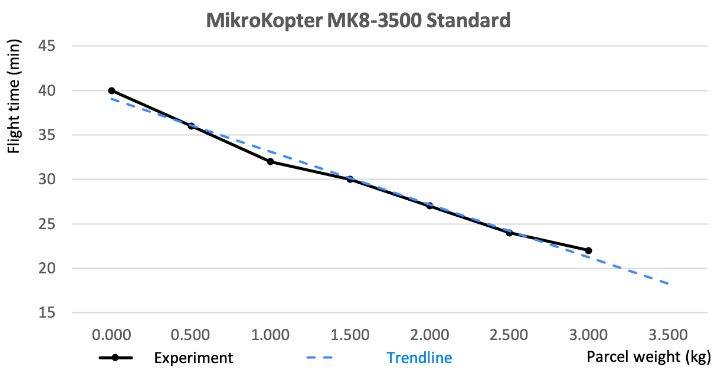
If we look at Figure 3a, we see that there exists a near-linear relation between the parcel weight and the required energy. Therefore we apply Ordinary Least Squares (OLS) to Equation (33) (source code can be found in Appendix B.1). OLS estimates the parameters in linear Equation (34). The maximum flight time (in minutes) will then be calculated using Equation (35), where η, C, V_n and P are the energy conversion efficiency, battery capacity (in mAh), nominal battery voltage (in Volts) with n rotors and power consumption (in Watts), respectively. We assume the energy conversion efficiency η to be 70%. The data we use is based on an 'MK8-3500 standard' drone, which can carry parcels up to 3500g. Its specifications can be found in MikroKopter (2019). In Figure 3b, we can see that our theoretical formulas (the trendline) match the data retrieved from some real flight experiments with this 'MK8-3500' drone.

$$p(w_j) = \alpha w_j + \beta \tag{34}$$

$$flightTime = \frac{\eta C V_n}{P} \tag{35}$$



(a) OLS regression on Eq. (33)



(b) Theoretical and experimental flight time

Figure 3: OLS regression and flight times

We can now implement equation (34) in our exact MILP-model because of its linearity. The new constraint will be as follows:

$$p(w_j)\left(t'_j - t'_i\right) + p(0)\left(t'_k - t'_j\right) \leq \eta E + M\left(1 - y_{ijk}\right)$$
$$\forall i \in N_d, j \in \{C' : j \neq i\}, k \in \{N_a : j \neq k, \langle i,j,k\rangle \in P\} \qquad (36)$$

In the heuristic we will create a function called `power(`w_j, i, j, k`)` which will check the constraint above. However, as the arrival time will be calculated later than this check, we will use that $t'_j = t_i + \tau'_{ij} + s_L$ for launching point i and UAV-eligible customer j.

6 Warm Start

As an extension of our flying sidekick travelling salesman problem, we will develop a method that uses warm starts. In our algorithm we choose to combine the MILP-formulation from (1)-(32) and the heuristic in Algorithm 1 - 5. We will first start with solving our heuristic and save the results we found. After that, we will switch to the exact method and load the results we just saved into the solver. The solver will see this as a warm start, which means that the solution will be no worse than the one found in the heuristic. As we solve our MILP-formulation with the same conditions as we did in Section 3, it will guarantee us to get a solution that is better than or equal to the current best solution (either the MILP-formulation or the heuristic). The pseudocode for our warm start algorithm can be found below in Algorithm 6.

Algorithm 6 Warm Start Algorithm

```
1: Create a new FSTSP in CPLEX, we will call this $fstsp1$
2: Load the data from the instance
3: Call $Main$ (Algorithm 1)
4: for all x- and y-variables in the heuristic solution equal to 1 do
5:     Create a new constraint in $fstsp1$ which set the lowerbound of the variable to 1
6: end for
7: Solve $fstsp1$
8: Save the results as a MIP-start
9: Create another FSTSP in CPLEX, which we call $fstsp2$
10: Load the MIP-start into the solver
11: Solve $fstsp2$
```

7 Results

7.1 Instances

For the evaluation of our performance, we have generated 72 instances. This generation is based on the procedure described by Murray and Chu (2015). All these instances consist of 10 customers and a depot uniformly randomly distributed over an 8 by 8 miles square. For each customer, we have the x- and y-coordinates and whether this customer is UAV-eligible (which is between 80% and 90%). For the formulation (1) - (32), the heuristic (Algorithm 1 - 5) and the warm start approach as described in Section 6, we choose a fixed flight endurance of 20 minutes. The truck and UAV speed were assumed to be both 25 miles/hour. The travel time of the truck (τ_{ij}, in minutes) between node i and j is calculated using Equation (37), where we use the Manhattan metric distance. The drone's travel time (τ'_{ij}, also in minutes) uses the Euclidean distance and can be calculated using Equation (38). The launch and retrieve service times were both set to 1 minute. For the exact MILP-formulation, we set a time limit of 30 minutes per instance.

$$\tau_{ij} = \frac{|X_i - X_j| + |Y_i - Y_j|}{\text{speedTruck}} \cdot 60 \qquad \forall i, j \in N \tag{37}$$

$$\tau'_{ij} = \frac{\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}}{\text{speedUAV}} \cdot 60 \qquad \forall i, j \in N \tag{38}$$

The MILP-formulation in (1) - (32) and (33) was solved via CPLEX 12.9. Both the MILP-formulation and the heuristics were programmed in Java Version 8 Update 211. For instance 1 to 36 the computation was performed on a MacBook Pro with a 2.9 GHz Intel Core i5-6267U processor and 8GB RAM. The computation of instance 37 to 72 was performed on an Acer TC710 with a 2.7 GHz Intel Core i5-6400 processor and 8GB RAM.

7.2 Flying Sidekick Travelling Salesman Problem

We start by discussing the results of the Mixed Integer Linear Programming formulation (1)-(32) in Section 3. In Table 2 (FSTSP-MILP), we can see that the average objective value has been improved by 7.2628 minutes (9.91%) when we use drones in the parcel delivery process. It also shows us that the best reduction was about 23%. There was only one instance that we could not improve; the use of drones did not lead to a reduction in the total delivery time for this specific set of customers. Note that for each instance, CPLEX required the full 30 minutes limit, which means that we cannot prove that our solutions are optimal. Moreover, we got a lower bound of 0 for all 72 instances. This is because our LP-relaxation is very weak. This relaxation will first visit all the solutions where node 0 and 11 (both the depot) are connected since there is no travel time between these nodes and this is the cheapest solution for the LP-relaxation. If the truck's first node is the depot itself, the arrival time at node 11 will be zero, which causes the objective value to be 0 as well. Nevertheless, will we use these solutions as a benchmark when we evaluate the other methods.

As it may take several hours to solve our MILP-formulation, we created a heuristic method in Algorithm 1 - 5. All instances had a computational time of less than a second, which is a huge improvement compared to the exact approach. As we can see in Table 2, the heuristic (FSTSP-Heuristic) also leads to solutions with a higher objective value than the MILP. On average, a delivery process took 2.4577 minutes extra, which is an increase of 3.89%. However, there are four instances in which the heuristic solution was better than the exact solution (due to the 30-min limit). In another fifteen instances, the solutions were equal. Altogether, this means the heuristic performs quite well. The results of each instance can be found in Table 3 in Appendix C.1.

7.3 Payload-dependent flight endurance

As an extension on the flying sidekick travelling salesman problem we decided to change fixed flight endurance to a variable payload-dependent flight endurance. Section 5 mentions that we use Ordinary Least Squares (OLS) to derive Equation (34). In this equation, α is the power consumption per kilogram of parcel weight (in Watts per kg) and β is the power we need to keep the UAV-frame in the air (in Watts). Applying OLS shows us that $\hat{\alpha} = 39.9982$, with a standard error of $\sigma_\alpha = 0.06584$ and $\hat{\beta} = 129.0528$, with a standard error of $\sigma_\beta = 0.13314$. The weights of the packages are uniformly randomly distributed between 0 and 3.5 kg. Table 7 in Appendix D contains the weights we used for each node in each instance.

For this extension, we use both the MILP-formulation and the heuristic algorithms with the extra constraints mentioned in Section 5. In Table 2 we see that the average objective value (FSTSP-PF-MILP and FSTSP-PF-Heuristic) is lower than the solutions from Section 7.2. This is not a surprise since the mean parcel weight of 1.75 kg leads to a maximum flight endurance of 26:12 minutes, which is higher than the fixed 20 minutes we set in Section 3. These better solutions are not caused by better solution methods, since we use the same MILP-formulation and heuristics, and the flight endurance is the only change we made. We also see that in most instances the customers delivered by the UAV are the ones with lighter packages, and thus with a greater flight endurance. Figure 4 shows that a lower average parcel weight does not always lead to a greater reduction. This means the reduction is most probably caused by the choice of customers delivered by UAV (lighter parcels, less waiting time at rendezvous, etc.) and the routing. If we look closer at the results, we also see that the average number of customers served by UAV increases (from 2.06 to 2.26) when we use payload-dependent flight endurance.

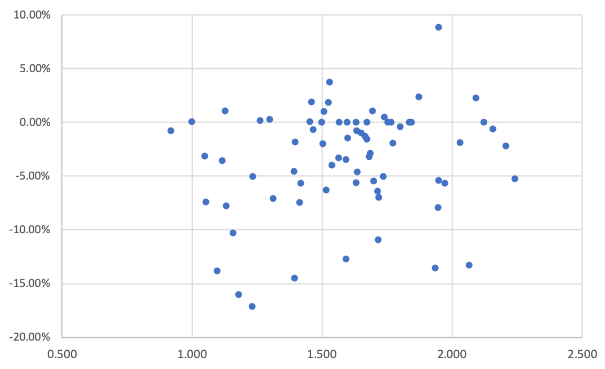


Figure 4: Relation between the average parcel weight (horizontal axis) and the reduction compared to the FSTSP with fixed flight endurance (vertical axis)

7.4 FSTSP with Warm Starts

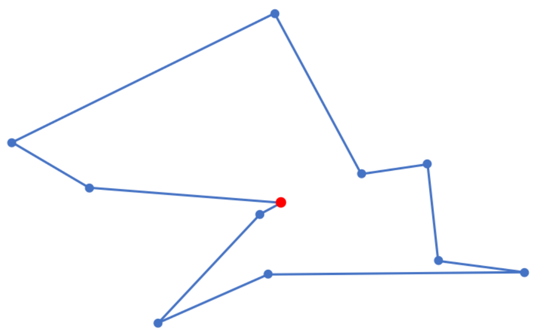
In our paper, we use the solutions of our heuristic method as a warm start in the MIP-solver. This means the solutions of our Warm Start Algorithm will be, in any way, no worse than our heuristic solutions. As the solver keeps everything (except the warm start) the same as in Section 3, we will not get any result worse than the ones of the MILP-formulation. Table 2 can show this, since the results are better than those in Section 7.2. Our Warm Start Algorithm gave better solutions for nine instances, with reductions up to 6.5% compared to our best solution thus far. Besides, some of the instances even got a lower bound, where the lower bounds from Section 7.2 were all 0. The results for all the instances individually can be found in Table 5 in Appendix C.3.

7.5 Comparison of the results

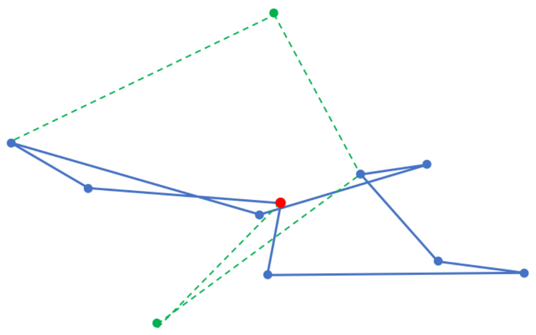
We also want to get a closer look at the results themselves. Therefore, Figure 5 gives a visual representation of instance 52, with the solutions of the TSP, FSTSP-MILP, FSTSP-Heuristic and FSTSP-Warm Start. The red node represents the depot, the blue line is the truck route, and the green lines represent the UAV-sorties. Figure 5a shows us the basic TSP, where all customers are assigned to the truck, and the drone is not used. Figure 5b is the result of our

MILP-formulation after 30 minutes, which is a good reduction of the total travel time. In Figure 5c, we can see the solution of the heuristic, which also a good reduction, but not as good as the exact method. Finally, Figure 5d is the best solution, which is the result of our Warm Start Algorithm that combines the solutions of Figure 5b and 5c. The visual representation also explains one of the reasons why the heuristic does not always lead to the same solutions as the MILP-formulation. Moreover, the heuristic is based on a TSP solution, where crossovers (like figure 5b and 5d) are not allowed. Therefore, the heuristic ignores all solutions with crossovers.

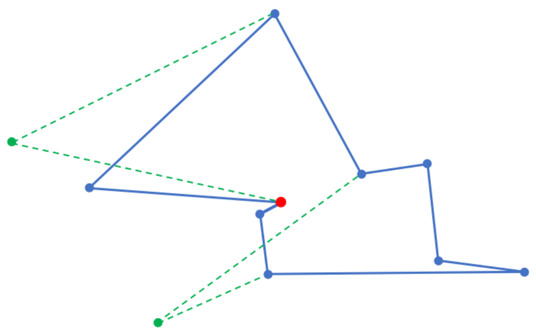
Lastly, we decided to run the Warm Start Algorithm one more time, but without the restricted time limit. This means that we solve the instances to optimality, where we have no gap between the lower and upper bound. The mean solving time was 3 hours and 20 minutes. The exact results can be found in Table 6 in Appendix C.4, Table 2 gives the averages for this 'FSTSP-Optimal' method.



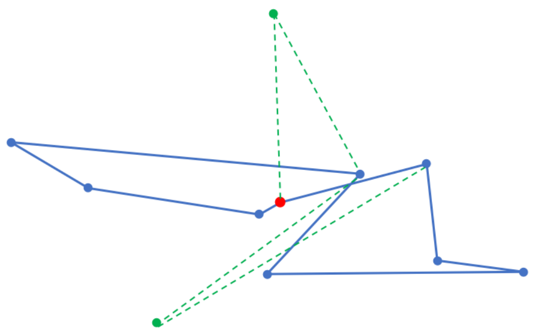
(a) TSP solution (57.4869 min)



(b) FSTSP with MILP-formulation, solution after 30 minutes (46.8147 min)



(c) FSTSP heuristic solution (48.1359 min)



(d) FSTSP solution with warm start algorithm (44.6444 min)

Figure 5: Visual representation of the different solution approaches applied on instance 52

Table 2: Comparison of our solutions with the travelling salesman problem without drones

Solution Approach	Objective Value	Gap TSP (%)		
	Average	Average	Min	Max
TSP (A.1)	73.4731			
FSTSP-MILP (7.2)	66.2103	-9.91	-23.42	0.00
FSTSP-Heuristic (7.2)	68.6680	-6.55	-21.94	0.00
FSTSP-PF-MILP (7.3)	63.8651	-13.01	-31.55	-0.21
FSTSP-PF-Heuristic (7.3)	66.1905	-10.00	-25.33	0.00
FSTSP-Warm Start (7.4)	65.9473	-10.29	-23.42	0.00
FSTSP-Optimal (7.5)	65.7948	-10.40	-23.42	0.00

8 Conclusion

International companies like Amazon, DHL and many others are developing and testing the usage of these drones, which have shown their potential. In this thesis, we analysed the Flying Sidekick Travelling Salesman Problem (FSTSP), where a truck can carry a UAV for their last-mile delivery. As we know, drones have some weakness in their limited flight time and payload. Therefore, we incorporated the payload-dependent flight endurance, where a lighter package means that the drone can cover a greater distance. Also, we sought to provide a new Warm Start Algorithm that combines the heuristic and exact approaches. We introduced formal definitions and mathematical formulations to optimise this kind of operations in the logistics sector. Because our FSTSP is very time-consuming to solve, only smaller instances can be solved to optimality. In our results, we found that the average solving time, for an instance of only ten customers and a depot, is already 3 hours and 20 minutes. Therefore, we came up with a heuristic method that solves the same instance is just one second. Although most objective values were higher than the MILP-formulation, there were still 19 instances (26.4%) that had an objective value equal to or even lower than the exact method. On average, there was an increase of 2:30 minutes (only 3.89%). The results from Murray and Chu (2015) have an average gap of 8.33%/4:48 minutes, which is similar to ours. The difference might be the fact that we used different instances. Besides, the Warm Start Algorithm we use gives better solutions than the MILP-formulation alone.

The change from a fixed to a payload-dependent flight endurance led to a better average objective value, and also increased the average number of customers served by UAV. After our MILP-formulation and heuristic algorithms, we introduced a Warm Start Algorithm. This algorithm was designed such that none of the solutions could be worse than the ones we found in the heuristic or exact methods. The implementation of this Warm Start Algorithm led to further reductions up to 6.5% compared to our best solution thus far. Jeong et al. (2019) pointed out that solving their instances (also ten customers and a depot, distributed across an 8 by 8 miles square) to optimality took over 8 hours on average. This paper reduced this mean solving time to 3 hours and 20 minutes, mainly because of our Warm Start Algorithm. For all our solution approaches, there is one thing very clear, the use of drones in a parcel delivery process is very helpful. We got significant decreases (10.3% on average) in the total delivery time, with reductions going up to 23%, compared to the standard Travelling Salesman Problem without any drones.

Although our models give satisfying results, we still have some limitations. For instance, we limited our instances to 10 customers, and real-world problems such as traffic jams and one-way streets can still influence the truck's travel time. Battery replacement times are also assumed to be constant. For the UAV's battery consumption, we used the specifications of the 'MK8-3500' delivery drone, while there are many drones on the market with other specifications that can also represent our problem. The heuristic we use is a local search and savings heuristic and, once it has assigned the UAV to the launch, delivery and rendezvous node, these nodes will not be checked for any other savings later on. Future research can be done in many aspects. This can range from simply testing larger instances to extending the problem itself (for example, considering multiple trucks or UAVs). Charging the battery en route, instead of replacing a new one can lead to different results that might be more realistic. We have also limited our problem to launching and retrieving the drone at customer nodes, while in the future we can also investigate any point in-between customers. Some companies might want to reduce the total travelling costs instead of the delivery time. Therefore, we can also change the objective goal in future research.

Appendix A Travelling Salesman Problem

A.1 IP Formulation

$$\min \sum_{i \in N_d} \sum_{\substack{j \in N_a \\ j \neq i}} \tau_{ij} x_{ij} \tag{39}$$

$$\text{s.t.} \sum_{\substack{i \in N_d \\ i \neq j}} x_{ij} = 1 \qquad \forall j \in N_a \tag{40}$$

$$\sum_{\substack{j \in N_a \\ j \neq i}} x_{ij} = 1 \qquad \forall i \in N_d \tag{41}$$

$$u_i - u_j + (n-1)\, x_{ij} = n - 2 \qquad \forall i \in N_a, j \in \{N_a : j \neq i\} \tag{42}$$

Appendix B Code

B.1 MATLAB Code for Ordinary Least Square

```
%% Load data
W = 1.500;
w = [0:0.01:3.5]';
g = 9.81;
rho = 1.204;
sigma = 0.2;
n = 6;
l = length(w);
X = [w ones(l,1)];
m = 2;

%% Formula for the power
P = (W+w).^(3/2) * sqrt(g^3/(2*rho*sigma*n));

%% OLS estimator
beta = (X'*X)\X'*P;

%% Statistics for the regression
e = P - X*beta;          % Disturbances
sse = e'*e;              % Sum of Square Errors (SSE)
s2 = sse/(l - m);        % Sigma^2 estimate
var = inv(X'*X)*s2;      % Variance of the OLS estimate
se = sqrt(diag(var));    % Standard Error (SE) of the estimates
```

Appendix C Results

C.1 MILP-formulation and heuristic

Table 3: FSTSP and Heuristic Results

Instance	TSP	FSTSP-MILP		Heuristic		
	Objective Value	Objective Value	Gap TSP (%)	Objective Value	Gap TSP (%)	Gap FSTSP (%)
1	67.256	57.166	-15.00	57.166	-15.00	0.00
2	79.374	73.132	-7.86	74.375	-6.30	1.70
3	73.426	63.001	-14.20	62.339	-15.10	-1.05
4	72.640	62.974	-13.31	63.752	-12.24	1.24
5	85.585	75.578	-11.69	78.922	-7.79	4.42
6	63.828	52.284	-18.09	57.330	-10.18	9.65
7	81.195	70.188	-13.56	77.192	-4.93	9.98
8	62.749	57.037	-9.10	58.239	-7.19	2.11
9	77.773	71.230	-8.41	73.234	-5.84	2.81
10	74.587	73.045	-2.07	73.677	-1.22	0.87
11	78.749	73.308	-6.91	73.308	-6.91	0.00
12	68.587	65.714	-4.19	68.587	0.00	4.37
13	77.115	67.418	-12.58	69.512	-9.86	3.11
14	72.601	65.843	-9.31	65.843	-9.31	0.00
15	67.476	61.167	-9.35	64.373	-4.60	5.24
16	70.486	65.817	-6.62	66.966	-4.99	1.75
17	76.784	62.536	-18.56	69.170	-9.92	10.61
18	82.726	74.556	-9.88	75.133	-9.18	0.77
19	55.533	53.780	-3.16	52.781	-4.96	-1.86
20	85.826	82.488	-3.89	81.722	-4.78	-0.93
21	68.896	58.870	-14.55	67.734	-1.69	15.06
22	70.414	64.128	-8.93	64.986	-7.71	1.34
23	75.590	72.007	-4.74	72.007	-4.74	0.00
24	74.672	67.725	-9.30	67.725	-9.30	0.00
25	53.690	45.395	-15.45	53.571	-0.22	18.01
26	74.314	59.489	-19.95	67.951	-8.56	14.22
27	75.104	73.969	-1.51	74.803	-0.40	1.13
28	76.284	65.287	-14.42	67.417	-11.62	3.26
29	56.455	48.263	-14.51	48.263	-14.51	0.00
30	58.285	49.390	-15.26	54.614	-6.30	10.58
31	74.898	74.898	0.00	74.898	0.00	0.00
32	70.073	53.664	-23.42	61.979	-11.55	15.50
33	66.354	64.552	-2.71	64.552	-2.71	0.00
34	69.211	57.674	-16.67	63.903	-7.67	10.80
35	81.598	75.528	-7.44	80.536	-1.30	6.63
36	81.402	70.986	-12.80	74.663	-8.28	5.18

37	77.854	69.971	-10.13	73.033	-6.19	4.38
38	78.980	70.897	-10.23	75.869	-3.94	7.01
39	62.796	58.694	-6.53	58.694	-6.53	0.00
40	75.922	70.747	-6.82	71.702	-5.56	1.35
41	72.505	63.048	-13.04	71.574	-1.28	13.52
42	67.689	66.406	-1.90	66.823	-1.28	0.63
43	53.501	50.613	-5.40	50.613	-5.40	0.00
44	66.616	65.296	-1.98	66.571	-0.07	1.95
45	74.174	65.613	-11.54	68.343	-7.86	4.16
46	66.464	61.424	-7.58	61.424	-7.58	0.00
47	69.184	60.113	-13.11	63.010	-8.92	4.82
48	64.800	55.299	-14.66	59.373	-8.37	7.37
49	76.152	72.667	-4.58	73.142	-3.95	0.65
50	92.591	80.734	-12.81	82.752	-10.63	2.50
51	65.181	64.983	-0.30	64.983	-0.30	0.00
52	57.487	46.815	-18.56	48.136	-16.27	2.82
53	75.558	68.292	-9.62	72.089	-4.59	5.56
54	71.500	64.743	-9.45	66.106	-7.54	2.10
55	81.120	68.417	-15.66	75.356	-7.11	10.14
56	79.733	72.908	-8.56	72.908	-8.56	0.00
57	69.661	56.956	-18.24	54.377	-21.94	-4.53
58	60.622	54.950	-9.36	59.979	-1.06	9.15
59	80.961	74.964	-7.41	74.964	-7.41	0.00
60	88.617	74.977	-15.39	77.123	-12.97	2.86
61	80.528	74.398	-7.61	76.491	-5.01	2.81
62	77.120	71.633	-7.11	75.204	-2.48	4.98
63	76.443	72.678	-4.93	73.500	-3.85	1.13
64	77.713	69.435	-10.65	72.738	-6.40	4.76
65	77.965	74.940	-3.88	74.940	-3.88	0.00
66	77.910	73.340	-5.87	74.247	-4.70	1.24
67	77.737	71.421	-8.12	72.418	-6.84	1.40
68	84.609	80.325	-5.06	81.009	-4.25	0.85
69	88.411	77.999	-11.78	83.254	-5.83	6.74
70	74.253	67.266	-9.41	67.266	-9.41	0.00
71	89.876	79.926	-11.07	88.543	-1.48	10.78
72	76.294	64.163	-15.90	72.317	-5.21	12.71
Average	73.473	66.210	-9.91	68.668	-6.55	3.89

C.2 Payload-dependent flight endurance

Table 4: Results with payload-dependent flight endurance

Instance	Variable Flight Endurance (MILP)			Variable Flight Endurance (Heuristic)		
	Objective Value	Gap TSP (%)	Gap FSTSP (%)	Objective Value	Gap TSP (%)	Gap MILP (%)
1	55.529	-17.44	-2.86	59.901	-10.94	7.87
2	61.395	-22.65	-16.05	68.811	-13.31	12.08
3	59.821	-18.53	-5.05	60.399	-17.74	0.97
4	59.580	-17.98	-5.39	65.877	-9.31	10.57
5	73.180	-14.49	-3.17	78.922	-7.79	7.85
6	52.837	-17.22	1.06	57.330	-10.18	8.50
7	71.481	-11.96	1.84	72.843	-10.29	1.90
8	56.695	-9.65	-0.60	60.444	-3.67	6.61
9	69.670	-10.42	-2.19	69.670	-10.42	0.00
10	71.661	-3.92	-1.90	71.661	-3.92	0.00
11	72.733	-7.64	-0.78	71.843	-8.77	-1.22
12	65.714	-4.19	0.00	68.587	0.00	4.37
13	66.096	-14.29	-1.96	67.736	-12.16	2.48
14	65.598	-9.65	-0.37	63.732	-12.22	-2.84
15	59.003	-12.56	-3.54	59.003	-12.56	0.00
16	65.321	-7.33	-0.75	63.954	-9.27	-2.09
17	62.632	-18.43	0.15	71.282	-7.17	13.81
18	74.556	-9.88	0.00	78.128	-5.56	4.79
19	52.102	-6.18	-3.12	52.781	-4.96	1.30
20	68.365	-20.35	-17.12	71.251	-16.98	4.22
21	64.078	-6.99	8.85	63.797	-7.40	-0.44
22	60.104	-14.64	-6.28	60.639	-13.88	0.89
23	64.140	-15.15	-10.93	64.140	-15.15	0.00
24	63.394	-15.10	-6.39	71.361	-4.43	12.57
25	45.858	-14.59	1.02	44.516	-17.09	-2.93
26	50.866	-31.55	-14.50	55.490	-25.33	9.09
27	73.484	-2.16	-0.66	74.803	-0.40	1.79
28	65.988	-13.50	1.07	67.316	-11.76	2.01
29	48.263	-14.51	0.00	48.263	-14.51	0.00
30	45.704	-21.59	-7.46	48.278	-17.17	5.63
31	69.594	-7.08	-7.08	74.898	0.00	7.62
32	53.664	-23.42	0.00	61.346	-12.45	14.32
33	59.788	-9.89	-7.38	55.017	-17.09	-7.98
34	55.012	-20.52	-4.62	62.897	-9.12	14.33
35	75.759	-7.16	0.31	77.402	-5.14	2.17
36	72.665	-10.73	2.36	68.712	-15.59	-5.44
37	64.434	-17.24	-7.91	65.024	-16.48	0.92
38	61.275	-22.42	-13.57	72.891	-7.71	18.96

39	58.694	-6.53	0.00	58.694	-6.53	0.00
40	70.039	-7.75	-1.00	68.747	-9.45	-1.84
41	64.263	-11.37	1.93	67.351	-7.11	4.81
42	66.406	-1.90	0.00	66.823	-1.28	0.63
43	50.613	-5.40	0.00	50.613	-5.40	0.00
44	64.030	-3.88	-1.94	66.571	-0.07	3.97
45	62.032	-16.37	-5.46	62.513	-15.72	0.78
46	60.313	-9.25	-1.81	61.424	-7.58	1.84
47	61.497	-11.11	2.30	59.030	-14.68	-4.01
48	54.589	-15.76	-1.28	53.048	-18.14	-2.82
49	68.847	-9.59	-5.26	68.847	-9.59	0.00
50	76.682	-17.18	-5.02	88.863	-4.03	15.88
51	65.045	-0.21	0.10	65.045	-0.21	0.00
52	42.000	-26.94	-10.28	43.415	-24.48	3.37
53	70.843	-6.24	3.74	67.720	-10.37	-4.41
54	59.717	-16.48	-7.76	59.717	-16.48	0.00
55	64.592	-20.38	-5.59	70.016	-13.69	8.40
56	67.832	-14.93	-6.96	75.524	-5.28	11.34
57	49.079	-29.55	-13.83	55.129	-20.86	12.33
58	55.236	-8.89	0.52	52.950	-12.66	-4.14
59	73.882	-8.74	-1.44	76.756	-5.19	3.89
60	71.543	-19.27	-4.58	77.103	-12.99	7.77
61	70.194	-12.83	-5.65	73.608	-8.59	4.86
62	69.149	-10.34	-3.47	70.408	-8.70	1.82
63	72.678	-4.93	0.00	73.500	-3.85	1.13
64	65.494	-15.72	-5.68	72.139	-7.17	10.15
65	71.952	-7.71	-3.99	74.940	-3.88	4.15
66	63.586	-18.38	-13.30	65.491	-15.94	3.00
67	69.070	-11.15	-3.29	72.418	-6.84	4.85
68	80.325	-5.06	0.00	81.009	-4.25	0.85
69	76.784	-13.15	-1.56	81.416	-7.91	6.03
70	67.305	-9.36	0.06	67.305	-9.36	0.00
71	79.926	-11.07	0.00	79.926	-11.07	0.00
72	56.012	-26.58	-12.70	66.717	-12.55	19.11
Average	63.865	-13.01	-3.45	66.191	-10.00	3.73

C.3 Warm Start Algorithm

Table 5: Results of the Warm Start Algorithm

Instance	Warm Start Algorithm								
	Objective Value	Gap TSP (%)	Gap Best(%)	LB		Objective Value	Gap TSP(%)	Gap Best(%)	LB
1	57.166	-15.00	0.00	0	37	69.971	-10.13	0.00	0
2	73.132	-7.86	0.00	0	38	70.897	-10.23	0.00	0
3	62.339	-15.10	0.00	0	39	58.694	-6.53	0.00	0
4	62.570	-13.86	-0.64	0	40	70.747	-6.82	0.00	0
5	75.578	-11.69	0.00	0	41	63.048	-13.04	0.00	0
6	52.284	-18.09	0.00	0	42	66.406	-1.90	0.00	0
7	70.188	-13.56	0.00	0	43	50.613	-5.40	0.00	0
8	56.695	-9.65	-0.60	0	44	65.296	-1.98	0.00	0
9	71.230	-8.41	0.00	0	45	65.613	-11.54	0.00	0
10	73.045	-2.07	0.00	0	46	61.424	-7.58	0.00	0
11	73.308	-6.91	0.00	0	47	59.824	-13.53	-0.48	0
12	65.714	-4.19	0.00	1.917	48	53.048	-18.14	-4.07	0
13	67.418	-13.32	-0.86	0	49	72.667	-4.58	0.00	0
14	65.843	-9.31	0.00	0	50	80.734	-12.81	0.00	0
15	61.167	-9.35	0.00	0	51	64.983	-0.30	0.00	0
16	65.817	-6.62	0.00	0	52	44.644	-22.34	-4.64	0
17	62.536	-18.56	0.00	0	53	68.292	-9.62	0.00	0
18	74.556	-9.88	0.00	0	54	64.743	-9.45	0.00	6.221
19	51.547	-7.18	-2.34	0	55	68.413	-15.66	-0.01	0
20	76.397	-10.99	-6.52	0	56	72.908	-8.56	0.00	7.625
21	58.870	-14.55	0.00	0	57	54.377	-21.94	0.00	0
22	63.526	-9.78	-0.94	0	58	54.950	-9.36	0.00	0
23	72.007	-4.74	0.00	0	59	74.964	-7.41	0.00	0
24	67.725	-9.30	0.00	0	60	74.977	-15.39	0.00	0
25	45.395	-15.45	0.00	0	61	74.398	-7.61	0.00	20.646
26	59.489	-19.95	0.00	0	62	71.633	-7.11	0.00	4.675
27	73.969	-1.51	0.00	0	63	72.678	-4.93	0.00	0
28	65.287	-14.42	0.00	0	64	69.435	-10.65	0.00	0
29	48.263	-14.51	0.00	0	65	74.940	-3.88	0.00	0
30	49.390	-15.26	0.00	0	66	73.340	-5.87	0.00	0
31	74.898	0.00	0.00	0	67	70.117	-9.80	-1.83	0
32	53.664	-23.42	0.00	0	68	80.325	-5.06	0.00	0
33	64.552	-2.72	0.00	0	69	77.999	-11.78	0.00	6.708
34	57.674	-16.67	0.00	0	70	67.266	-9.41	0.00	0
35	75.528	-7.44	0.00	0	71	79.926	-11.07	0.00	0
36	70.986	-12.80	0.00	0	72	64.163	-15.90	0.00	0
Average	65.947	-10.29	-0.31	0.664					

C.4 Optimal solution compared to other methods

Table 6: Optimal Solutions from the Warm Start Algorithm without time limit, compared to the other solutions methods. Solving time is given in hours

Instance	Objective Value	Gap TSP(%)	Gap FSTSP(%)	Gap Heuristic (%)	Gap Warm Start Alg. (%)	Solving Time
1	57.166	-15.00	0.00	0.00	0.00	02:22
2	73.132	-7.86	0.00	-1.67	0.00	01:48
3	62.339	-15.10	-1.05	0.00	0.00	02:42
4	61.668	-15.10	-2.07	-3.27	-1.44	02:50
5	75.578	-11.69	0.00	-4.24	0.00	02:54
6	51.345	-19.56	-1.80	-10.44	-1.80	05:39
7	70.188	-13.56	0.00	-9.07	0.00	02:23
8	56.695	-9.65	-0.60	-2.65	0.00	02:26
9	71.230	-8.41	0.00	-2.74	0.00	02:55
10	73.045	-2.07	0.00	-0.86	0.00	02:13
11	73.039	-7.25	-0.37	-0.37	-0.37	04:26
12	65.714	-4.19	0.00	-4.19	0.00	03:49
13	67.418	-12.58	0.00	-3.01	0.00	03:12
14	65.843	-9.31	0.00	0.00	0.00	01:38
15	61.167	-9.35	0.00	-4.98	0.00	04:11
16	65.817	-6.62	0.00	-1.72	0.00	03:38
17	62.536	-18.56	0.00	-9.59	0.00	02:25
18	74.556	-9.88	0.00	-0.77	0.00	02:15
19	51.547	-7.18	-4.15	-2.34	0.00	03:36
20	76.397	-10.99	-7.38	-6.52	0.00	02:03
21	58.870	-14.55	0.00	-13.09	0.00	02:04
22	63.515	-9.80	-0.96	-2.26	-0.02	01:46
23	72.007	-4.74	0.00	0.00	0.00	02:59
24	67.725	-9.30	0.00	0.00	0.00	03:38
25	45.395	-15.45	0.00	-15.26	0.00	03:45
26	57.056	-23.22	-4.09	-16.03	-4.09	03:36
27	73.969	-1.51	0.00	-1.12	0.00	02:22
28	65.287	-14.42	0.00	-3.16	0.00	03:37
29	48.263	-14.51	0.00	0.00	0.00	07:10
30	49.390	-15.26	0.00	-9.57	0.00	03:18
31	74.898	0.00	0.00	0.00	0.00	04:24
32	53.664	-23.42	0.00	-13.42	0.00	09:59
33	64.552	-2.72	0.00	0.00	0.00	03:50
34	57.674	-16.67	0.00	-9.75	0.00	02:06
35	75.528	-7.44	0.00	-6.22	0.00	01:50
36	70.986	-12.80	0.00	-4.93	0.00	02:26
37	69.971	-10.13	0.00	-4.19	0.00	03:02
38	70.032	-11.33	-1.22	-7.69	-1.22	08:58

39	58.694	-6.53	0.00	0.00	0.00	05:29
40	70.747	-6.82	0.00	-1.33	0.00	02:10
41	63.048	-13.04	0.00	-11.91	0.00	03:23
42	66.406	-1.90	0.00	-0.62	0.00	09:48
43	50.613	-5.40	0.00	0.00	0.00	04:09
44	65.296	-1.98	0.00	-1.92	0.00	02:30
45	65.613	-11.54	0.00	-3.99	0.00	03:47
46	61.424	-7.58	0.00	0.00	0.00	02:53
47	59.824	-13.53	-0.48	-5.06	0.00	04:06
48	53.048	-18.14	-4.07	-10.65	0.00	03:39
49	72.667	-4.58	0.00	-0.65	0.00	01:53
50	80.734	-12.81	0.00	-2.44	0.00	02:46
51	64.983	-0.30	0.00	0.00	0.00	03:25
52	43.244	-24.78	-7.63	-10.16	-3.14	03:58
53	68.292	-9.62	0.00	-5.27	0.00	01:39
54	64.743	-9.45	0.00	-2.06	0.00	03:51
55	68.413	-15.66	-0.01	-9.21	0.00	04:36
56	72.908	-8.56	0.00	0.00	0.00	01:56
57	54.377	-21.94	-4.53	0.00	0.00	02:42
58	54.950	-9.36	0.00	-8.38	0.00	02:50
59	74.964	-7.41	0.00	0.00	0.00	01:59
60	74.977	-15.39	0.00	-2.78	0.00	02:53
61	74.398	-7.61	0.00	-2.74	0.00	00:49
62	71.633	-7.11	0.00	-4.75	0.00	01:28
63	72.678	-4.93	0.00	-1.12	0.00	03:32
64	69.435	-10.65	0.00	-4.54	0.00	03:57
65	74.940	-3.88	0.00	0.00	0.00	03:36
66	73.338	-5.87	0.00	-1.22	0.00	03:27
67	70.117	-9.80	-1.83	-3.18	0.00	04:11
68	80.325	-5.06	0.00	-0.84	0.00	01:22
69	76.784	-13.15	-1.56	-7.77	-1.56	04:01
70	67.266	-9.41	0.00	0.00	0.00	01:45
71	79.926	-11.07	0.00	-9.73	0.00	01:59
72	63.647	-16.58	-0.80	-11.99	-0.80	03:45
Average	65.829	-10.40	-0.58	-4.14	-0.18	03:20

Appendix D Weights

Table 7: Weights (kg) assigned to the packages

Instance	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9	Node 10	Average
1	3.308	0.499	1.126	0.534	1.809	2.445	0.814	2.797	2.308	2.888	1.853
2	1.239	1.291	0.612	2.022	0.486	2.763	0.812	1.573	0.923	1.251	1.297
3	0.745	0.708	3.259	2.309	0.704	3.400	1.206	3.346	2.522	0.871	1.907
4	2.833	1.556	2.902	2.751	2.682	3.333	2.132	0.957	2.180	0.105	2.143
5	2.785	2.508	0.988	1.300	3.089	0.096	2.409	1.125	1.755	2.435	1.849
6	0.017	0.344	0.814	3.485	2.096	1.818	2.155	1.352	0.151	0.161	1.239
7	1.394	1.598	1.557	0.861	1.033	0.472	1.705	2.458	3.218	2.472	1.677
8	2.912	0.645	2.870	2.542	2.510	3.354	2.451	3.488	1.623	1.325	2.372
9	1.389	2.278	0.441	3.440	1.718	3.142	3.026	2.595	2.954	3.275	2.426
10	2.090	3.308	3.065	1.761	3.260	1.163	0.610	3.006	3.360	0.700	2.232
11	1.784	2.251	0.739	3.448	0.216	1.408	1.405	1.986	2.190	2.520	1.795
12	0.979	1.180	1.025	2.049	1.074	1.925	3.197	2.911	0.953	3.100	1.839
13	1.627	1.593	2.241	0.395	2.959	0.633	1.259	1.583	1.505	2.735	1.653
14	0.179	1.495	2.105	2.299	1.103	3.252	2.817	0.042	3.494	3.016	1.980
15	1.059	0.331	1.967	1.607	2.063	1.275	0.400	0.071	2.379	1.120	1.227
16	1.124	0.690	0.837	1.111	0.692	1.000	0.722	3.417	0.009	0.499	1.010
17	1.468	1.063	1.855	0.399	1.573	1.594	0.882	1.665	2.700	0.679	1.388
18	1.488	1.422	1.599	1.657	1.575	0.814	3.324	2.917	2.565	0.570	1.793
19	0.483	0.724	0.458	1.021	1.363	1.316	1.609	2.103	2.249	0.201	1.153
20	2.521	0.473	2.351	0.920	2.145	0.023	0.139	3.133	0.262	1.568	1.354
21	1.331	1.966	0.925	3.340	3.199	3.467	1.335	0.543	2.356	2.957	2.142
22	2.454	1.303	1.817	1.231	0.645	2.546	0.504	0.009	3.143	3.027	1.668
23	1.251	1.334	3.006	1.750	3.486	1.045	0.253	2.666	2.376	1.703	1.887
24	3.444	1.823	2.265	3.195	0.825	1.821	2.496	0.633	2.064	0.288	1.885
25	0.063	2.679	1.747	2.862	0.920	2.693	0.201	2.325	2.806	0.266	1.656
26	3.112	0.716	1.918	2.687	0.753	0.555	1.776	0.464	0.756	2.604	1.534
27	3.073	1.217	0.641	1.178	0.547	3.094	2.404	1.176	0.050	2.743	1.612
28	2.616	1.049	0.039	2.944	2.139	0.939	1.150	0.889	3.399	3.472	1.864
29	3.497	1.313	0.110	1.128	1.080	3.160	1.815	2.262	3.051	2.005	1.942
30	0.253	3.185	3.050	0.902	0.884	1.532	3.286	1.744	0.141	0.581	1.556
31	2.107	0.949	2.073	1.488	0.690	0.938	0.865	1.612	1.671	2.037	1.443
32	3.256	2.662	0.318	2.496	1.985	2.249	2.976	2.602	1.229	0.497	2.027
33	0.347	1.499	0.521	0.461	0.159	0.360	2.326	2.771	2.816	0.322	1.158
34	0.748	3.227	0.059	3.397	2.291	0.791	0.013	0.740	3.210	3.498	1.797
35	3.235	0.224	1.558	2.721	0.244	2.912	2.748	0.392	0.023	0.217	1.427
36	3.153	1.139	2.818	0.906	3.279	1.373	2.387	3.408	1.938	0.185	2.059
37	1.319	2.784	1.698	2.533	2.412	3.379	3.420	0.563	3.056	0.229	2.139
38	2.944	3.041	3.406	0.145	1.965	2.244	3.494	2.077	1.720	0.237	2.127
39	0.834	3.016	1.995	3.147	1.285	0.999	1.656	1.979	1.563	1.081	1.756
40	2.206	0.091	1.600	2.325	2.420	1.175	2.592	1.424	3.259	1.057	1.815
41	1.620	1.510	0.149	0.472	0.420	3.477	2.121	2.507	1.681	2.101	1.606
42	2.474	0.545	0.269	2.156	3.126	2.914	2.953	1.451	0.666	0.662	1.722

Instance	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9	Node 10	Average
43	1.307	3.391	1.367	3.021	3.402	1.368	2.160	2.656	0.434	1.072	2.018
44	3.454	2.648	0.507	3.460	2.489	0.667	3.306	0.269	2.089	0.610	1.950
45	2.561	1.461	2.127	1.783	1.792	1.144	3.412	0.929	3.220	0.239	1.867
46	1.427	3.441	1.729	1.724	0.308	1.236	1.226	2.014	1.229	1.014	1.535
47	3.377	2.369	3.318	0.807	2.717	1.207	2.785	3.088	2.995	0.349	2.301
48	0.630	2.798	0.460	3.489	1.809	2.943	0.289	1.744	3.182	0.981	1.833
49	2.814	3.366	2.438	2.982	0.525	3.293	1.543	3.147	1.647	2.890	2.465
50	2.170	0.065	1.302	1.101	1.877	1.027	1.228	0.916	1.460	2.418	1.357
51	0.104	1.876	2.161	2.659	1.788	2.611	0.559	2.780	0.883	0.561	1.598
52	0.877	0.793	1.714	0.923	0.549	1.104	1.476	0.704	2.783	1.808	1.273
53	2.681	2.377	0.001	2.347	1.234	0.892	0.159	3.498	1.640	1.971	1.680
54	2.290	0.219	2.341	0.930	1.849	2.965	0.792	0.773	0.061	0.222	1.244
55	0.101	2.894	1.416	1.559	1.303	0.407	2.678	3.487	2.761	1.340	1.795
56	1.702	1.149	2.161	1.445	0.305	3.209	2.118	1.791	1.759	3.250	1.889
57	0.321	1.110	1.851	2.174	0.223	2.346	0.577	0.668	1.113	1.666	1.205
58	1.370	3.185	1.411	2.379	1.535	2.788	0.946	2.130	1.003	2.389	1.914
59	1.898	0.529	3.319	0.734	1.204	1.430	2.591	2.564	1.116	2.181	1.757
60	0.805	0.204	0.709	2.459	0.242	1.650	2.549	1.656	2.045	2.999	1.532
61	0.345	0.079	3.269	3.369	2.618	2.564	2.136	3.129	1.277	2.896	2.168
62	0.844	1.679	2.955	0.983	2.225	0.091	3.038	2.858	0.709	2.132	1.751
63	0.386	0.582	2.532	3.017	3.251	2.276	3.117	3.193	2.924	2.067	2.334
64	0.029	1.286	3.004	1.855	1.762	2.786	0.349	2.574	0.939	1.008	1.559
65	0.099	2.222	2.694	1.950	1.519	2.262	0.253	2.437	2.093	1.382	1.691
66	3.127	1.551	1.506	3.332	1.248	2.713	2.980	0.703	2.150	3.409	2.272
67	3.149	3.335	0.777	0.398	0.960	1.729	3.413	1.396	0.034	2.000	1.719
68	2.469	2.867	1.144	1.131	1.043	0.466	3.336	1.339	2.769	2.709	1.927
69	3.108	2.771	2.647	0.681	2.140	2.602	2.450	1.002	0.769	0.228	1.840
70	1.201	1.323	0.730	0.998	0.086	3.443	1.179	0.084	0.334	1.609	1.099
71	2.180	1.166	2.488	3.448	0.351	3.173	0.479	0.094	2.823	0.262	1.647
72	0.217	1.483	2.845	3.064	3.466	1.418	0.092	1.633	0.415	2.867	1.750
Average	1.720	1.631	1.691	1.939	1.593	1.899	1.793	1.834	1.818	1.580	1.750

Appendix E Programs and Files

Folder	**File**	**Description**
/src/FSTSP		Source files for the MILP-formulation
	Graph.java	Graph that keeps track of all nodes
	Main.java	Main class to read/write files and start the solver
	Node.java	Represents a node in our problem with its properties
	TSP.java	MILP model and CPLEX solver
/src/VarFlightE		Source files for the MILP-formulation with payload-dependent flight endurance
	Main2.java	Main class to read/write files and start the solver
	TSP2.java	MILP model and CPLEX solver

/src/HeurVarFlightE		Source files for the heuristic with payload-dependent flight endurance
	Main.java	Main method with all the heuristic algorithms
	OnlyResults.java	Main.java, with only the objective value as output
/src/HeurFSTSP		Source files for the heuristic of the FSTSP
	Main.java	Main method with all the heuristic algorithms
	OnlyResults.java	Main.java, with only the objective value as output
/src/TSP		Source files for the TSP IP-formulation (without drones)
	Main3.java	Main class to read/write files and start the solver for a single instance
	TSP.java	IP model and CPLEX solver for the TSP for a single instance
	All_Instances.java	Main class to read/write files and start the solver for all instances
	TSP_All.java	IP model and CPLEX solver for the TSP for all instances
/src/WSA		Source files for the Warm Start Algorithm of the FSTSP
	Main_WSA	Solver for Warm Start Algoritm for reading/writing files and solving the heuristc
	Main_Windows.java	Main_WSA class, with some minor changes so it can run on Windows systems
	TSP.java	MILP model and CPLEX solver
/instances		Data for the instances (coordinates, uav-eligible)
/lib		CPLEX files required for the solver
/solutionFSTSP		Solutions of the FSTSP with MILP-formulation
/solutionFSTSPheur		Solutions of heuristic of the FSTSP
/solutionFSTSPext		Solutions of the FSTSP with payload-dependent flight endurance
/FSTSPheur		Solutions of the heuristic with payload-dependent flight endurance
/weights		Weights used for the payload-dependent flight endurance
/solutionTSP		Solutions of the TSP without drones
/WarmStart		Solutions of the Warm Start Algorithm
/noTimeLimit		Optimal solutions of the Warm Start Algorithm without time limit

References

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

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

Carlson, N. (2013). The real reason amazon announced delivery drones last night: $3 million in free advertising on cyber monday. `https://www.businessinsider.com/why-amazon-announced-delivery-drones-2013-12?international=true&r=US&IR=T`. [Online; accessed 15-May-2019].

Clement, J. (2019). Statista: E-commerce worldwide - statistics & facts. `https://www.statista.com/topics/871/online-shopping/`. [Online; accessed 4-July-2019].

De Freitas, J. C. and Penna, P. H. V. (2018). A randomized variable neighborhood descent heuristic to solve the flying sidekick traveling salesman problem. *Electronic Notes in Discrete Mathematics*, 66:95–102.

Deutsche Post AG (2019). Deutsche Post DHL Group — DHL Parcelcopter. `https://www.dpdhl.com/en/media-relations/specials/dhl-parcelcopter.html`. [Online; accessed 14-May-2019].

Dorling, K., Heinrichs, J., Messier, G. G., and Magierowski, S. (2016). Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85.

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

Federal Aviation Administration (2019). Unmanned aircraft systems (uas). `https://www.faa.gov/uas/`. [Online; accessed 15-May-2019].

Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., and Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management (JIEM)*, 9(2):374–388.

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

Hern, A. (2014). DHL launches first commercial drone 'parcelcopter' delivery service. `https://www.theguardian.com/technology/2014/sep/25/german-dhl-launches-first-commercial-drone-delivery-service`. [Online; accessed 15-May-2019].

Jeong, H. Y., Song, B. D., and Lee, S. (2019). Truck-drone hybrid delivery routing: Payload-energy dependency and no-fly zones. *International Journal of Production Economics*, 214:220 – 233.

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

Lawler, E. (1985). *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley & Sons.

Lin, C. (2011). A vehicle routing problem with pickup and delivery time windows, and coordination of transportable resources. *Computers & Operations Research*, 38(11):1596–1609.

MikroKopter (2019). MikroKopter - Technical Specifications. `http://www.mikrokopter.de/en/products/nmk8stden/nmk8techdaten`. [Online; accessed 22-June-2019].

Murhpy, M. (2016). The future is here: Drones are delivering domino's pizzas to customers. `https://qz.com/838254/dominos-is-delivering-pizza-with-autonomous-drones-to-customers-in-new-zealand/`. [Online; accessed 15-May-2019].

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

Perlow, B. (2016). Amazon completes 1st drone delivery. `https://abcnews.go.com/US/amazon-completes-drone-delivery/story?id=44185981`. [Online; accessed 1-July-2019].

Ponza, A. (2016). Optimization of drone-assisted parcel delivery.

Schermer, D., Moeini, M., and Wendt, O. (2019). A hybrid VNS/Tabu Search algorithm for solving the vehicle routing problem with drones and en route operations. *Computers & Operations Research*.