The Flexible Delivery Problem with Time Windows

Master Thesis
Econometrics and Management Science

Florian Maas[*]
Supervisor: Dr. R. Spliet
Co-reader: Prof.Dr. A.P.M. Wagelmans

Erasmus School of Economics
Erasmus University Rotterdam
October 2014

Abstract

In this thesis, we propose the Flexible Delivery Problem with Time Windows (FDPTW) as a new member to the family of vehicle routing problems. In this problem, there is a set of parcels with multiple potential delivery locations and corresponding time windows that have to be delivered by a set of vehicles originating from a single depot. Since this is a new problem, we create various sets of benchmark instances for this problem to allow for empirical analysis of solution methods. We develop three exact formulations for this problem, but using these formulations and the general purpose mixed integer programming solver CPLEX we are unable to solve moderately-sized problem instances. Therefore, the focus of this thesis lies on developing a route construction and route improvement local search heuristic for the FDPTW.

Keywords. *FDPTW, vehicle routing, local search heuristic*

[*]Studentnumber: 342750

Contents

1 Introduction

1.1 Problem description

Over the past decade, online shopping has become increasingly popular. Even during times of economic recession, online retail sales have continued to grow. For example, US web sales rose with over 15.7% from every quarter of 2012 to the same quarter in 2013 (U.S. Department of Commerce, 2013), while the same market expanded with 9% over the same period in the Netherlands (Thuiswinkel Marktmonitor, 2013). Also, the total markets value rose from 119 billion in 2008 to 170 billion in 2010, an increase of 30%.

When products are ordered online there are various ways of receiving the parcel. The most common way is to have the parcel delivered at home. However, the parcel might be too large to fit in the mailbox and it is often the case that people are not at home during the day, since they are at work or they are at school. Therefore, it often happens that a courier cannot deliver his package because there is no one at home to receive the parcel. This issue can be overcome by allowing customers to choose a certain time window in which they would like to have the parcel delivered. However, if the offered time windows are very wide, this is inconvenient for the customer, since the customer potentially has to stay at home for the entire duration of the time window. If the offered time windows are tight it is difficult or expensive for the supplier to arrive in time. Also, this solution may lead to a lot of deliveries scheduled at certain times during the day since a lot of people might like to have the parcel delivered either before they leave for, or after they return from work.

An alternative that is now offered by a lot of companies is to have the goods delivered at pickup points. These points are often conveniently located, for example next to a supermarket, and they serve as a place where suppliers can deliver the parcels, so that the customers can pick them up at that location at a later time.

It might be advantageous for both the supplier and the customer to give the customer the opportunity to provide the company with the various locations where he or she will spend the day and the corresponding time frames. For example, a customer might provide the supplier with the information that he will be at home until 9:00 and at work from 9:00 to 18:00. He then passes a pickup point at 18:30 while doing groceries, and he arrives home at 19:00. Since the company knows exactly where the customer is at various times of the day, the parcel can be delivered at any of the locations, given that the delivery takes place in the corresponding time window. For the customer, this is beneficial since he does not have to stay home for a long period of time. It might also be beneficial for the company, since this provides the company with some flexibility. The company can select the most convenient location and time window to deliver the parcel, which can lead to reduced costs.

For the company, the problem then boils down to minimizing the costs by delivering the packages at one of their delivery locations. These locations are geographically dispersed and the travel times and travel costs between each of these locations are given. For this purpose, the company has a fleet of vehicles at its disposal which are all stored in a single depot. All vehicles can leave the depot after a given time, and they all have to be back in the depot before another given time. Every package has to be delivered once, and each package also has an associated service time, which consists of the time that it takes to finish the delivery after the courier has arrived at the customer. Furthermore, each delivery has to start and end within the time window for the location at which it is delivered. A vehicle cannot arrive too late to complete the serivce, but it can arrive too early. In this case, the courier waits until the opening of the time window to start completing the service time. Each vehicle also has a capacity, and each package has a weight. A vehicle can not carry more packages than its capacity allows.

1.2 Existing related literature

To the best of our knowledge, this is a problem which has not yet been researched, however a lot of research has already been done into similar problems. The vehicle Routing Problem (VRP), first proposed by Dantzig and Ramser (1959), is a generalization of the Traveling Salesman Problem (TSP). The TSP deals with finding a lowest cost route given a depot and various spatially scattered locations that have to be visited. The VRP can be described as the problem of assigning routes that start and end in a depot to a fleet of vehicles, that together visit all of the locations. The goal of the VRP is to minimize the costs. Baldacci et al. (2012) mention that recently the most sucessful exact methods for solving the CVRP are based on the two-index flow formulation, the two-commodity flow formulation (Baldacci et al., 2004) and the set partitioning formulation (Balinski and Quandt, 1964). The VRP knows many extensions, each dealing with complications that arise when considering the application of vehicle routing models in practice. For example, certain goods may only be transported by certain vehicles, or vehicles may be stored at a variety of depots. The most well-known extension to the VRP is the Capacitated Vehicle Routing Problem (CVRP), where each vehicle has a carrying capacity which may not be exceeded.

Another well-known member of the set of Vehicle Routing Problems, which is in its turn an extension of the CVRP, is the Vehicle Routing Problem with Time Windows (VRPTW). Next to the capacity constraint, in the VRPTW every location has to be visited within a certain time window, which can be either a soft or a hard time window. Soft time windows allow vehicles to arrive after the latest time that the service should have started by incurring a penalty. In the case of hard time windows, vehicles are under no circumstances allowed to arrive too late, however they can arrive too early. In the latter case, the vehicle waits until it can start service. The focus in this thesis will be on hard time windows. Baldacci et al. (2004) state that the most promising exact solution methods for the VRPTW are based on the set partitioning formulation. Bräysy and Gendreau on the other hand find promising results in classical route construction and improvement methods (Bräysy and Gendreau, 2005a) and metaheuristics (Bräysy and Gendreau, 2005b), and they expect research in these two fields to intensify.

The Multi-Period Vehicle Routing Problem (MPVRP) is another extension of the VRP. In this problem, the time windows consist of entire days, and customers have to be visited on a specified number of days within the planning horizon. This problem bears some resemblance to our problem, since it also provides various options on how to service a customer. However, in the MPVRP, customers have to be serviced multiple times, while in our problem a customer only has to be serviced once. Also the multi-period element of the MPVRP makes the problem structure fundamentally different from our problem.

1.3 Approach for this thesis

In the VRPTW, one task may be considered as visiting a location in the corresponding time window. In our problem, one task is the delivery of a parcel. For each parcel, a set of locations with corresponding time windows is given. The task can be completed by visiting any of the locations during its corresponding time window. We shall call this problem the Flexible Delivery Problem with Time Windows (FDPTW).

The problem considered in this thesis clearly is a generalization of the VRPTW, which is known to be NP-hard. Consider the case in which every package to be delivered has exactly one potential location with a corresponding time window. This problem reduces to a VRPTW, which shows that the FDPTW is NP-hard.

In this thesis, we first propose several MIP formulations for the FDPTW in Section 2. The first proposed formulation is a vehicle flow formulation which is strongly based on the vehicle flow formulation for the VRPTW which can for example be found in Fisher et al. (1997). In this formulation, every vehicle is assigned its own set of variables for keeping track of which locations the vehicle visits on its route. The second formulation is based on a formulation that can be found in Baldacci et al. (2004), who based their multicommodity flow formulation for the VRPTW on earlier work by Garvin et al. (1957), Gavish and Graves (1979) and Gavish and Graves (1982). This formulation no longer requires each vehicle to have its own set of variables, but this comes at the cost of extra variables and restrictions for modeling the capacity constraint. The third formulation is also based on the work of Baldacci et al. (2004), who developed a two-commodity flow formulation for the CVRP. They in turn based their formulation on the work of Finke et al. (1984), who created a two-commodity network flow formulation for the TSP. We compare the performance of these MIP formulations with one another by solving several benchmark instances by implementing the formulations in the general purpose mixed integer programming solver CPLEX.

Since it proves to be computationally infeasible to solve large problem instances of the FDPTW with exact solution methods, the focus of this thesis lies on developing a heuristic in Section 3. Because the FDPTW bears similarity with the VRPTW, we will base stages of our heuristic on a heuristic that has good performance on solving VRPTW instances. Since Bräysy and Gendreau (2005a) state that traditional route construction and route improvement approaches not only provide good solutions with a low computational effort, but also that they are a major component of all metaheuristics for the VRPTW, we will develop such a local search heuristic. We will base stages of our heuristic on the heuristic of Bräysy (2002), since that heuristic proved to perform very well on solving VRPTW instances in comparison to other route construction and improvement heuristics (Bräysy and Gendreau, 2005a).

For the comparison of the performances of the algorithms and MIP formulations we create a set of benchmark instances in Section 4. Since the benchmark instances of Solomon (1987) are widely used within the VRPTW literature, we analyze the procedure used in creating these instances and base the FDPTW benchmark instances on those. In order to obtain an indication of the size of the potential savings that can be obtained by allowing customers to specify multiple delivery locations for their parcels, we shall create another set of instances of which the design is based on the topography of a city. In Section 5, we use the FDPTW benchmark instances to tune the parameter values of the heuristic. We shall apply our heuristic and formulations on the created benchmark instances to draw conclusions on their performance in Section 6.

There are no standards concerning the criteria on which a heuristic should be judged in algorithmic research (Barr et al., 1992). In our opinion, the following set of properties constitute a set which accurately measures the quality of a heuristic: *Quality of the solutions, runtime, simplicity, flexibility and robustness* (Barr et al. (1992), Cordeau et al. (2002)). We will judge our heuristic on this criteria and make our concluding remarks in Section 7.

2 Formulations

In this section, three formulations of the FDPTW are presented. The performance of these formulations is compared by solving a set of benchmark instances in section 6.1.

2.1 Definitions and notation

For the modeling of the problem, we introduce the concept of nodes. A node is a location where a package can be delivered within an associated time frame. Every package is assigned one or more nodes, corresponding to the locations where this package can be delivered. If a node is visited on a route, it implies that the corresponding package is delivered there. Therefore, exactly one of the nodes associated with a package should be visited by a route in a feasible solution, since the package needs to be delivered and it cannot be delivered more than once.

In our models of the FDPTW, every package p has its own set of unique nodes at which it can be delivered. This means that if two separate packages p and q can be delivered at the same office building, this building will be considered as two separate nodes, one for p and one for q. The travel time between these two nodes is 0. Throughout this thesis the following sets will be used:

V	Set of vehicles, each with capacity C
P	Set of packages to be delivered
L_0	Set of all nodes
L	Set all nodes except the depot, $L_0 \setminus \{0\}$,
L^p	Subset of L which includes the nodes where package p can be delivered
Z	Set which contains all sets of nodes which share the same geographical location

Furthermore, we make use of the following parameters:

q_i	The size of the package to be delivered at node $i \in L$
s_i	Individual service time at node $i \in L$
(e_i, l_i)	The time window in which the package can be delivered at node $i \in L$
c_{ij}, t_{ij}	The costs and the time it takes to drive from node $i \in L_0$ to $j \in L_0$, respectively. The fixed costs of using a vehicle are included in $c_{0i}, \quad \forall i \in L$
L_0	Subset of L which contains the nodes where package p can be delivered.
K	The total number of vehicles available ($\lvert V \rvert = K$)

e_0	the earliest time that vehicles are allowed to leave the depot
l_0	the time before which all vehicles should have returned to the depot

To simplify some of the expressions in the formulations, we define a graph $G = (L_0, E)$. Here, L_0 is the set of vertices and E is the set of edges, $E = \{(i,j), i,j \in L_0, i \neq j\}$. The travel times and costs are assumed to be symmetric in this thesis. Note that the use of nodes in our models also allows two packages to be delivered simultaneously at the same location, say an office building. For example, for package p this is node i, and for package q this is node j. Then by setting $t_{ij} = 0$ and $c_{ij} = 0$, those packages can be delivered together, although the model assumes them to be delivered consecutively. However, when multiple packages are delivered at the same location, it usually makes little sense to account for all the individual service times. Most of the service time is likely to consist of operations such as parking the car and walking to the right building, which are operations that do not have to be repeated when delivering multiple packages. Therefore, we propose to divide each service time into two parts; the shared service time (r_i) and the individual service time (s_i). The shared service time is that part of the delivery which does not have to be carried out twice if two packages are delivered together, while the individual service time does have to be carried out for both packages. In our model, the shared service time is included in the travel time t_{ij} if i and j are not the same geographical location. We also assume that the shared service time can be completed before the time interval of the location opens.

2.2 Formulation I

The formulation proposed as follows is strongly based on a vehicle flow formulation of the VRPTW which can be found in Fisher et al. (1997). This formulation makes use of the following variables:

x_{ij}^v	A binary variable which takes value 1 if node $j \in L$ is visited after node $i \in L$ is visited
t_i	the time at which the delivery at node i starts.

The formulation then looks as follows:

$$\min \sum_{(i,j)\in E} \sum_{v \in V} c_{ij} x_{ij}^v \tag{1}$$

$$\sum_{\{(i,j): i \in L_0, j \in L^p, i \neq j\}} \sum_{v \in V} x_{ij}^v = 1, \quad \forall p \in P \tag{2}$$

$$\sum_{\{(i,j): i \in L^p, j \in L_0, i \neq j\}} \sum_{v \in V} x_{ij}^v = 1, \quad \forall p \in P \tag{3}$$

$$\sum_{j \in L} x_{0j}^v \leq 1, \quad \forall v \in V \tag{4}$$

$$\sum_{i \in L} x_{i0}^v \leq 1, \quad \forall v \in V \tag{5}$$

$$\sum_{(i,u)\in E} x_{iu}^v - \sum_{(u,j)\in E} x_{uj}^v = 0, \quad \forall v \in V, u \in L_0 \tag{6}$$

$$\sum_{\{(i,j):i\in L_0, j\in L, i\neq j\}} x_{ij}^v q_j \leq C, \quad \forall v \in V \tag{7}$$

$$\sum_{\{(i,j):i,j\in F', i\neq j\}} x_{ij}^v \leq |F'| - 1, \quad \forall F' \subseteq F, |F'| \geq 2, F \in Z, v \in V \tag{8}$$

$$t_i + s_i + t_{ij} - (1 - x_{ij}^v)M \leq t_j, \quad \forall i,j \in L, i \neq j, v \in V \tag{9}$$

$$e_0 + t_{0j} - (1 - x_{0j}^v)M \leq t_j, \quad \forall j \in L, v \in V \tag{10}$$

$$t_i + s_i + t_{i0} - (1 - x_{i0}^v)M \leq l_0, \quad \forall i \in L, v \in V \tag{11}$$

$$e_i \leq t_i \leq l_i - s_i, \quad \forall i \in L \tag{12}$$

$$t_i \geq 0, \quad \forall i \in L_0 \tag{13}$$

$$x_{ij}^v \in \{0,1\}, \quad \forall i,j \in L_0, i \neq j, v \in V \tag{14}$$

The objective function (1) aims to minimize the costs corresponding to driving between the various nodes. Equations (2) and (3) make sure that every package is delivered exactly once at one of its possible nodes, before and after another package has been delivered at its possible nodes, respectively. Inequalities (4) and (5) ascertain that a vehicle can deliver at most one package at one node directly after leaving and directly before entering the depot. Equation (6) models the flow conservation constraints, ensuring that a vehicle can only leave a node if it has also arrived there, and vice versa. Inequalities (7) ensure that the total weight of the packages each vehicle carries does not exceed its capacity. Constraints (8) and (9) together eliminate all subtours. When packages are not allowed to be delivered at the same time by one vehicle, constraints (9) suffice as subtour elimination constraints. However, in our case it is now possible that $s_i = 0$, $t_{ij} = 0, t_i = t_j$, and $x_{ij}^v = x_{ji}^v = 1$. Therefore, we introduce inequalities (8), which ensure that there are no subtours in any set of nodes that are the same geographical location. Inequalities (9), (10) and (11) cause the delivery times and the times of departure and arrival at the depot to be consistent and feasible. Here, M should be a sufficiently large number, such that the constraint becomes inactive when $x_{ij}^v = 0$. Inequalities (12) ensure that each package is delivered within the time interval available at that node. Constraints (13) and (14) are straightforward, stating that the delivery times have to be positive, and that the x_{ij}^v variables have to be binary.

2.3 Formulation II

In Formulation I, the variables x_{ij}^v keep track of which vehicle uses which arcs. This might seem redundant, since the variables x_{ij} without the additional index for the vehicles could also suffice for this purpose. The number of vehicles used is simply the number of vehicles leaving the depot ($\sum_{j\in L} x_{0j}$) and their corresponding routes can also easily be found. Most of Formulation I can easily be modified to use the variables x_{ij} instead of x_{ij}^v, but this does not hold for the capacity constraint. Two methods for restricting the capacity of the vehicles in the CVRP are found in Baldacci et al. (2004). The first method they describe is a multicommodity flow formulation based on earlier work by Garvin et al. (1957), Gavish and Graves (1979) and Gavish and Graves (1982). This method requires the introduction of the variables y_{ij}^l, specifying the amount of demand flowing from location i to j that is destined for delivery at location l. The adjusted formulation is as follows:

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{15}$$

$$\sum_{\{(i,j):i\in L_0, j\in L^p, i\neq j\}} x_{ij} = 1, \quad \forall p \in P \tag{16}$$

$$\sum_{\{(i,j):i\in L^p, j\in L_0, i\neq j\}} x_{ij} = 1, \quad \forall p \in P \tag{17}$$

$$\sum_{j\in L} x_{0j} \leq K \tag{18}$$

$$\sum_{i\in L} x_{i0} \leq K \tag{19}$$

$$\sum_{(i,u)\in E} x_{iu} - \sum_{(u,j)\in E} x_{uj} = 0, \quad \forall u \in L_0 \tag{20}$$

$$\sum_{(i,u)\in E} y_{iu}^l - \sum_{(u,j)\in E} y_{uj}^l = \begin{cases} q_l \sum_{(i,l)\in E} x_{il}, & \text{if } \ u = l \\ 0, & \text{if } \ u \neq l \\ -q_l \sum_{(i,l)\in E} x_{il}, & \text{if } \ u = 0 \end{cases} \quad \forall u \in L_0, l \in L \tag{21}$$

$$y_{ij}^l \leq q_l x_{ij}, \quad \forall (i,j) \in E, l \in L \tag{22}$$

$$\sum_{j\in L}\sum_{l\in L} y_{ij}^l \leq C - q_i, \quad \forall i \in L_0 \tag{23}$$

$$\sum_{\{(i,j):i,j\in F', i\neq j\}} x_{ij} \leq |F'| - 1, \quad \forall F' \subseteq F, |F'| \geq 2, F \in Z \tag{24}$$

$$t_i + s_i + t_{ij} - (1 - x_{ij})M \leq t_j, \quad \forall i,j \in L, i \neq j \tag{25}$$

$$e_0 + t_{0j} - (1 - x_{0j})M \leq t_j, \quad \forall j \in L \tag{26}$$

$$t_i + s_i + t_{i0} - (1 - x_{i0})M \leq l_0, \quad \forall i \in L \tag{27}$$

$$e_i \leq t_i \leq l_i - s_i, \quad \forall i \in L_0 \tag{28}$$

$$t_i \geq 0, \quad \forall i \in L_0 \tag{29}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in L_0, i \neq j \tag{30}$$

$$y^l_{ij} \geq 0, \quad \forall i, j \in L_0, i \neq j, l \in L \tag{31}$$

Constraints (15) to (20) and (24) to (30) are very similar to the restrictions in Formulation I and require no further explanation. Constraints (21)-(23) together ensure that the capacity of the vehicles is not exceeded. Equalities (21) ensure that the y^l_{ij} variables take on the right values. When $u = l$ and a package is delivered at location l, the incoming flow destined for l should be q_l higher than the outgoing flow destined for l, since the package is delivered in between. When $u \neq l$, the amount of flow destined for l should remain equal after visiting location u, since the package for location l is still carried by the vehicle. When u is the depot, the incoming flow destined for l should be lower by an amount of q_l than the outgoing flow destined for l, since the package should have been delivered while driving the route. Note that out version of these restrictions is slightly different than those of Baldacci et al. (2004), since we have to account for the fact that there is not always a package delivered at a location by adding the summation over the x_{il} variables. Inequalities (22) restrict the flow destined for l over arc (i, j) to zero whenever arc (i, j) is not used in the solution, or to q_l when it is used. Furthermore, the inequalities (23) restrict the outgoing flow of each location i to $C - q_i$, the total capacity minus the weight of the package to be delivered at i.

2.4 Formulation III

Baldacci et al. (2004) developed a two-commodity flow formulation for the CVRP, which they in turn based on work of Finke et al. (1984), who created a two-commodity network flow formulation for the TSP. From the LP-relaxation of this formulation, Baldacci et al. were able to derive lower bounds which could compete with the best lower bounds developed up until then. Also, they were able to solve a CVRP instance with 135 customers by using a branch-and-cut algorithm based on this formulation, which was the largest solved CVRP instance to date. The idea of Finke et al.'s formulation is that there are two commodities; commodity X, which has to be delivered at the customers, and commodity Y, which has to be picked up at the customers. At the beginning of this route, a vehicle carries at most C of commodity X, and C minus this quantity of Y, while at the end of the route, the vehicle carries 0 of X, and C of Y. During its trip, the vehicle always has a combined load of exactly C. In modeling vehicle routing problems, commodity X can be interpreted as the total load of the vehicle still to be delivered, while commodity Y can be interpreted as the total spare capacity on the vehicle.

Baldacci et al. (2004) came up with a formulation that uses variables x_{ij} to indicate whether arc (i, j) is used in the solution, with no regard for the direction the vehicle is traveling in. They use variables z_{ij} to indicate the flow (i.e. the weight of the cargo on the vehicle) on arc (i, j), and z_{ji} to indicate the spare capacity (i.e. $C - z_{ij}$) on this same arc. This way of formulating the problem allows them to only define the x_{ij} variables for $i < j$ under the assumption of symmetric travel times and costs, thus greatly reducing the total number of variables by halving the number of x_{ij} variables. The exact routes, and in particular the order of visiting the locations of the routes, can be extracted from the variables after solving the model. However, this approach does not suffice in formulating the FDPTW, since the order of visiting the locations needs to be explicitly incorporated in the formulation for the modeling of time window-related constraints.

The two-commodity flow formulation of the FDPTW formulation requires us to alter some of the used sets and parameters, and it requires the use of new variables. Firstly, the depot is split into two separate nodes, a start depot and an end depot, denoted 0 and $n + 1$. Each route should start at 0 and end in $n + 1$. It holds that $c_{0i} = c_{i,n+1}$ and $t_{0i} = t_{i,n+1}, \quad \forall i \in L$, and $e_0 = e_{n+1}$ and $l_0 = l_{n+1}$. Secondly, for efficiency we use L_s to refer to the set $L \cup \{0, n+1\}$, L_0 to refer to $L \cup \{0\}$ and L_{n+1} to refer to $L \cup \{n + 1\}$. Furthermore, $q(P)$ is used to denote the cumulative value of all packages to be delivered. Note that this is not simply $\sum_{i \in L} q_i$, since packages can have multiple delivery locations and each package should only be counted once.

For this formulation, we make use of the modified graph $\bar{G} = (L_s, \bar{E})$. Here, L_s is the set of vertices and $\bar{E}$ is the set of edges, $\bar{E} = \{(0, i), i \in L\} \cup \{(i, j), i, j \in L, i \neq j\} \cup \{(i, n+1), i \in L\}$. The formulation makes use of the following variables:

z_{ij}	The size of the flow on the arc from node i to j
x_{ij}	A binary variable which takes value 1 if a vehicle visits node j directly after it visits node i, and 0 otherwise.
t_i	the time at which the delivery at node i starts.

If $x_{ij} = 1$, the variable z_{ij} denotes the weight of cargo on the truck between nodes i and j, while z_{ji} denotes the spare capacity $(C - z_{ij})$. The two-commodity flow formulation of the FDPTW is as follows:

$$\min \sum_{(i,j) \in E} c_{ij} x_{ij} \tag{32}$$

$$\sum_{\{(i,j): i \in L_0, j \in L^p, i \neq j\}} x_{ij} = 1, \quad \forall p \in P \tag{33}$$

$$\sum_{\{(i,j): i \in L^p, j \in L_{n+1}, i \neq j\}} x_{ij} = 1, \quad \forall p \in P \tag{34}$$

$$\sum_{(i,u) \in E} x_{iu} - \sum_{(u,j) \in E} x_{uj} = 0, \quad \forall u \in L \tag{35}$$

$$\sum_{(i,u) \in E} z_{iu} - \sum_{(u,j) \in E} z_{uj} = 2q_u \left(\sum_{(i,u) \in E} x_{iu} \right), \quad \forall u \in L \tag{36}$$

$$\sum_{j \in L} z_{0j} = q(P) \tag{37}$$

$$\sum_{i \in L} z_{i0} \leq KC - q(P) \tag{38}$$

$$\sum_{j \in L} z_{n+1,j} \leq KC \tag{39}$$

$$z_{ij} + z_{ji} = \begin{cases} Cx_{0j}, & \forall j \in L \\ C(x_{ij} + x_{ji}), & \forall i, j \in L, i \neq j \\ Cx_{i,n+1}, & \forall i \in L \end{cases} \tag{40}$$

$$t_i + s_i + t_{ij} - (1 - x_{ij})M \leq t_j, \quad \forall i, j \in L, i \neq j \tag{41}$$

$$e_0 + t_{0j} - (1 - x_{0j})M \leq t_j, \quad \forall j \in L \tag{42}$$

$$t_i + s_i + t_{i,n+1} - (1 - x_{i,n+1})M \leq l_{n+1}, \quad \forall i \in L \tag{43}$$

$$e_i \leq t_i \leq l_i - s_i, \quad \forall i \in L \tag{44}$$

$$t_i \geq 0, \quad \forall i \in L \tag{45}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in L_0, j \in L_{n+1}, i \neq j \tag{46}$$

$$z_{ij} \geq 0, \quad \forall i, j \in L_s, i \neq j \tag{47}$$

The objective function (32) aims to minimize the costs, while equalities (33) and (34) ensure that every package is delivered exactly once. The equalities (35) are the flow conservation constraints. These constraint were not necessary in Baldacci's formulation of the CVRP, since it had an additional constraint stating that every location should have two arcs incident to it, which is not the case in the FDPTW. Since we dot not use that restriction, the flow conservation constraint is needed in our formulation for the VRPTW. Equalities (36) ensure that the incoming flow is exactly twice the size of the outgoing flow at a node if a package is delivered there. Constraints (37) to (39) ensure that the flow variables z_{ij} take the right values when they are incident to the depot. When a package is delivered at j directly after visiting i, equalities (40) force the sum of the flow variables between these nodes to be equal to the capacity. Constraints (41), (42) and (43) cause the delivery times and the times of departure and arrival at the depot to be consistent and feasible. Here, M should be a sufficiently large number, such that the constraint becomes inactive when $x_{ij} = 0$. Constraints (44) ensure that each package is delivered within the time interval available at that node. Constraints (45), (46) and (47) are straightforward, stating that the delivery times and the z_{ij} variables have to be positive, and that the x_{ij} variables have to be binary.

3 The local search heuristic

In this section, we develop a local search algorithm for the FDPTW. Since the FDPTW shares similarities with the VRPTW, we shall base several stages of our heuristic on the heuristic proposed by Bräysy (2002), since that heuristic showed to perform well in comparison to other route construction and improvement heuristics (Bräysy and Gendreau, 2005a). The local search heuristic for the VRPTW proposed by Bräysy (2002), which aims to minimize the total number of vehicles in the solution, consists of three stages. In the first stage, initial routes are created to obtain various feasible solutions. The parameter *solutions* is used to define the total number of initial solutions to be made. In the second stage, an ejection chain procedure is used on a subset of these initial solutions to decrease the total number of routes in each solution. The size of this subset is defined by the parameter x, such that the heuristic only continues with the $x\%$ solutions with the lowest costs. Finally, the total costs of the solutions in this subset are minimized by reordering the sequence in which customers are visited on a route using an Or-opt neighborhood.

Next to the adaptation of these steps for solving the FDPTW, we shall make some other important changes and introduce a new route improvement procedure. Most importantly, our objective is to minimize the total costs of the solution, and not to minimize the total number of vehicles. Bräysy only minimizes the costs after the route elimination procedure by changing the sequence of the nodes within the routes. However, if solutions could improve by also relocating nodes between routes it would be wise to do so. For this purpose we will develop a route improvement procedure, which will aim to minimize the total costs by relocating and replacing nodes in a given solution.

Due to the conflict in objectives between our algorithm and that of Bräysy, it is difficult to tell whether the use of the ejection chain procedure would be beneficial for our solutions. On the one hand, one would expect the best solutions with respect to the total costs to be solutions with a low number of vehicles. Also, when our route improvement procedure proves to be very time consuming, first applying route elimination might reduce the total running time. On the other hand, the ejection chain procedure might eliminate too many routes, or it might eliminate routes in a way that does not benefit the total costs of the solution. Because of these reasons, we will decide on the use of a route elimination stage by running various tests. Also, the route elimination procedure might prove to be computationally too complex for solving FDPTW instances.

3.1 Forward time slack

In order to efficiently implement our heuristic, we maintain a set of variables for each route. Firstly, we denote a path as $(0, r_1, r_2, \ldots, r_d, n+1)$, where both 0 and $n+1$ are the depot, and r_d is the last node visited before returning to the depot. We use D_{r_i} and W_{r_i} to denote the departure and the waiting time for each node r_i on the route respectively. Along with these variables, we also keep track of *the forward time slack* $F_{r_i}^{(r_i,\ldots,n+1)}$ for each node, which is introduced in Savelsbergh (1992) to construct an efficient search strategy for Or-exchanges, which we shall introduce in Section 3.3. The forward slack $F_{r_i}^{(r_i,\ldots,r_j)}$ is defined as the time that the departure at location r_i, D_{r_i}, can be moved forward in time without making the route infeasible. Since Savelsbergh does not take service time into consideration explicitly, we give the mathematical expression of the forward time slack below to match our notation. The mathematical definition of the forward time slack is as follows:

$$F_{r_i}^{(r_i,\ldots,r_j)} = \min_{i\leq k\leq j}\{l_{r_k} - (D_{r_i} + \sum_{i\leq p<k} t_{r_p,r_{p+1}} + \sum_{i<p\leq k} s_{r_p})\} \tag{48}$$

Note that the waiting time along the path is not included in the equation Also, $D_{r_i} + \sum_{i\leq p<k} t_{r_p,r_{p+1}} + \sum_{i<p\leq k} s_{r_p}$ can be seen to represent the earliest possible departure at r_k, which occurs only in the case that there is no waiting time on $(r_i,\ldots,r_k)$. This may seem counter-intuitive at first glance, but it becomes clear when taking a closer look at the derivation of the equation. Intuitively, one might say that instead of the earliest departure time at r_k, the actual departure time should be used in the equation. However, one should also take into account the waiting time on the path $(r_i,\ldots,r_k)$. Any waiting time that occurs on the path $(r_i,\ldots,r_k)$ can be seen as a 'buffer' for an increase in the departure time at r_i. As long as the increase in departure time at r_i stays below the total waiting time on $(r_i,\ldots,r_k)$, the departure time at r_k remains the same. Note that feasibility of the arrival times at the nodes between r_i and r_k does not have to be taken into consideration when calculating the k^{th} term of the minimization, since $F_{r_i}^{(r_i,\ldots,r_j)}$ is chosen to be the minimum of those terms. The adjusted formula for the forward time slack would look as follows:

$$\begin{aligned} F_{r_a}^{(r_i,\ldots,r_j)} = \min_{i\leq k\leq j}\{&l_{r_k} - (\text{actual departure time at } r_k) \\ &+ \text{total waiting time on } (r_j,\ldots,r_k)\} \end{aligned} \tag{49}$$

Let us now introduce the notation W_p, which is the waiting time at node p, and rewrite the equation in mathematical notation:

$$F_{r_i}^{(r_i,\ldots,r_j)} = \min_{i\leq k\leq j}\{l_{r_k} - (D_{r_i} + \sum_{i\leq p<k} t_{r_p,r_{p+1}} + \sum_{i<p\leq k} s_{r_p} + \sum_{i<p\leq k} W_p) + \sum_{i<p\leq k} W_p\} \tag{50}$$

It is easy to see that this reduces to Equation (48).

3.2 Creating initial solutions

The basic approach in the creation of initial solutions is sequentially adding nodes to a route until this is no longer possible, and then initializing a new route if there are still undelivered parcels. Bräysy (2002) refers to this method as the Hybrid Construction Heuristic. For the VRPTW, this heuristic is initialized by selecting a set of seed nodes from the set of all locations to be visited. These seeds are chosen in a way that the distances to the depot and the distances between these locations are very high, and they are then used to initialize new routes with. The reason behind this is that it is preferable to have these nodes routed at an early stage of the construction heuristic, since we risk having to create a separate route for any of these locations when a lot of locations are already routed in a late stage of the heuristic.

For the FDPTW however, this argument is not directly applicable, since we may not have to visit all the locations that are located far from the depot, since some of the packages may also be delivered at a location that is closer to the depot. Therefore, for each package only the delivery location closest to the depot is considered as a potential seed node. The set of all these nodes together will be denoted as N. The seed nodes are then selected from this set of nodes as follows: The first seed node is simply the node in N that is furthest from the depot. Then, three nodes are sequentially added by each time selecting the node from N that maximizes the distance to the depot and the previously selected seed nodes. These nodes are called the primary nodes (PN). In mathematical notation, one can find the next node to add to PN using the following equation:

$$\text{next node to be added to PN } \in \arg\max_{i \in N/PN} \left\{ \sum_{k \in PN \cup 0} t_{k,i} \right\} \tag{51}$$

The secondary nodes (SN) are then obtained by finding the nodes in N that are closest to the midpoints of each of the sides of the convex hull of the primary nodes, and that are not in PN. We use a convex hull where Bräysy uses a quadrangle, since four points do not uniquely define a quadrangle, but their convex hull is unique. The convex hull is found using Jarvis' algorithm (Jarvis, 1973). The basic idea of this algorithm is to start the convex hull with the leftmost node, which we will call p_0, and then continue by sequentially adding nodes in counter-clockwise orientation. The next node to be added after p_i, p_{i+1}, is the node such that all other nodes in PN lie to the left of the directed line (p_i, p_{i+1}), where the perspective used is as if one is standing at p_i, looking in the direction of p_{i+1}. This in turn is the node for which the path (p_i, p_{i+1}, p_j) makes a left turn for every other node $p_j \in PN/\{p_i, p_{i+1}\}$. The path (p, q, r) makes a left turn if $(q_y - p_y)(r_x - q_x) - (q_x - p_x)(r_y - q_y) < 0$, where the subscript is used to denote the x- or y-coordinate of a node.

Finally, a set of $\bar{n}$ tertiary nodes (TN) with the largest distance to the depot that are not yet in PN or SN is selected from N. Then, the set of seed nodes is $S = PN \cup SN \cup TN$. An example of the selection of the primary, secondary and tertiary nodes can be found in Figure 1. Note that while the nodes for packages 14 and 15 at the top right are far away from the depot, they are not selected as seed nodes. This is because those packages can also be delivered at locations closer to the depot, and they were therefore not included in the set N.

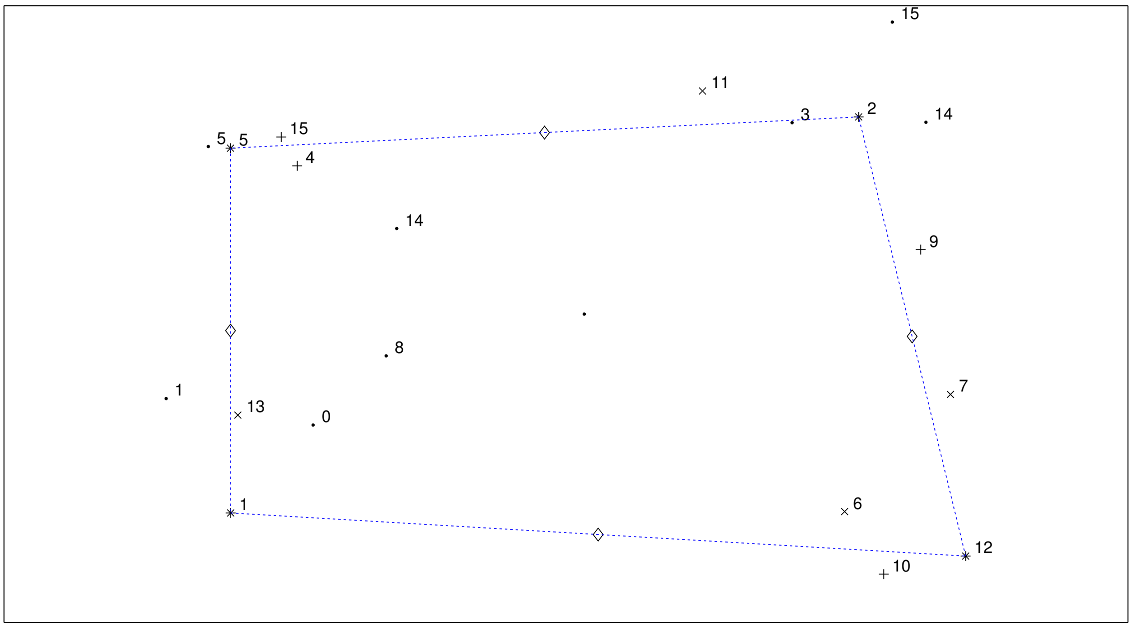

Figure 1: An example of the selection of the primary, secondary and four tertiary nodes for an instance with 16 packages to be delivered. The numbers right above the nodes denote the number of the package to be delivered at that node. The asterisks mark primary nodes, the small crosses mark secondary nodes, and the plus signs mark tertiary nodes.

When the set S is chosen, the creation of routes is started. The first route is initialized by selecting one of the seed nodes, and then creating a route which leaves from the depot for this node and returns immediately afterwards. In the route construction stage of the heuristic, every seed node is used once to initialize the first route. If the number of initial solutions to be made is larger than the total number of seed nodes, the remaining solutions are initialized by selecting one of the nodes in S at random. Once the first route is initialized, we no longer restrict ourselves to the nodes in N. All nodes for which the parcel is not yet delivered within a distance $\bar{d}$ of the nodes currently on the route are then considered for insertion. The following function is used to evaluate the insertion costs of a node u between two nodes i and j on a route:

$$C_u = \alpha_1 \times \bar{C_u} + \alpha_2 \times W_u - \alpha_3 \times t_{0u} \tag{52}$$

Here, $\bar{C_u}$ is equal to the old driving costs minus the new driving costs, which is equal to $c_{iu} + c_{uj} - c_{ij}$, and W_u is equal to the total waiting time after the insertion minus the total waiting time on the route before the insertion. α_1, α_2 and α_3 are user defined weight parameters.

For each node within the specified distance threshold the feasible insertion places on the route are determined and the corresponding insertion costs are calculated. An insertion of node u between nodes i and j on a route is feasible if it meets the following three conditions:

- Adding this node to the route does not exceed the capacity constraint of the vehicle on this route
- The vehicle arrives in time for node u: $D_i + t_{iu} + s_u \leq l_u$.
- The vehicle also arrives in time at node j such that the route remains feasible: $\max\{\max\{D_i + t_{iu}, e_u\} + s_u + t_{uj}, e_j\} + s_j \leq D_j + F_j$. Here, $\max\{D_i + t_{iu}, l_u\} + s_u$ can be seen to represent the departure time at u, such that the entire left hand side is equal to the departure time at j after the insertion.

The minimum cost of all feasible insertions is found, and the node is then inserted at the corresponding location on the route. If there are no nodes left within the threshold distance that can be feasibly inserted on the route, a new route is initialized. Here, we allow for some randomness to create distinct initial solutions: All nodes for which the parcel is delivered are removed from S, and the next seed node is randomly selected from the remaining set. Again, a route is initialized by only visiting this node and the previous procedure is repeated. This is done until all parcels are delivered. Every time a route is finished, and after each $\bar{k}$ insertions on a route, it is optimized with the Or-opt operator, which will be explained in the following subsection.

We introduce another important alteration in the procedure with respect to Bräysy (2003). Since our objective is to minimize the total costs rather than to minimize the total number of vehicles in the solution, we introduce a new parameter $\bar{c}$. This parameter restricts the insertion of nodes: A node can only be inserted in a route if the corresponding increase in travel costs is below $\bar{c}$. Note that we do not use the cost evaluation function here, but simply the increase in travel costs, $c_{iu}+c_{uj}-c_{ij}$. If an insertion has the minimum value in the cost evaluation function, but the increase in travel costs are higher than the threshold $\bar{c}$, the node is not inserted and it is no longer considered for insertion on this route. An example of a situation where this is profitable can be found in Figure 2. We assume that the nodes 1 to 6 have disjunct increasing time intervals, such that node i can only be visited before node j on a route if $i < j$. Furthermore, we assume the planning horizon to be long, such that one vehicle is able to deliver all the packages. In the figure, 1 is chosen as seed node, and (a) is the last situation during the route building procedure where there is no difference between our routes and those of Bräysy. Independent of whether nodes 2 and 4 are within the distance threshold $\bar{d}$, the route is likely to become $(1, 3, 5, 6)$ if α_1 is sufficiently large. At this stage of the route building procedure, node 4 will be the next node to be considered for insertion. This will even be the case for very low values of $\bar{d}$, since node 4 is in very near proximity of node 6. However, due to the time windows, node 4 has to be placed between 3 and 5, which leads to a very large increase in the total driving distance. The resulting situation of continuing with Bräysys procedure is shown in (b). The result of using our modified procedure with the parameter $\bar{c}$ is shown in (c), which is clearly the more cost efficient one.

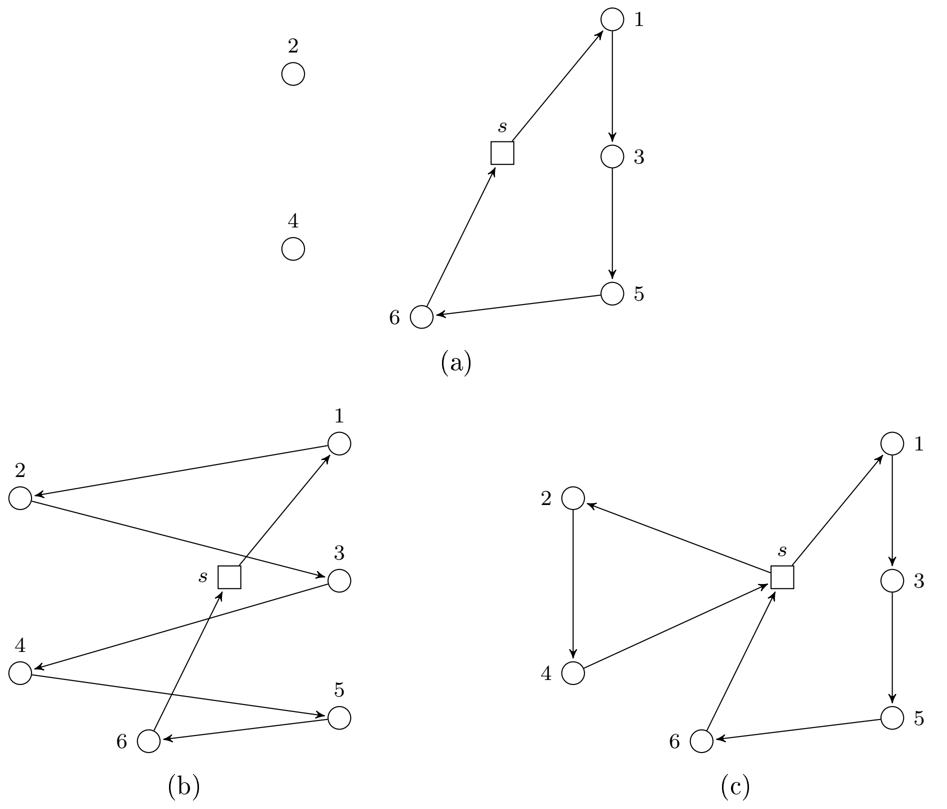

Figure 2: An example of the use of parameter $\bar{c}$ in the route building procedure. (a) is the last stage at which the procedure of Bräysy and our procedure remain the same. (b) is the end result of Bräysys route construction heuristic, and (c) is the result of our modified heuristic.

3.3 Or-exchanges

In routing problems, it can often be profitable to reorder the locations on a route in order to save costs or minimize the driving time. A very common way of reordering the locations on a route is by using 2-exchanges, where two arcs are removed from the route and they are replaced with two others. There is only one way in which the new arcs can be placed such that the resulting path is still a tour.

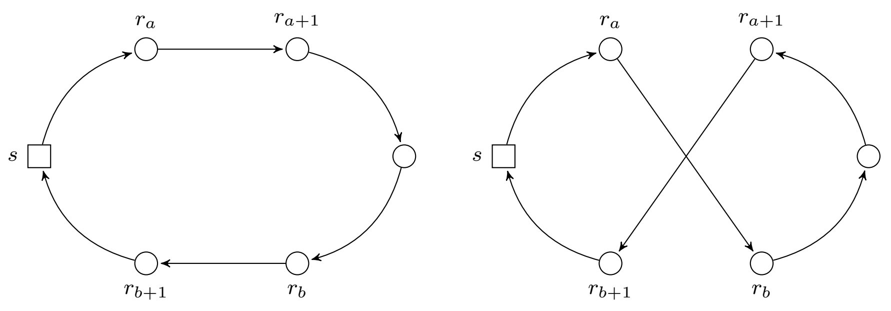

Figure 3: An example of a 2-exchange

An example of a 2-exchange is shown in Figure 3, where the arcs (r_a, r_{a+1}) and (r_b, r_{b+1}) are replaced with the arcs (r_a, r_b) and (r_{a+1}, r_{b+1}). As a result, the order in which the locations on the path $(r_b, \ldots, r_{a+1})$ are visited, also referred to as the orientation of a path, is reversed. In a vehicle routing problem with time windows, this is an undesirable property, since this increases the chance of the exchange being infeasible, especially when the path $(r_b, \ldots, r_a+1)$ is long or the time windows of the nodes on this path are short. Therefore, we shall not use the 2-opt neighborhood for the reordering of our routes. Instead, we shall use the Or-opt neighborhood, which has the property that the orientation of the paths after an exchange is not reversed. Or-exchanges are a subset of all possible 3-exchanges for a route, which are exchanges in which three arcs are removed and replaced with new ones such that the result is still a tour. An example of a 3-exchange can be found in Figure 4. The original route is displayed at (a), and the result of removing three edges is shown at (b). The result of placing three new arbitrary edges to construct a new route can be found in (c). As becomes clear from looking at the example, for 3-exchanges, there is not one straightforward way to place three new edges after the three edges have been removed from the tour as is the case in 2-exchanges, but there are several. Also, most of these outcomes have reversed orientations for one or more of the paths in the tour, which can be considered an undesirable property. Also, the computational effort required to verify 3-optimality can become very high. A route is said to be 3-optimal if there is no 3-exchange possible that decreases the costs of the route. Indeed, if we look at our example, the orientation of the path $(r_{a-1}, \ldots, r_{c+1})$ is reversed after the 3-exchange has been performed. Therefore, Or (1976) proposed to only consider the 3-exchanges in which a path of one, two, or three consecutive vertices is placed between two other vertices. An example of an Or-exchange is shown in Figure 4 (d), where the path (r_a, r_b) is relocated between the vertices r_c and r_{c+1}. It has been shown that the Or-opt procedure finds solutions of similar quality as the 3-opt procedure, but in considerably less time (Solomon and Desrosiers, 1988).

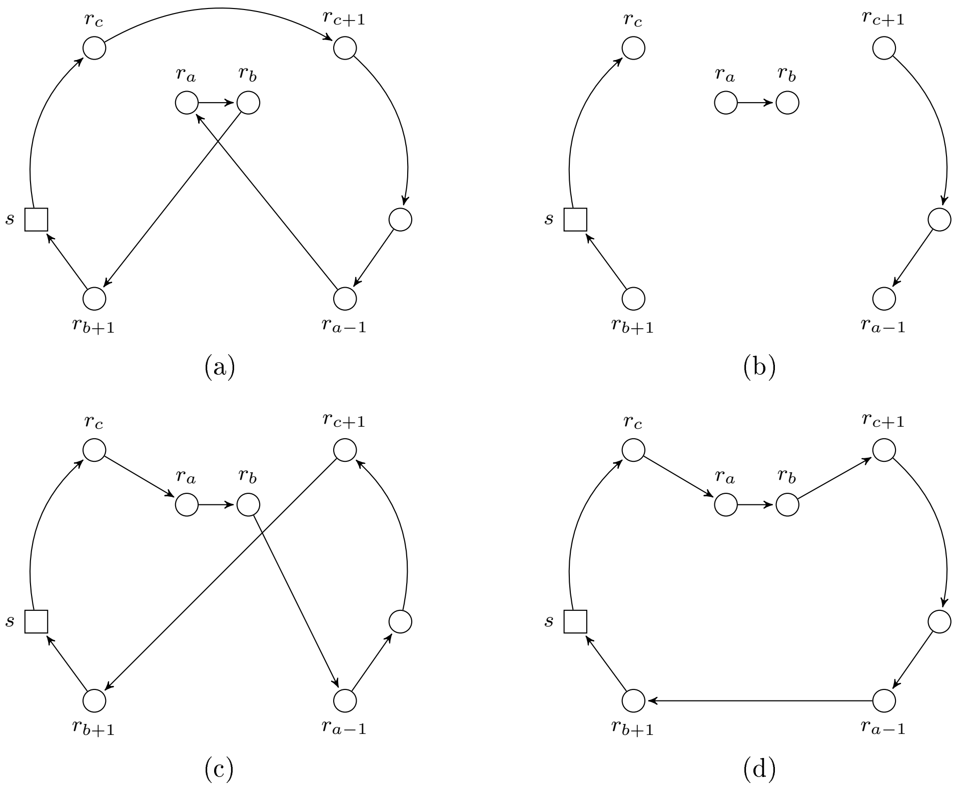

Figure 4: An example of an arbitrary 3-exchange and an Or-exchange. (a) is the original route, (b) is the route with three arcs removed, (c) is an arbitrary three exchange by placing three new arcs such that the result is still a tour, and (d) is the result of an Or-exchange

Similar to 3-optimality, a route is said to be Or-optimal if the costs of the route can no longer be decreased by performing an Or-exchange. A straightforward implementation of verifying Or-optimality has complexity $O(n^3)$, since there are $O(n^2)$ possible Or-exchanges and verifying that an exchange is profitable and feasible has complexity $O(n)$. However, Savelsbergh (1992) provides a lexicographic search strategy which decreases the time required for the verification of the feasibility and profitability of an exchange to constant time by maintaining a set of global variables. This search strategy requires us to divide the Or-exchanges into two different sets; backward Or-exchanges and forward Or-exchanges. A backward Or-exchange is an Or-exchange where a path $(r_a, \dots r_b)$ is relocated earlier on a route, while a forward Or-exchange moves the path $(r_a, \dots r_b)$ forward on the route. The example in Figure 4 is an example of a backward Or exchange.

The search for backward Or-exchanges is as follows. Firstly, recall that a path is denoted by $(0, r_1, r_2, \dots, r_d, n+1)$, where both 0 and $n+1$ are the depot, and r_d is the last node visited before returning to the depot. In the search for backward Or-exchanges, the idea is to relocate a path $(r_a, \dots r_b)$ between r_c and r_{c+1} earlier on the route, such that the resulting route becomes $(0, \dots, r_c), (r_a, \dots, r_b)$, $(r_{c+1}, \dots, r_{a-1}), (r_{b+1}, \dots, n{+}1)$. The algorithm starts by fixing the path $(r_a, \dots r_b)$, which is done as follows. First, the path length is set to 1, which implies that the path $(r_a, \dots r_b)$ has one vertex, and it is therefore simply r_a. We then fix r_a to be the second node visited after leaving the depot (r_2) on this route and we then consider all places earlier on the route at which the path $(r_a, \dots r_b)$ can be placed. We start by selecting $r_c = r_{a-2}$, so we consider to place $(r_a, \dots r_b)$ between r_{a-2} and r_{a-1}. If this is feasible and profitable, we place the path at the considered location, otherwise we select $r_c = r_{a-3}$, and we continue in this fashion until $r_c = 0$. We then increment r_a by one, if this does not cause r_b to be $n+1$, and try to place the new $(r_a, \dots r_b)$ path earlier on the route according to the earlier described strategy. Whenever r_a cannot be incremented anymore, we increment the path length by one, with a maximum length of three, and repeat the search for backward Or-exchanges. This search strategy is summarized in Algorithm 1. We shall describe the set of global variables and the corresponding functions and the feasibility and profitability functions later, since this requires us to introduce some new concepts and definitions first.

Note that in the search for backward Or-exchanges, the path $(r_{c+1}, \dots, r_{a-1})$ is that same path of the previously considered exchange, expanded with one edge. If the path of the previously considered exchange was $(r_{c+1}, \dots, r_{a-1})$, this path is expanded with the arc (r_c, r_{c+1}) for the current exchange, resulting in the path $(r_c, \dots, r_{a-1})$. The search for forward Or-exchanges is done in a similar fashion as the search for backward Or-exchanges. In this case, r_a is initialized as r_1, and (r_c, r_{c+1}) is chosen as $(r_{b+1}, r_{b+2}), (r_{b+2}, r_{b+3}), \dots, (r_d, n+1)$. For forward Or-exchanges, the path $(r_{b+1}, \dots, r_c)$ is the same path of the previously considered exchange, but with (r_{c-1}, r_c) added to the end. This is a crucial part of low computational complexity of the this lexicographic search strategy.

Algorithm 1 Pseudocode for backward Or-exchanges

```
input: a route $(0, r_1, r_2, \dots, r_d, n+1)$

START
for $pathLength = 1$ to 3 do
    for $r_a = 2$ to $r_d - (pathLength - 1)$ do
        initialize global variables
        for $r_c = r_{a-2}$ to 0 do
            update the global variables
            if the exchange is profitable then
                if the exchange is feasible then
                    perform the exchange
                    go to START
                end
            end
        end
    end
end
```

In order to define the functions which test the exchanges for profitability and feasibility, we are required to introduce the *concatenation theorem* first, which also plays a crucial role in this lexicographic search strategy. Savelsbergh proposed a concatenation theorem, which is of great importance in the search strategy. This theorem allows the calculation of the forward time slack of a concatenated path by using the forward time slack of the two smaller paths. The concatenation theorem for the case with service times is as follows:

Concatenation Theorem. If two feasible paths $(r_i, \dots r_j)$ and $(r_k, \dots r_h)$ with associated forward time slacks $F_{r_i}^{(r_i,\dots,r_j)}$ and $F_{r_k}^{(r_k,\dots,r_h)}$ for the first vertices are concatenated, the forward time slack for the first vertex of the resulting path is given by:

$$F_{r_i}^{(r_i,\dots,r_j,r_k,\dots,r_h)} = \min\{F_{r_i}^{(r_i,\dots,r_j)}, F_{r_k}^{(r_k,\dots,r_h)} + \sum_{i<p\leq j} W_{r_p} + D_{r_k} - (D_{r_j} + t_{r_j,r_k} + s_{r_k})\} \tag{53}$$

The proof for this theorem can be found in Appendix A. Using this formula, we can iteratively find the forward time slack at the depot through either backward or forward recursion with the following two formulas.

$$F_0^{(0,\dots,r_i,r_{i+1})} = \min\{F_0^{(0,\dots,r_i)}, l_{r_{i+1}} - D_{r_i} - t_{r_i,r_{i+1}} - s_{r_{i+1}} + \sum_{0<p\leq i} W_{r_p}\} \tag{54}$$

$$F_{r_i}^{(r_i,r_{i+1},\dots,n+1)} = \min\{l_{r_i} - D_{r_i}, F_{r_{i+1}}^{(r_{i+1},\dots,n+1)} + W_{r_i+1}\} \tag{55}$$

Earlier, it was mentioned that the resulting path of a backward Or-exchange can be defined as the concatenation of four paths, namely: $(0, \dots, r_c), (r_a, \dots, r_b)$, $(r_{c+1}, \dots, r_{a-1})$, $(r_{b+1}, \dots, n+1)$. The exchange is profitable if $c_{r_c,r_a} + c_{r_b,r_{c+1}} + c_{r_{a-1},r_{b+1}} < c_{r_c,r_c+1} + c_{r_{a-1},r_a} + c_{r_b,r_{b+1}}$. If the profitability of the exchange is asserted, we can test the feasibility of the resulting path of the Or-exchange by concatenating these paths to obtain $F_0^{(0,\dots,n+1)}$, and verify that it is positive.

Initializing the search. Now we shall summarize the approach that is used in calculating and maintaining the global variables that are used for asserting the feasibility of an exchange. For each of the four parts of $(0, \dots, r_c), (r_a, \dots, r_b)$, $(r_{c+1}, \dots, r_{a-1})$, $(r_{b+1}, \dots, n+1)$, the variables are initialized as follows:

Precalculation. We can calculate $F_0^{(0,\dots,r_c)}$, $F_{r_c}^{(r_c,\dots,n+1)}$, $\sum_{0<k\leq c}(s_{r_k} + t_{r_{k-1},r_k})$, and $\sum_{0<k\leq c} W_{r_k}$ in advance in $O(n)$ time for $\forall r_c \in \{0, r_1, \dots, n+1\}$ based on the current sequence of nodes on the path.

The path $(\mathbf{0}, \dots, \boldsymbol{r_c})$. When the path $(r_a, \dots r_b)$ is fixed, we can determine the total waiting time, the total travel and service time and the forward time slack for the path $(0, \dots, r_c)$ from the precalculated values.

The path $(\boldsymbol{r_a}, \dots, \boldsymbol{r_b})$. Using the values that were just obtained for the path $(0, \dots, r_c)$, we are able to calculate the departure time at r_a. This is equal to $D_0 + \sum_{0<k\leq c}(s_{r_k} + t_{r_{k-1},r_k}) + \sum_{0<k\leq c} W_{r_k} + W_{r_a} + s_{r_a}$, where W_{r_a} is equal to $\max\{0, e_{r_a} - (D_0 + \sum_{0<k\leq c}(s_{r_k} + t_{r_{k-1},r_k}) + \sum_{0<k\leq c} W_{r_k})\}$. We then initialize $F_{r_a}^{(r_a,\dots,r_b)}$, $\sum_{r_a<k\leq r_b} W_k$ and $\sum_{a<k\leq b}(s_{r_k} + t_{r_{k-1},r_k})$ by simply calculating, which is straightforward once D_{r_a} is known.

The path $(\boldsymbol{r_{c+1}}, \ldots, \boldsymbol{r_{a-1}})$. Based on the previously calculated values, we also know the departure for the first vertex of the path $(r_{c+1}, \ldots, r_{a-1})$, which at this point has length 1, and $c+1 = a-1$. The variable $F_{r_{c+1}}^{(r_{c+1},\ldots,r_{a-1})}$ is calculated as $l_{r_{c+1}} - D_{r_{c+1}}$, and the variables $\sum_{c+1<k\leq a-1} W_{r_k}$ and $\sum_{c+1<k\leq a-1}(s_{r_k} + t_{r_{k-1},r_k})$ are initialized as 0.

The path $(\boldsymbol{r_b} + \mathbf{1}, \ldots, \boldsymbol{n} + \mathbf{1})$. Since we can determine the departure time at $r_b + 1$ based on the departure time at r_{c+1} and the total waiting, travel and service time on $(r_{c+1}, \ldots, r_{a-1})$, we can easily calculate $F_{r_{b+1}}^{(r_{b+1},\ldots,n+1)}$. We do this by taking the precalculated value $F_{r_{b+1}}^{(r_{b+1},\ldots,n+1)}$ for the original path, adding the departure time at r_{b+1} of the original path, and subtracting the new departure time at this location. Clearly, we now have all the values needed for concatenating $(0, \ldots, r_c), (r_a, \ldots, r_b)$, and $(r_{c+1}, \ldots, r_{a-1}), (r_{b+1}, \ldots, n+1)$, and then concatenating the two resulting paths to obtain $F_0^{(0,\ldots,n+1)}$.

Updating the variables. If this exchange turns out to be infeasible, we consider the new path $(0, \ldots, r_{c-1}), (r_a, \ldots, r_b), (r_c, \ldots, r_{a-1}), (r_{b+1}, \ldots, n+1)$ in terms of the previously considered exchange.

The paths $(\mathbf{0}, \ldots, \boldsymbol{r_c})$ **and** $(\boldsymbol{r_a}, \ldots, \boldsymbol{r_b})$. Again, the variables for $(0, \ldots, r_c)$ and $(r_a, \ldots, r_b)$ can be calculated in constant time: We can calculate the variables for $(0, \ldots, r_c)$ by using the precalculated values, and since we can also find the departure time at r_a with this information, we can calculate the variables for $(r_a, \ldots, r_b)$ in a small number of calculations.

The path $(\boldsymbol{r_{c+1}}, \ldots, \boldsymbol{r_{a-1}})$. The global variables for the paths $(r_{c+1}, \ldots, r_{a-1})$ and $(r_{b+1}, \ldots, n+1)$ can also be obtained in constant time by using the variables of the previously considered exchange. The forward time slack for $(r_{c+1}, \ldots, r_{a-1})$ can be obtained using the concatenation theorem, where $F_{r_{c+2}}^{(r_{c+2},\ldots r_{a-1})}$ is the forward time slack of this path in the previously considered exchange, adjusted for the departure time at the first vertex. The total travel and service time for this path is also easily updated. However, the determination of the total waiting time on $(r_{c+1}, \ldots, r_{a-1})$ is slightly more difficult. For this purpose, we have to introduce the *backward time slack* $B_{r_i}^{(r_i,\ldots,r_j)}$, defined as the time the departure at r_i can be shifted backwards in time without causing additional waiting time. Mathematically this can be written as $B_{r_i}^{(r_i,\ldots,r_j)} = \min_{i\leq k\leq j}(D_{r_k} - e_{r_k} - s_{r_k})$. This variable is also easy to initialize and update in constant time for the path $(r_{c+1}, \ldots, r_{a-1})$. This can be done by adjusting this variable from the previously considered exchange for the departure time, and comparing it to the backward time slack of the new first vertex of the path. With this variable, the waiting time of a path that is created by concatenating two paths can be based on the waiting times of the two paths that are concatenated. How this is done is shown in Table 1, where $(r_i, \ldots, r_j)$ and $(r_k, \ldots, r_h)$ are concatenated. Here, Δ is the difference between the departure time at r_k in the original route and the departure time at r_k in the route after the Or-exchange that is currently considered. The old departure time is simply D_{r_k} and the new departure time is equal to $D_{r_j} + t_{r_j,r_k} + s_{r_k}$. Therefore, $\Delta = D_{r_j} + t_{r_j,r_k} + s_{r_k} - D_{r_k}$.

		$\sum_{i<p\leq h} W_{r_p}$
$\Delta \geq 0$	$\sum_{k<p\leq h} W_{r_p} = 0$	$\sum_{i<p\leq j} W_{r_p}$
	$\sum_{k<p\leq h} W_{r_p} > 0$	$\sum_{i<p\leq j} W_{r_p} + \max\{0, \sum_{k<p\leq h} W_{r_p} - \Delta\}$
$\Delta < 0$	$\sum_{k<p\leq h} W_{r_p} = 0$	$\sum_{i<p\leq j} W_{r_p} + \max\{0, -\Delta - B_{r_k}^{(r_k,\dots,r_h)}\}$
	$\sum_{k<p\leq h} W_{r_p} > 0$	$\sum_{i<p\leq j} W_{r_p} + \sum_{k<p\leq h} W_{r_p} + \Delta$

Table 1: The waiting time of the path $(r_i, \dots, r_h)$ if two feasible paths $(r_i, \dots, r_j)$ and $(r_k, \dots, r_h)$ are concatenated

The first case is very straightforward. If the waiting time on the second path is zero, while the departure at its first node is delayed, the total waiting time on the concatenated path is simply the waiting time on the path $(r_i, \dots, r_h)$.

In the second case, where the change in departure time is nonnegative and there is waiting time on the path $(r_k, \dots r_h)$, the delay in departure will cause some of the waiting time on the second path to decrease. Of course, the waiting time cannot become negative and therefore the maximum decrease in waiting time on $(r_k, \dots r_h)$ is equal to Δ.

The third case is the most complex one. In this case, the change in departure time at r_k is negative, and the waiting time along the path $(r_k, \dots r_h)$ is zero. The issue that needs to be adressed here, is the question whether the change in departure time is sufficient to cause waiting time on the path $(r_k, \dots r_h)$. This question can be answered by using the earlier introduced backward time slack.

And in the last case, when the departure time at r_k is advanced to an earlier time while there was already waiting time on the path $(r_k, \dots r_h)$, the delay will just cause extra waiting time on this path.

The path $(\boldsymbol{r_b + 1}, \dots, \boldsymbol{n + 1})$. Based on the previously calculated variables, we can again determine $F_{r_{b+1}}^{(r_{b+1},\dots,n+1)}$ by adjusting the precalculated value for the departure time at $i_2 + 1$. At this point, all variables have been updated and we can calculate $F_0^{(0,\dots,n+1)}$ to assert the feasibility of this exchange.

For the forward Or-exchanges a similar method can be applied. It will be slightly easier since there is no need of maintaining backward time slack variables for the calculation of the total waiting times. This is because in the search for backward Or-exchanges, the path $(r_{c+1}, \dots, r_{a-1})$ is expanded by adding an arc at the front, while in the search for forward Or-exchanges, the path is expanded by adding an arc at the end of $(r_{b+1}, \dots, r_c)$. While it is possible in the first case that the total waiting time decreases, this is not possible for the path $(r_{b+1}, \dots, r_c)$ in the search for forward Or-exchanges.

3.4 Route elimination

After the initial routes have been created and they have all been optimized by the Or-opt operator, Bräysy applies a route elimination procedure. As mentioned before, we have to decide on the use of this stage of the algorithm by running tests in a later stage.

The route elimination procedure starts by first considering the route with the least nodes for elimination. Once the route to be removed is selected for elimination, we try to deliver the packages that are delivered on this route on other routes. This is done by finding feasible insertion places on the other routes for any of the nodes at which the packages can be delivered. The insertion with the lowest value for Equation (52) is executed and the corresponding node is removed from the route to be eliminated. This is repeated until no nodes can be directly removed anymore.

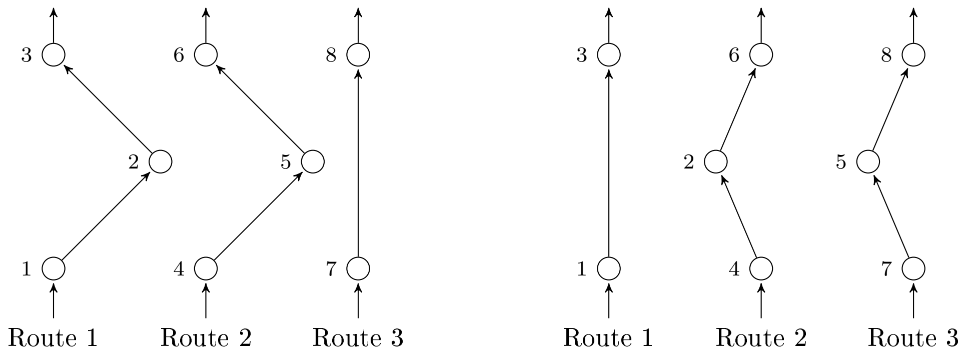

Figure 5: An example of an ejection chain for the VRPTW

After this, the ejection chain procedure starts. In VRPTW problems, the idea behind ejection chains is as follows: Consider a route R_k which we want to eliminate. We remove a node r_i from the route, but there are no feasible insertion places on any of the other routes. However, if we remove node r_j from another route R_l, it might be possible to place r_i on route R_l. In this case, the search for a feasible insertion place for r_j on other routes than R_k begins. If a direct insertion is possible, the ejection chain is finished. If a direct insertion place is not found, we can repeat the previous procedure by ejecting a node from another route R_m to make room for r_j. An example of this is shown in Figure 5, where Route 1 is the route to be eliminated. Direct insertion of node 2 into another route is not possible. However, by first ejecting node 5 from Route 2 and inserting node 2 in that same route, a feasible solution is obtained as node 5 can be directly inserted into Route 3. This ejection chain procedure can be improved for FDPTW problems. The package that was delivered at r_j might have multiple potential delivery locations. So instead of searching for insertion places only for node r_j, it is possible to look for insertion places for any of the nodes where the package can be delivered. An example of this is shown in Figure 6, where the package that is delivered at node 2 could also be delivered at node 9. Node 5 is moved to the left with regard to the example in Figure 5, which causes the insertion of node 5 into Route 3 to become infeasible. However, there is no need of an ejection chain for the elimination of node 2 from the first route, since instead of delivering the package at node 2, it can be delivered at node 9 on Route 3. This allows for much more flexibility in the route elimination process. However, this approach also increases the computational complexity by vastly increasing the number of feasible ejection chains.

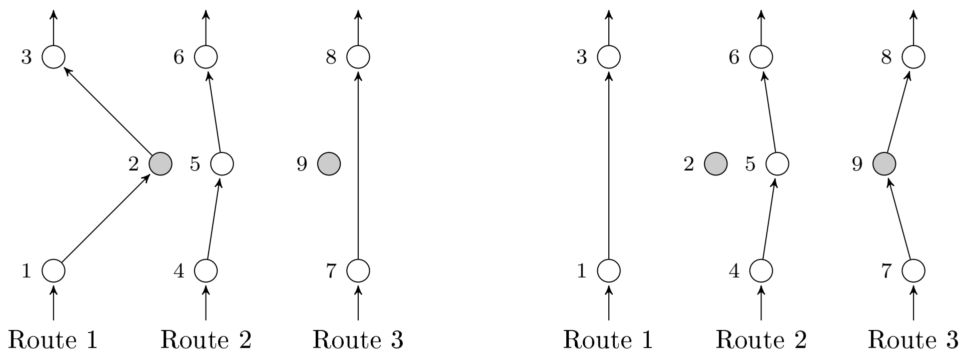

Figure 6: An example of an ejection chain for the FDPTW

The ejection chain procedure starts by removing the first node from the route considered for elimination. The parcel corresponding to this node now has to be delivered on another route. For each other route we remove the nodes one by one and check whether the package can now be delivered at one of its potential locations on that route. If so, the new node is inserted on the route, but the parcel corresponding to the node that was just removed is now not delivered anymore. Therefore, the process is repeated and we search for feasible insertion places by removing the nodes of the other routes one by one. If the parcel can be directly delivered on one of the other routes without having to eject a node first, the ejection chain is finished. Similar to Bräysy (2003), we use a breadth-first search strategy, first focusing on short ejection chains and later expanding the search to longer ejection chains. If there are multiple finished ejection chains of equal length, the one that minimizes the total difference in driving distance is executed. Furthermore, since the number of potential ejection chains grows exponentially in the length of the ejection chains, we place a limit $\bar{l}$ on the maximum length of ejection chains that are considered.

If a node cannot be removed from the route that is currently considered for elimination, or if no finished ejection chain can be found of a length lower than the threshold, another route will be considered for elimination. First, any changes to the current route are reversed, and then a post processing procedure is initialized. In this post processing algorithm, all nodes corresponding to parcels that are not delivered on the route that was previously considered for elimination are considered for insertion on that route. Again, the insertion that minimizes Equation (52) is executed, and the node where the parcel was delivered on another route is removed from that route. This procedure is repeated until there are no feasible insertions left. Then, the entire procedure is repeated with the shortest route that has not yet been considered for elimination. However, only short routes are considered for elimination (we shall assume that this is the case when they visit less than 10 nodes). If the shortest route consists of 10 nodes or more, we do not execute the route elimination procedure.

3.5 Route improvement

In order to minimize the total costs of a solution, we have developed a route improvement procedure. The route improvement procedure we use is based on the ejection chains which were also used in the route elimination procedure. Routes are improved by sequentially trying to find the most profitable exchanges by considering to remove each node from the route one by one. Pseudocode of the procedure can be found in Algorithm 2. All routes are considered for the removal of nodes one by one, and after every route has been considered the Or-opt procedure is executed on all routes individually.

Algorithm 2 Pseudocode for the route improvement procedure

```
input: a feasible set of routes

START
randomize order of routes in $routes$
$exchangeDone$ = true
while $exchangeDone$ = true do
    for $route \in routes$ do
        try to decrease the total costs by removing nodes from $route$
    end
    if All routes remained the same in the for-loop then
        $exchangeDone$ = false
    end
    for $route \in routes$ do
        execute Or-opt procedure on $route$
    end
end
```

We shall now explain how the solutions are improved by removing nodes from the routes. All nodes on a route are considered for removal one by one. First, the node is removed from the route. If it is profitable to create a separate route which delivers only this parcel at any of its potential delivery locations, the decrease in total costs of this solution is stored. Then, it is checked whether any of the potential delivery locations for the parcel can be directly inserted in another route, or somewhere else on the current route. If this is the case, and when the decrease in total costs with respect to the original solution is larger than the decrease in costs of the currently stored exchange, the exchange is kept in memory. Of course, it would be redundant to consider relocating the node on the route it was already on, since this is already efficiently considered in the Or-opt procedure. Then, all ejection chains with a length of two are considered. Remember that an ejection chain of length two consists of two ejections and two insertions. The reason only ejection chains of length two are considered is that we do not use the breadth-first search strategy that was employed earlier. If we would also include ejection chains of length three or longer, it would have enormous consequences on the computational- and memory complexity of the heuristic. Again, if the cheapest solution in the ejection chain stage has a higher decrease in cost than the one that is currently stored, the exchange currently in memory is replaced with the cheaper one. This procedure is repeated for all nodes on the route. Then, the exchange that has the highest reduction in costs is executed, and the entire procedure is repeated for this route. If there is no exchange found that leads to a reduction in costs, the same procedure is applied on the next route. When there are no exchanges found that lead to a reduction in costs for any of the routes, the procedure is terminated. This last step is executed in a slightly smarter fashion than it is described in the pseudocode. Every time a node is eliminated from a route, the route from which this node was eliminated in stored in memory. As soon as this route is considered again in the next run through the for-loop of routes, and this is still the last route from which a node was eliminated, the for-loop is terminated prematurely, since this means that there are no more profitable exchanges to be found.

This procedure does not require an exhaustive search strategy as was the case with Or-exchanges for efficient implementation, since maintaining the set of the variables D_{r_i}, W_{r_i} and $F_{r_i}^{(r_i,\ldots,n+1)}$ for each node r_i on each route suffices. For all routes other than the route which is currently considered for node removal, these variables are already known from previous stages of the heuristic. For the route from which nodes are currently being removed, we calculate the variables D_{r_i}, W_{r_i} and $F_{r_i}^{(r_i,\ldots,n+1)}$ after the removal of the node, so feasibility of insertions can then be verified easily. If a node is inserted in or removed from a route, the variables for this route are updated. The three conditions mentioned in Section 3.2 can be used to verify the feasibility of considered insertions. For the insertion of other potential delivery locations of the node that was just removed on the same route as it was removed from, a similar search strategy as was used in the search for Or-exchanges could be beneficial. However, if the amount of other potential delivery locations is high, it might be faster to recalculate the D_{r_i}, W_{r_i} and $F_{r_i}^{(r_i,\ldots,n+1)}$ variables for the entire route since the feasibility checks then require very few calculations. Therefore, we decide against the use of such a search strategy.

We have now explained our default route improvement procedure. However, there are various adjustments possible for the route improvement procedure to possibly alter the trade-off between runtime and solution quality. For example, in the current version of the the procedure, the order of the routes is randomized before the search for profitable exchanges starts. However, it might also be beneficial to order the routes from shortest to longest, where a route is considered shorter than another route when there are less parcels delivered. This way, the chance that the shortest route is eliminated in its entirety is larger than if the routes were ordered randomly. But always ordering the routes from shortest to longest could also decrease the quality of the solutions, since there is less diversity in the route improvement procedures between various solutions. Therefore, we propose a third option, where there is a 50% chance that the routes are ordered randomly, and a 50% chance that the routes are ordered from shortest to longest. Thus, we have three options, which we shall refer to as follows:

random	The routes are ordered randomly.
sorted	The routes are ordered shortest to longest.
mixed	There is a 50% chance that the routes are ordered randomly, and a 50% chance that the routes are ordered from shortest to longest.

If the entire route improvement procedure proves to be computationally too complex for very large or difficult problem instances, there are various adjustments possible that will speed up the process. For example, currently we consider all nodes on a route for removal and then the best exchange found is executed. Instead, we could consider a more greedy approach in which a node is considered for removal and then the best exchange for this node is executed if the corresponding reduction in costs is positive. This is then repeated for each node on the route. Along with these two options, we also implement a third option. In this case, the original solution is copied first. Then the greedy version of the route improvement procedure is applied on one copy, while the extensive route improvement procedure is applied on the other. Only the best of the two resulting solutions is stored. We refer to these three options as:

greedy	The best exchange is executed after all possible exchanges for a node are considered, if it leads to a reduction in costs.
extensive	The best exchange is executed after all possible exchanges for each node on a route are considered, if it leads to a reduction in costs.
both	Both the greedy and the extensive procedures are applied on a copy of the original solution, and the best result of the two is kept.

Of course, there are many more alterations possible. For example, another approach would be to immediately execute every feasible exchange if the corresponding percentage decrease in costs is larger than a certain threshold. We will however restrict ourselves to the methods specified above.

3.6 On the use of the route elimination procedure

We have not decided on the use of the route elimination procedure in our heuristic yet. However, for reasons which will become clear in Section 5.2, we propose a potential modification to the heuristic. We do not use this modification until stated otherwise in Section 5.2, but it will be explained here. This alteration allows the heuristic to provide both solutions in which the route elimination procedure has been used, as solutions in which it has not been used. We do this by first creating a number of initial solutions as specified by the parameter *solutions*. Then, every route is copied, and the route elimination procedure is applied on the copy of each solution. Every copy in which the costs have not been altered by the route elimination procedure are removed, and the remaining copies are added to the collection of all original solutions. We are then left with a total number of solutions between *solutions* and $2 \times$ *solutions*. The route improvement procedure is then applied on the $x \times$ *solutions* solutions with the lowest costs.

4 Benchmark Instances

4.1 The Solomon instances

Since the CVRP and the VRPTW are widely studied variants of the VRP, there are various benchmark instances available for these problems. These instances can be used to compare the performance of solution methods against other methods that have been used to solve these instances. The most well-known benchmark instances for the VRPTW are the Solomon instances (Solomon, 1987). Since there are no benchmark instances available for the FDPTW, we have to create a new set of instances. We shall partially base those instances on the Solomon instances, since their wide acceptance in vehicle routing literature supports the presumption that they are of good quality.

When creating benchmark instances, it is important to take into account all factors that might significantly influence the behavior of the algorithms. Varying these factors in the test instances makes it possible to find the potential strengths and weaknesses of the various heuristics, and it allows us to find the most robust heuristic. According to Solomon, factors of a VRPTW instance that might influence the behavior of heuristics include:

Geographical data. Solomon deals with the geographical aspect by varying the geographical distribution of customers. The instances know three different distributions; randomly scattered over a square according to a uniform distribution (R), clustered (C) or semi-clustered (RC), where half the customers are clustered and the other half is located randomly by a uniform distribution.

The number of customers served by a vehicle. The number of customers served by a vehicle can be varied by altering the length of the scheduling horizon, $[e_0, l_0]$. For this, Solomon considers two options; a short scheduling horizon, and a long scheduling horizon. The short scheduling horizon allows approximately 5 to 10 customers to be served per vehicle, while the long scheduling horizon provides enough time for a vehicle to visit over thirty customers before returning to the depot.

Time window characteristics. The time window characteristics can be varied in two ways. The first way is to vary the amount of customers that are time-constrained. In the creation of the Solomon instances, problems were created with 25%, 50%, 75%, and 100% of the customers having time constraints. For the randomly placed customers, time windows were created by randomly generating the end of the time window for customer i on the interval $(e_0+t_{0i}, l_0-t_{i0}-s_i)$. However, Solomon defines the time window as the time in which the service should start, while we assume that the service should be completed within the time window. In our case, we should generate the end of the time window for location i on the interval $(e_0 + t_{0i} + s_i, l_0 - t_{i0})$ if we were to use the same approach. The width of the time windows is also varied, but the exact parameters used for determining the width of the time interval are not stated.

4.2 FDPTW benchmark instances

Obviously, the aforementioned factors are all factors that play a role in the creation of benchmark instances for the FDPTW as well. The latter two could be dealt with in a similar fashion as done in the creation of the Solomon instances, but the geographical factor needs some elaboration. In instances for the VRPTW, the only geographical factor is the distance between customers. However, for the FDPTW one also needs to take into account the distances between various potential delivery locations for a package. To make sure there is enough variation in that factor, in the clustered versions of the benchmark instances packages are assigned to a cluster. In the semi-clustered instances, 50% of the packages will be assigned to a cluster. If a package can be delivered at multiple locations and the package is assigned to a cluster, the delivery locations for this package are within the same cluster. This approach ensures that in the clustered instances the delivery locations for one package are always close together, while the distance between these delivery locations can be large in the random instances. In the semi-clustered instances, there are both packages which have their delivery locations close together, as well as packages with large distances between their delivery locations.

The introduction of packages in the FDPTW brings about two other important factors that can influence the behavior of heuristics and which should therefore be varied in the benchmark instances. The first of those two is the amount of delivery locations that each package should have. The second is the amount of unique geographical locations to the amount of packages ratio. If this ratio is low (e.g. < 0.5) there are a lot of packages, while there are only relatively few locations. If the time windows allow it, this causes vehicles to be able to deliver multiple packages to a location in one visit. This means that the total distance driven could be significantly lower, which might result in an easier problem.

If we were to vary over all the factors mentioned by Solomon, and also the FDPTW specific factors mentioned above, the amount of benchmark instances would rise well above 200. This might however cause the total computational times to become too high, depending on the speed of the algorithms. Therefore, we decide to focus on the FDPTW specific factors, regarding the packages and their delivery locations. Where Solomon also varied the percentage of customers with time windows, we shall keep this fixed at a 100%. Solomon also considered various distributions for the length of the time intervals, but we shall distinguish between only two values; long and short time windows. This leaves us with five factors to vary between the instances, resulting in a total of 72 benchmark instances. The factors and their possible values can be found in Table 2. For the number of delivery locations per package, x%/y%/z% means that x% of the packages has one delivery location, y% of the packages has two delivery locations, and z% of the packages has three delivery locations. With the currently chosen values, the effects of more packages with more delivery locations per package should become clearly visible in the results. In every test instance, a total of 50 packages has to be delivered, but the number of unique locations is varied. This allows us to see the results of the possibility to deliver multiple packages at one location. The alphanumeric values between parentheses are used to refer to the problems, where we use the notation RSsAa to refer to a problem instance with the following features; Random locations, short horizon length short time windows, 75%/25%/0%, and one unique location for every two packages to be delivered.

factor	values
geographical distribution (GD)	random (R), clustered (C), semi-clustered (RC)
horizon length (HL)	short (S), long (L)
length of time windows (LOTW)	short (s), long (l)
locations per package (LPP)	75%/25%/0% (A), 20%/60%/20% (B)
unique locations to package ratio (ULTPR)	1/2 (a), 1/1 (b), 2/1 (c)

Table 2: The attributes which are varied in creating the benchmark instances with their values

Now that there is a set of fixed parameters selected for the creation of benchmark instances, we shall give an in-depth explanation of the process used in generating those instances. The area in which the locations are generated is a rectangle with a width of 150 units and a height of 100 units, as shown in Figure 7. The depot, denoted by a small cross in the figure, is always located in the middle of the rectangle, since a very poorly located depot might skew the results too much. If the geographical distribution is random, the locations are placed by independently sampling x- and y-coordinates from a uniform distribution. If the geographical distribution used is clustered, the number of locations per cluster is divided as evenly as possible over the clusters. Then, x-coordinates for the locations within the cluster are sampled from a normal distribution with as mean the x-coordinate of the center of the cluster. The standard deviation of the distribution is chosen such that 99% of the generated coordinates fall within the horizontal limits of the cluster. In our case, this boils down to choosing σ such that $P(\mu - 2.58\sigma < X < \mu + 2.58\sigma) = 0.99$. Since $\mu - 2.58\sigma$ should be equal to $\mu - 25$ and $\mu + 2.58\sigma = \mu + 25$, $\sigma = 9.71$. The same approach is used for generating the corresponding y-coordinates. If the distribution is semi-clustered, half of the points are randomly located over the entire map, and the other half is divided over the clusters. The number of clusters used varies with the amount of generated locations. If there are only 25 locations, only clusters II and III are used. If there are 50 locations, clusters I, II and III are used, and if there are 100 locations all four clusters are used.

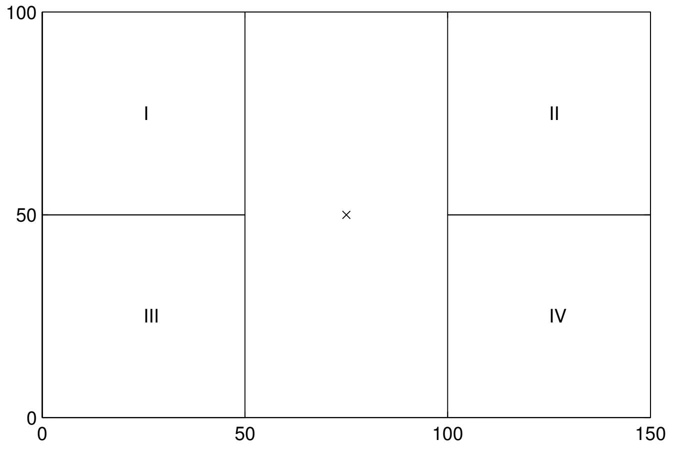

Figure 7: The lay-out of the area which is used in the creation of all benchmark instances

The packages have a randomly generated weight within the interval $[1, 5]$. The packages have a 20% chance of having a service time drawn from the uniform distribution with range $[1, 10]$, and a 80% chance of having no service time. If the geographical distribution of the locations used is random, the packages are generated and they are then randomly assigned delivery locations from the set of generated locations. If the distribution used is clustered, half of the packages can only be delivered at the locations which were randomly generated over the entire area, while the other half is evenly distributed over the clusters. The packages are then assigned delivery locations within the cluster to which the packages themselves have been assigned.

The length of a long planning horizon is 600 units and the length of a short planning horizon is 250 units. Since the driving time is equal to the driving distance, this means that uncapacitated vehicles could drive around the entire rectangle once with a long horizon, while the driving time is severely more restricted in the case of a short planning horizon. We also restrict the number of packages a vehicle can carry by setting the carrying capacity to 60 in the case of a long planning horizon, and 20 in short planning horizons, so vehicles are not able to carry an unrealistic amount of packages. This also eliminates the possibility of very unrealistic solutions in which one vehicle delivers almost every package, while others deliver almost none.

The length of a short time window is generated by sampling from a uniform distribution with range $[0.2x, 0.3x]$, where x is the length of the planning horizon. Similarly, long time windows are drawn from the uniform distribution with range $[0.3x, 0.6x]$.

Fractional values can be encountered when dividing the number of locations and the number of packages over the area and the clusters. Whenever this happens, the numbers are manually modified so that they are as close to the reported distribution as possible and the total number of packages and locations always match the reported numbers.

In some cases, we require instances with a smaller number of packages to be delivered. Therefore, we also create a set of benchmark instances with 12 parcels to be delivered, and a set with 16 parcels to be delivered. However, whenever the number of packages becomes too low, the total number of locations would be extremely low in the case where there is only a single unique location for every two packages. The unique locations to package ratios will therefore be changed to 1/1 (a*) , 2/1 (b*), and 4/1 (c*) whenever we use benchmark instances with less than 50 parcels. We do not change any of the other parameters, so a short horizon length would allow for two or three vehicles in the solution, while it is likely that all packages could be delivered by a single vehicle in the case of a long planning horizon.

4.3 Urban instances

In order to investigate the impact of the introduction of the possibility of having multiple delivery instances per package, we shall create another set of instances. This set of instances will be designed in a way that matches reality closer than the previously constructed FDPTW test instances. In order to do so, we shall create a map based on the topography of a theoretical city where there are a 100 parcels to be delivered to 100 different inhabitants of this map. These parcels can be delivered at the homes of customers, and they are assigned one or more delivery locations from a set of pick-up points and offices. A map consists of a city center with office buildings, surrounded with a densely populated area. Beyond this densely populated area there is a sparsely populated area, after which the suburbs begin. Along with locations for homes and offices, we also generate locations for the pick-up points. We shall now explain the procedure of creating these instances in more detail.

An example map is shown in Figure 8. Each map consists of 100 homes, 25 offices, a depot and five pick-up points. The depot of the supplier is fixed at the border of the city, with coordinates $[30, 15]$. The locations of the offices in the city center are then generated by sampling random x- and y-coordinates from a normal distribution with center $[25, 30]$ and standard deviation 4. The locations of the pick-up points are fixed at their locations as shown in Figure 8.

The generating of the locations of the homes is slightly more complex. There are 15 homes in each suburb, 40 homes around the city center, and another 15 homes are placed at random. The homes in the densely populated ring outside the city center are scattered uniformly over a circular ring (also known as doughnut)

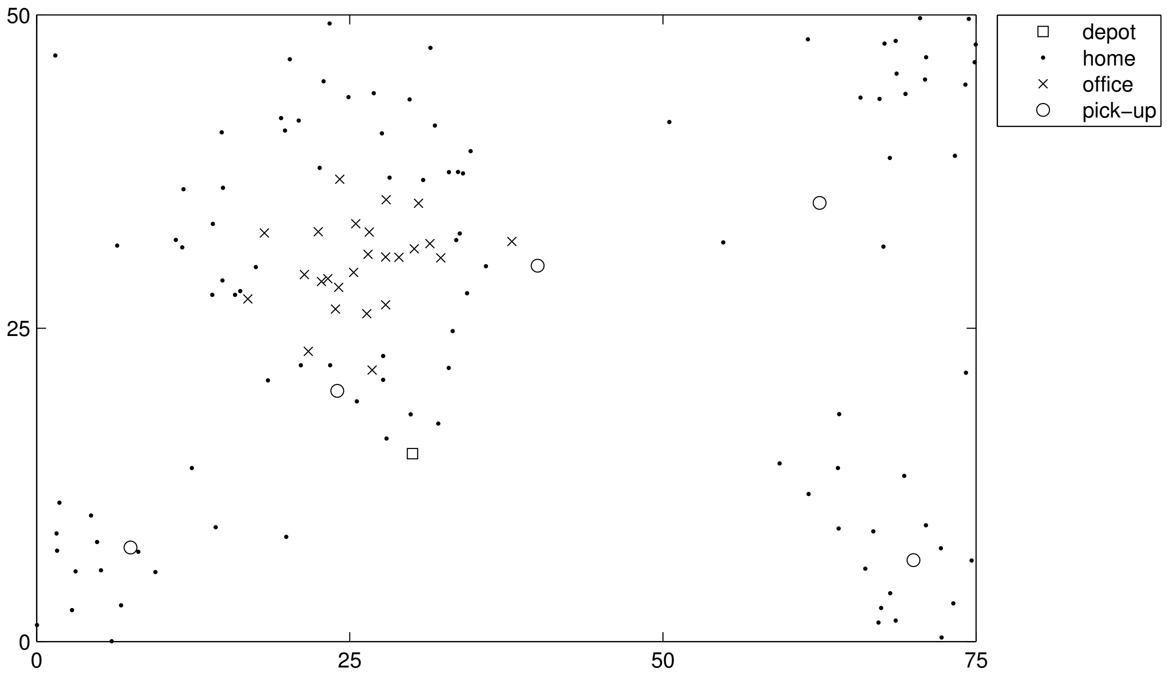

Figure 8: An example of a map of the urban benchmark instances

with inner radius $R_1 = 7.5$ and outer radius $R_2 = 15$ with center $[25, 30]$. This is achieved by randomly sampling a variable U_1 from a Uniform distribution within the interval $[0, 2\pi]$, and sampling a second variable U_2 from a Uniform distribution within the interval $[0, 1]$. We then apply the equation $R = \sqrt{(R_2^2 - R_1^2)U_2 + R_1^2}$ to obtain the distance of the location to the center. We use U_1 as the angle at which the point is placed, such that the coordinates of the new location can be calculated as $x = 25 + R * cos(U_1)$ and $y = 30 + R * sin(U_1)$. After the homes directly around the city center have been generated, we generate the homes in the suburbs. The suburbs are located in the bottom-left corner, the bottom-right corner, and the top-right corner. The locations within the bottom-left suburb are generated by independently generating x- and y-coordinates from a normal distribution with mean 0 for both coordinates and a standard deviation of 7. This approach leads to the possibility of points being generated with coordinates that do not lie within the borders of our map. This issue is easily resolved by imagining a coordinate system with its origin at $[0, 0]$. If the point falls in the top-left quadrant of this system, it is mirrored along the vertical axis (i.e. its x-coordinate is multiplied by -1), such that it falls in the top-right quadrant. Similarly, if it falls in the bottom-right quadrant it is mirrored horizontally, and if it falls in the bottom-left quadrant it is mirrored diagonally. A similar approach is used in the creation of the other suburbs, where the mirror-rules are adjusted accordingly. Also, the standard deviation used in generating the other suburbs is 10. Finally, 15 points are randomly placed by using a Uniform distribution that spans the entire area.

Once all the locations have been generated, we assign time windows and delivery locations to the 100 packages. First, every home is assigned exactly one unique parcel, and the weight of the parcel is randomly sampled from a discrete Uniform distribution on the interval $[1, 5]$, such that the average parcel weighs 3 units. All parcels should be delivered between 8:00 and 20:00, but the time windows for the homes are often a lot shorter, since most people are away from home during the day. For 70% of the homes, the inhabitants are at work or school during daytime, but there is at least one person home after 17:00, leading to a time window of [17:00, 20:00]. There are also some homes where there is a person present during the entire day with a time window of [8:00, 20:00] (15%), or homes where there is no one present until 19:00 (15%), such that the time window becomes [19:00, 20:00].

The assigning of office buildings or pick-up points as delivery locations to parcels is done based on two parameters, p_{office} and $p_{pick-up}$. These parameters represent the chance that a parcel can also be delivered at an office or at a pick-up point respectively. For offices, there is a p_{office} chance that it can be delivered at an office, in which case the office at which the parcel can be delivered is selected at random. There is also a $p_{pick-up}$ chance that a parcel can be delivered at the pick-up point that is spatially closest to the home at which the parcel can be delivered. A parcel should always be delivered at an office or pick-up point between 8:00 and 16:00.

Vehicles have a capacity of 75 units, such that they are able to carry 25 parcels of average weight. Furthermore, they are allowed to depart from the depot at 8:00 and they should be back at 22:00.

In order to find the influence of the offices and the pick-up points on the costs, we create four versions of every generated instance. The first version is the original instance. The second version is the original instance in which all offices are removed from the map and thus customers who previously were able to receive their package at their office now no longer have this opportunity. For the creation of the third version, all pick-up points are removed from the original instance in a similar fashion. In the fourth version, both the offices and the pick-up points are eliminated from the map, and all parcels have to be delivered at home. For future reference, we use the letters A, B, C, and D to refer to these four versions of each instance respectively.

We will create five urban instances with various settings for the p_{office} and $p_{pick-up}$ parameters, each instance with its four versions A, B, C and D. For the parameters we use the following values: $(p_{office}, p_{pick-up}) = (0.3, 0.2), (0.2, 0.3), (0.5, 0.5), (0.7, 0.3), (0.3, 0.7)$.

5 Parameter selection and heuristic settings

The developed heuristic has a lot of parameters and settings that can be tuned to alter the trade-off between runtime and solution quality. In Section 5.1 we shall perform a grid search to find the best combination of parameters. For this purpose, we use the default implementation of our route improvement procedure, i.e. with the random setting for the order of the routes and the extensive setting in the search for exchanges as explained in Section 3.5. After the values for the parameters have been chosen, we will choose the combination of settings for the route improvement procedure that gives the best trade-off between solution quality and runtime in Section 5.2.

The implementation of the heuristic is done in Java. All experiments are done on a PC with a Intel Core i5 2500K 3.3GHz quad core processor and 8GB of DDR3 RAM. The implementation of the heuristic only runs on a single core of the processor. This means that ideally, the heuristic could become four times as fast by altering the implementation such that it uses all four cores of the processor. It is easy to imagine various ways in which we could speed up our heuristic by using multi-threading. For example, in the route construction stage, each core could create a new solution, such that four solutions can be constructed simultaneously. Another approach would be to create one solution at a time, and to have each core check the feasibility and costs of potential insertions, such that a new node can be inserted about four times as fast. The same two approaches can be used in multi-threading the route improvement stage.

5.1 Parameter selection

The local search heuristic makes use of a lot of parameters. There are two parameters that are used through the entire heuristic, which is the following:

solutions	The total number of solutions to be made
x	The percentage of solutions with the lowest costs with which the heuristic should continue after the route construction stage

The other parameters can be divided over the stages of the heuristic. The route construction stage makes use of the following parameters:

$\alpha_1, \alpha_2, \alpha_3$	The weights for the cost evaluation function
$\bar{n}$	The number of tertiary nodes to be added to the set S
$\bar{d}$	The distance threshold for the consideration of the insertion of nodes
$\bar{c}$	The cost threshold for executing insertions based on their cost function value
$\bar{k}$	The total number of nodes to be added to a route before it is reordered using the Or-opt operator

The route elimination stage has the following parameters:

$\alpha_1, \alpha_2, \alpha_3$	The weights for the cost evaluation function
$\bar{l}$	The maximum length of the ejection chains

Here, the weights for the cost evaluation function in the route construction stage and the route elimination stage are equal. Our route improvement procedure does not use any parameters. Due to the large amount of parameters in the heuristic it is very difficult to find the combination of parameters that provides the best results. One potential approach is to use only a very small subset of the benchmark instances with 50 parcels in the parameter optimization process. However, it is difficult to find a small subset that would be representative for the entire set of benchmark instances due to the large variability in properties of the benchmark instances. Another approach would be to use instances with less parcels in the parameter selection, but this might skew the results. For example, the influence of the parameter $\bar{c}$ is larger in small instances, since it is usually only actively used in the construction of the last routes in the construction stage. Because of these complications, we shall use another approach to select the parameters for the heuristic. In our heuristic, the most important and time consuming stage, the route improvement stage, does not use any parameters. On the other hand, the route construction stage, which amounts for negligible time in the heuristic's run time, makes use of a lot of parameters. Therefore, if we could assert that the best solutions of the route construction stage also provide the best solutions after the route improvement stage, we could optimize the parameters by only optimizing the solutions created in the route construction stage. This would allow for a much more extensive search for parameters.

To verify that good solutions of the route construction stage provide good results in the route improvement stage, we will apply the heuristic to the benchmark instances with parameter values that were obtained by preliminary testing. The parameter values used are the following:

solutions	24
x	100%
$\alpha_1, \alpha_2, \alpha_3$	$\{1, 0, 0\}$
$\bar{n}$	4
$\bar{d}$	100
$\bar{c}$	$\{60, 200\}$
$\bar{k}$	5

We do not use the route elimination stage, since preliminary testing showed that this does not improve results, while having a slightly longer runtime. Note that we have specified two values for the parameter $\bar{c}$. Preliminary tests on small instances showed that solutions for some instances with long planning horizons improved a lot by using a low value for $\bar{c}$, while other instances became worse. Since in practice it may be difficult to determine whether the planning horizon is considered long or short, and to maintain robustness of the heuristic, we always use both values. The value of 60 is used in the creation of the first 12 solutions, and $\bar{c} = 200$ is used in the creation of solutions 13 to 24.

To obtain more information on the importance of the costs after the route construction stage on the final solution, we have included Figure 9. From this figure we learn that most good solutions were obtained from solutions that had below average costs for that instance after the route construction stage. In total, 67.4% of the solutions that were within 1% of the best solution found were obtained by improving a solution with below average costs after the route construction stage. For solutions within 2% of the best found solutions, this was 63.6%. In 52 out of the 72 instances, the best solution was found by improving a solution with below average costs after the route construction stage, and in all cases either the best or the second best solution would have been found by only improving those solutions. It is important to note that the average difference in costs between the best and the second best solution is only 1.15%.

We perform a grid search to find the best set of parameter values for the route construction stage. Since it is computationally infeasible to include every parameter in the grid search and still vary the values for these parameters sufficiently, only a subset of the parameters is used in the grid search. The parameters that are most dependent on each other, and which should therefore be optimized simultaneously are $\alpha_1, \alpha_2, \alpha_3$, $\bar{d}$ and $\bar{c}$. However, due to our introduction of the parameter $\bar{c}$, the importance of $\bar{d}$ has decreased significantly. The main purpose of $\bar{d}$ is now to decrease the runtime of the route construction stage if its runtime becomes too high, by choosing a lower value for $\bar{d}$. The parameters *solutions*, $\bar{n}$ and $\bar{k}$ can be modified to alter the trade-off between speed and solution quality, and these can be considered to be independent from $\alpha_1, \alpha_2, \alpha_3$, $\bar{d}$ and $\bar{c}$. Therefore, we can choose proper values for these parameters afterwards. We use the following values for the aforementioned parameters in our grid search:

solutions	24
x	100%
α_1, α_2	$\{1,0\}, \{0.9,0.1\}, \{0.8,0.2\}, \{0.7,0.3\}, \{0.6,0.4\}, \{0.5,0.5\}$
α_3	$0, 0.1, 0.2, 0.3, 0.4, 0.5$
$\bar{n}$	4
$\bar{d}$	100
$\bar{c}$	$\{20,20\}, \{20,40\}, \{20,60\}, \{20,100\}, \{20,150\}, \{20,200\}, \{40,40\}, \{40,60\}, \{40,100\}, \{40,150\}, \{40,200\}, \{60,60\}, \{60,100\}, \{60,150\}, \{60,200\}, \{100,100\}, \{100,150\}, \{100,200\}, \{150,150\}, \{150,200\}, \{200,200\}$
$\bar{k}$	5

We apply the route construction procedure on all 72 benchmark instances with every possible combination of the mentioned values for the parameters. A bar plot showing the influence of the parameters α_1, α_2 and $\bar{c}$ is shown in Figure 10. From this figure it becomes clear that decreasing α_1 while increasing α_2 provides worse results than giving full priority to travel costs in the cost evaluation function. The conclusion that can be drawn with regard to the parameter $\bar{c}$ is slightly less clear. Any combination of two values where one of the values is equal to 40 outperforms the others. This might suggest that the best solutions are always found with the setting $\bar{c} = 40$, but this is not the case. For example, with the setting $\{40, 100\}$, in approximately 28% of the instances the best solution was found in a case where $\bar{c}$ was 100. We expect that having a diverse set of initial solutions might prove to be beneficial for the route improvement stage. Therefore, we shall use $\bar{c} = \{40, 100\}$ in our heuristic. Of the twenty parameter setting combinations that provided the lowest average costs, there were exactly five settings with $\alpha_3 = 0.2$, five with $\alpha_3 = 0.3$, five settings with $\alpha_3 = 0.4$ and five settings with $\alpha_3 = 0.5$. From this one can conclude that it is important to give priority to nodes that are located far from the depot, but there is no use in increasing the priority after the threshold of 0.2.

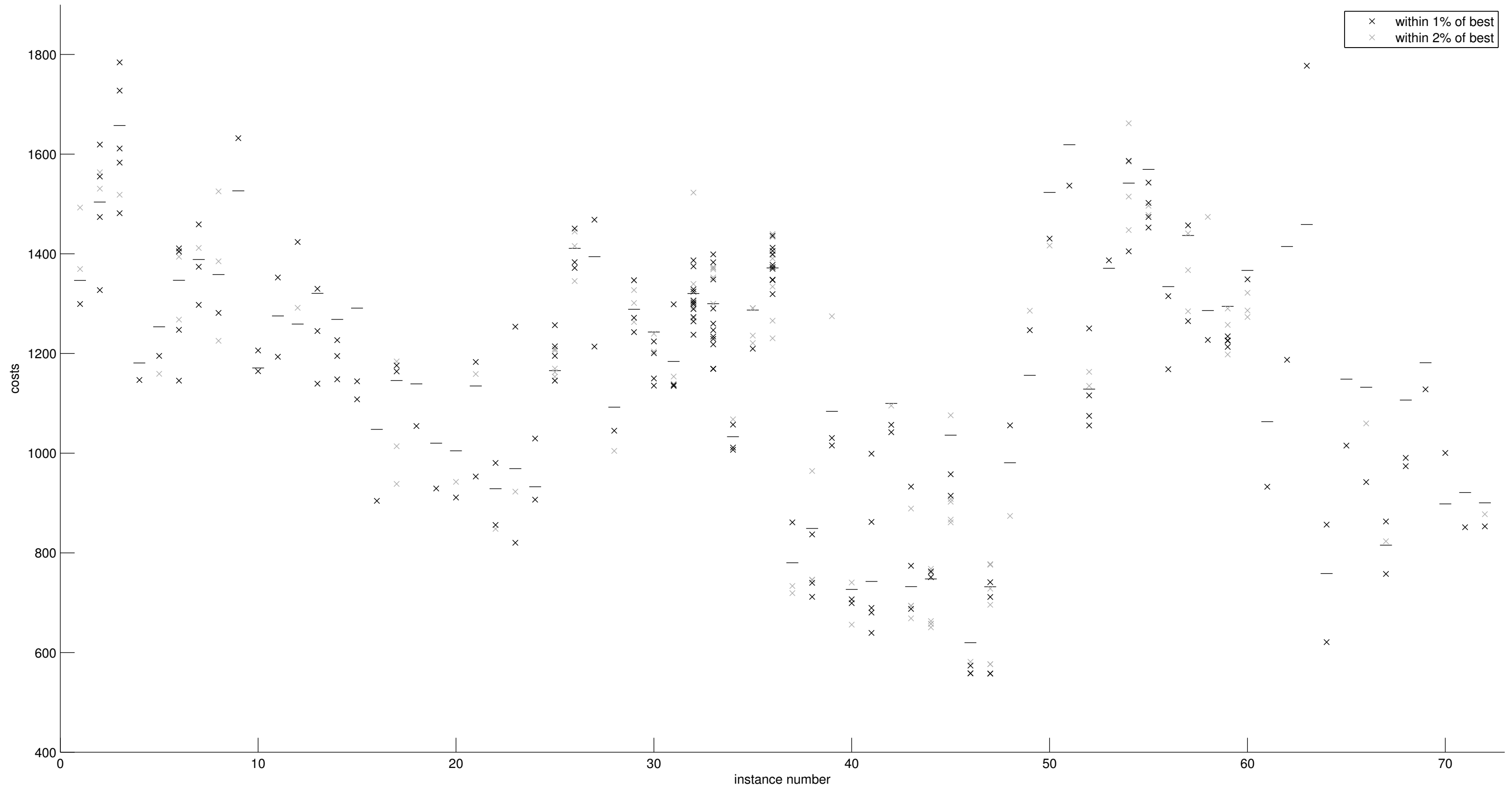

Figure 9: This figure contains information on the importance of the costs of a solution after the construction stage. The costs directly after the route construction stage are shown on the the vertical axis, and the horizontal axis contains the numbers 1 to 72 for the benchmark instances. The horizontal lines mark the average costs of 24 runs after the route construction stage per instance. The black and grey crosses mark the costs of the solutions after the construction stage that eventually became solutions that had total costs within respectively 1% and 2% of the best solution found for that instance after the improvement stage.

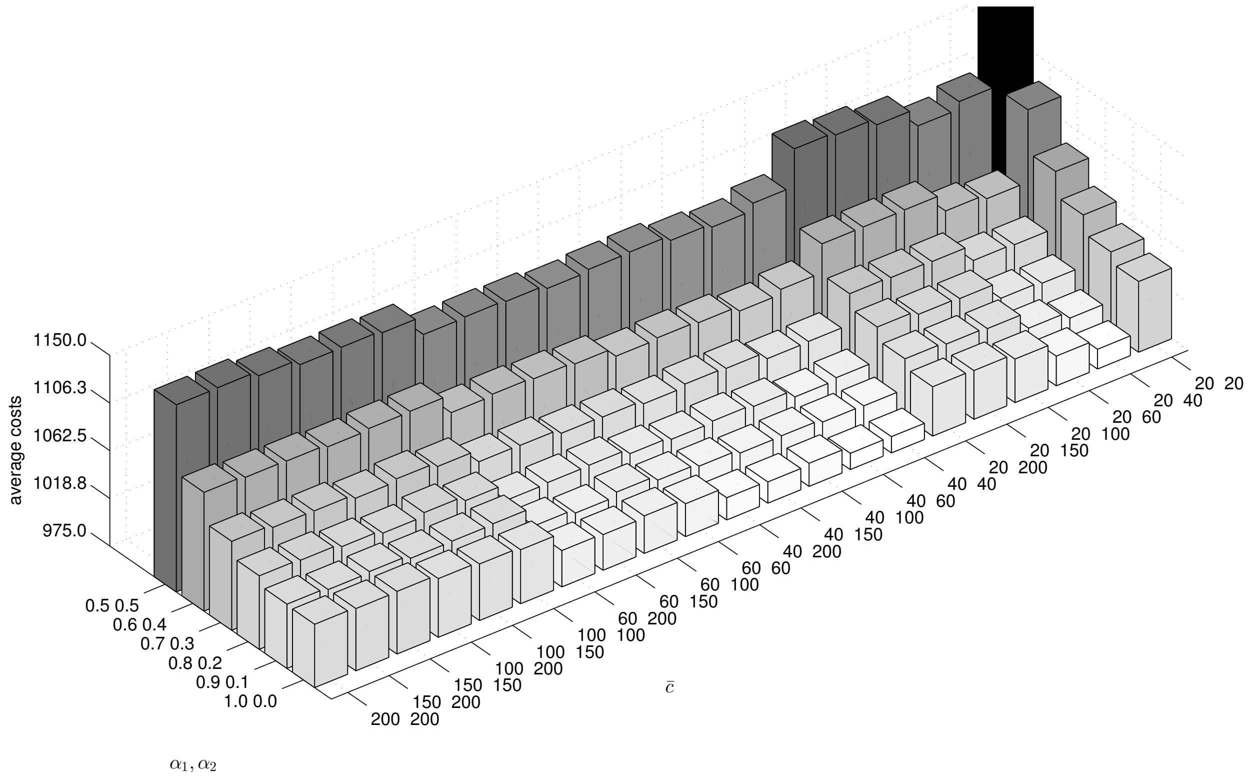

Figure 10: A bar plot which shows the influence of the parameters $\bar{c}$ and α_1 on the average costs of the solutions directly after the route construction stage. The average costs are calculated by taking the average over all 432 best solutions (72 instances with 6 different parameter settings for α_3) created with the specified combination of parameters

Additional testing shows that small changes in the values for $\bar{n}, \bar{k}$ and $\bar{d}$ have no significant effect on the speed of the heuristic or the solution quality. We will therefore leave them at their initial values. Since we have found that solutions with low costs after the route construction stage have a better potential to lead to a good result after the route improvement stage and the route construction stage only accounts for a small percentage of the total runtime of the heuristic, we will increase the value of the parameter *solutions*. However, applying the route improvement procedure to every one of the initial solutions might cause the runtime of the heuristic to become too high. Therefore, after all initial solutions are created, the $(100 - x)\%$ worst solutions in terms of costs are removed and the route improvement procedure is used on the $x\%$ best solutions. We found that choosing $solutions = 50$ and $x = 50\%$ provided a good trade-off between solution quality and speed.

To show the influence of introducing the parameter x, we apply the heuristic with $x = 100\%$ on the 72 benchmark instances with 50 packages, and then perform the following procedure for each instance. We denote the costs of the best solution after all initial 50 solutions have been improved by *best*. We then remove the 25 solutions that have the highest costs after the route construction stage and find the solution with the lowest final costs among the remaining 25 solutions. This would be the objective value if we had only continued with the best 50% of the initial solutions, and we denote the costs of this solution with *obj*. We then calculate $\gamma = \frac{obj-best}{best} \times 100$, which is the percentage increase in objective value caused by the parameter x. The effect of continuing with the best 50% of the solutions can be found in Table 3. The heuristic becomes approximately twice as fast, while still finding solutions with costs within 1% of the *best* in 61 out of 72 instances, and it finds a solution with costs equal to *best* in 49 out of the 72 instances, which is significantly more than half of the number of instances.

	$0 \leq \gamma < 0.5$	$0.5 \leq \gamma < 1$	$1 \leq \gamma < 2$	$2 \leq \gamma < 3$
# instances	56	5	8	3

Table 3: The total number of instances for which solutions are obtained with an objective value within certain percentages (γ) of the best solution known if only the best 50% of the solutions after the route construction stage are used in the route improvement stage.

A closer look at the instances which have γ-values larger than 0.5 learns us that there are mostly instances with long planning horizons and long time windows that suffer from the introduction of the parameter x: Out of the 16 instances with $\gamma > 0.5$, there were 11 instances with long horizons and 11 instances with long time windows. However, these usually are also the instances that have the longest runtimes, so increasing x to obtain better solutions for these instances has large consequences on the runtime of the heuristic. Therefore, we shall use $x = 50\%$ in our heuristic from now on, unless specified otherwise. In conclusion, the following parameter values will be used in the heuristic, unless stated otherwise:

solutions	50
x	50%
$\alpha_1, \alpha_2, \alpha_3$	$\{1, 0, 0.2\}$
$\bar{n}$	4
$\bar{d}$	100
$\bar{c}$	$\{40, 100\}$
$\bar{k}$	5

5.2 Heuristic settings

As mentioned in Section 3.5, there are various options for implementing the route improvement procedure of the heuristic. In order to decide which settings we shall use in our final version of the heuristic, we solve the 72 benchmark instances with 50 parcels with all possible combinations of the proposed settings as mentioned in Section 3.5. In order to decide on the use of the route elimination procedure, we will do this twice: Once with the route elimination stage applied before the route improvement procedure is started, and once without it. Preliminary testing shows that the maximum length of the ejection chains in FDPTW instances is 3. Choosing the parameter value for $\bar{l}$ any higher causes the computational complexity to increase drastically due to the exponential growth in the number of feasible ejection chains. The summarized results of solving the 72 FDPTW benchmark instances with all combinations of heuristic settings can be found in Table 4.

			Exchanges				Exchanges		
			greedy	extensive	both		greedy	extensive	both
Route order	random	obj.	873.5	873.0	867.8		869.4	870.0	865.8
		time (m)	39.3	84.0	120.3		42.1	83.2	122.6
	sorted	obj.	871.4	870.4	868.0		870.8	869.3	868.5
		time (m)	38.7	74.3	114.0		41.6	77.7	116.1
	mixed	obj.	870.7	871.4	865.6		869.6	867.0	865.8
		time (m)	37.8	77.1	116.9		41.4	81.3	121.2
			Without route elimination				With route elimination		

Table 4: The average objective values and the total runtimes of solving the 72 benchmark instances with various settings for the route improvement stage. The results on the left side of the table were obtained by directly using solutions obtained from the route construction stage, while the results on the right side of the table were obtained by first applying the route elimination procedure.

From this table, we learn that the way the routes are ordered before the route improvement stage is of little influence on the average objective values. However, the 'mixed' setting seems to perform slightly better than the other two settings. From this table one can also observe that although using the 'extensive' optimization setting approximately doubles the total runtime of the heuristic with respect to the 'greedy' setting, it does not significantly improve the objective values. Applying both the 'greedy' and 'extensive' settings in the search for profitable exchanges takes about 1.5 times as long as using only the 'extensive' setting, but it does improve the results. Also, the route elimination procedure seems to be beneficial for the quality of the solutions, although the differences in objective values caused by applying the route elimination procedure are not large. In conclusion, combining the 'mixed' route order setting with using both the 'extensive' and the 'greedy' strategy in the search for exchanges gives the best results, and the route elimination procedure has no effect on the average objective values. However, a lot of information is lost by looking only at average or total values. A closer look at the data, which is not reported due to its large size, learns us that it is never the case that one of the combinations of settings performs better on all instances than another combination: It has better performance on several instances, but it also performs worse on some other instances. We have performed research to find out which heuristic settings performed well on instances with certain factors, but this did not provide significant results. From this we conclude that it seems that a robust and well performing heuristic has to use a large variety of the proposed settings, rather than just one. Therefore, we use the proposed alteration from Section 3.6 in our heuristic from now on. This allows the heuristic to both come up with solution on which the route elimination procedure has been applied, as solution on which the route elimination procedure has not been applied. For the route improvement procedure, we use the the mixed route order setting with using both the extensive and the greedy strategy in the search for exchanges.

6 Results

In this section, we shall compare the performance of the formulations for the FDPTW, and we will analyze the performance of the heuristic on both small and large problem instances. All results of the exact formulations are obtained by implementing the formulations in Java using the optimization package CPLEX 12.5. Note that we mentioned that the implementation of the heuristic only runs on a single core of the processor, while CPLEX uses all four of its cores. As explained earlier, the heuristic could ideally become four times as fast by multi-threading it, which would allow a fairer comparison of runtimes between the exact formulations and the heuristic.

6.1 Formulation comparison

For each of our formulations of the FDPTW, we can express the number of constraints and variables as a function of the cardinality of the sets that are used in the formulation. For Formulations I to III, these expressions are shown in Table 5. The number of variables and constraints in a formulation can not be directly related to its performance. However, if those numbers rapidly rise if the size of the problem instance increases, this usually indicates bad performance in terms of runtime. Using this information, one expects Formulation II to have the worst performance, while Formulation III is likely to perform best. The fact that we expect the performance of Formulation II to be the worst is due to the term $|L|^3$ in both the expression for the constraints and the variables, whereas Formulation I and III have $|V||L|^2$ and $2|L|^2$ as the largest term of the polynomial in $|L|$ respectively. Also, the number of constraints of both Formulation I and II can drastically increase if there is a geographical location where many different packages can be delivered, due to the term $\sum_{F \in Z} 2^{|F|}$. Since Formulation III does not contain any subtour elimination constraints, it does not suffer from this large increase in constraints. This, combined with the earlier mentioned fact that the number of constraints and variables of Formulation III grows significantly less fast when $|L|$ rises in comparison to Formulation I and II, raises the expectation that Formulation III will outperform the other two.

Formulation I	constraints	$2|P| + |V|(|L|^2 + 2|L| + \sum_{F \in Z}(2^{|F|} - |F| - 1) + 7$
	variables	$|V|(|L|^2 + |L|) + |L| + 1$
Formulation II	constraints	$2|P| + |L|^3 + 3|L|^2 + 5|L| + \sum_{F \in Z}(2^{|F|} - |F| - 1) + 2$
	variables	$|L|^3 + 2|L|^2 + 2|L| + 1$
Formulation III	constraints	$2|P| + 2|L|^2 + 6|L| + 5$
	variables	$2|L|^2 + 6|L| + 4$

Table 5: The number of constraints and variables per formulation

In order to compare the actual performance of the formulations with each other, we will have them solve a set of benchmark instances. We use scaled down versions of the FDPTW benchmark instances proposed in Section 4 for this purpose, since instances with 50 parcels proved to be far too difficult to solve to optimality within reasonable time. We found that all formulations are capable of solving instances with 12 parcels, so we shall use the set of FDPTW benchmark instances in which there are 12 parcels to be delivered for each instance. We restrict the maximum runtime of solving a single instance to one hour.

A quick summary of the performance of the formulations is given in Table 6.

Clearly, Formulation III outperforms the other two. While Formulation I and II need a total of about 2½ and 3 hours respectively to solve the benchmark instances, Formulation III is done within a total of 3½ minutes. An analysis of the performance per factor of the benchmark instances is given in Tables 7 to 9. Here, the average runtimes of the formulations are shown per factor. For example, from Table 7 one can learn that on average it took Formulation III 2.20 seconds to solve a benchmark instance that had a clustered distribution for the placement of its locations. The values in these tables are as one could expect them to be: The more feasible solutions an instance has, the more difficult it becomes to solve the instance. For example, when nodes are clustered geographically there are likely to be more feasible sequences in which the nodes can be visited, and thus there might be more feasible solutions. The same reasoning can be applied to the other factors. For each formulation, the length of the time windows and the locations per package factor have the largest influence on the runtime. However, Formulation III is more robust against changes in the instances than the other two formulations. For example, while Formulations I and II take respectively 30 and 45 times as long to solve instances with long time windows as instances with short time windows on average, Formulation III only needs 7 times as much time. It is also worth noticing that while Formulations I and II perform better on instances with long horizon lengths than on instances with short horizon lengths, the reverse is true for Formulation III.

A more detailed overview of the individual statistics per instance can be found in Table 14 in Appendix B. In this table we have also included the percentage gap of the optimal solution of the LP-relaxation in comparison to the objective value along with the time it took to solve the LP-relaxation. The reported LP-relaxation runtime is the time it takes CPLEX to solve the original LP-relaxation. However, when CPLEX tries to solve a MIP, it first aplies a presolve procedure in which it eliminates rows and columns and then solves the LP-relaxation of this modified problem. Because of this, for some simple instances it can happen that the reported LP-relaxation time can exceed the MIP solve time. From Table 14 in Appendix B, we learn that Formulation I has the worst LP-bounds, especially in cases where the set Z is large, in which case it often finds a solution with an objective value of 0. This is due to the fact that two separate vehicles are now allowed to make subtours with a flow lower than 1 while still adhering to the subtour elimination constraints. Formulation II has the best LP-bounds, but the LP-relaxation usually takes very long to solve in comparison with the two other formulations. The LP-relaxation of Formulation III has a better trade-off between solution quality and speed than the other two formulations, which is likely to contribute to its superiority.

Formulation	Formulation I	Formulation II	Formulation III
runtime	9955.0	11080.3	212.5

Table 6: Total runtimes in seconds of the three formulations for solving 72 benchmark instances with 12 parcels

factor	average runtime					
GD	7.44	(R)	227.58	(C)	179.77	(RC)
HL	200.51	(S)	76.02	(L)		
LOTW	8.88	(s)	267.65	(l)		
LPP	14.63	(A)	261.90	(B)		
ULTPR	377.39	(a*)	23.49	(b*)	13.91	(c*)

Table 7: Average runtimes in seconds of solving benchmark instances with 12 parcels with Formulation I

factor	average runtime					
GD	40.52	(R)	182.17	(C)	238.99	(RC)
HL	222.37	(S)	85.42	(L)		
LOTW	6.66	(s)	301.13	(l)		
LPP	2.87	(A)	304.92	(B)		
ULTPR	329.80	(a*)	84.55	(b*)	47.33	(c*)

Table 8: Average runtimes in seconds of solving benchmark instances with 12 parcels with Formulation II

factor	average runtime					
GD	1.74	(R)	2.20	(C)	4.92	(RC)
HL	2.25	(S)	3.65	(L)		
LOTW	0.75	(s)	5.16	(l)		
LPP	0.58	(A)	5.32	(B)		
ULTPR	3.13	(a*)	3.29	(b*)	2.43	(c*)

Table 9: Average runtimes in seconds of solving benchmark instances with 12 parcels with Formulation III

6.2 Heuristic results

6.2.1 Performance on small FDPTW benchmark instances

Although the heuristic is created to solve large problem instances and its parameters have been chosen accordingly, it is also interesting to compare the results of the heuristic to the optimal solutions of smaller instances. For this purpose, we use the set of FDPTW benchmark instances with 16 packages to be delivered, which is the highest amount for which Formulation III is able to solve every instance within reasonable time.

Details on the performance of the heuristic for every becnhmark instance can be found in Table 15 in Appendix C. A summary of the performance of the heuristic on these instances can be found in Table 10. The heuristic performs very well on small problem instances. While it took 102 minutes to obtain the optimal solutions with Formulation III, the heuristic only requires 1½ minutes to find solutions for every instance, a decrease of 98.6% in total runtime. This very large decrease in runtime comes at the cost of a slight decrease in solution quality. Overall, the heuristic found solutions that had costs that were on average 0.23% above the optimal solution. The heuristic finds solutions with optimal costs for a total of 65 out of 72 instances. For the 7 instances for which the heuristic finds sub-optimal solutions, the mean percentage gap with respect to the optimal solution is 2.3%, and the largest percentage gap is 6.0%. This was the case for instance RCLlBc, where the optimal costs are 369.56 and the heuristic finds a solution with total costs of 391.65. The reason the heuristic performs not very well on this instance is that the optimal instance consist of a single route. The heuristic does not find this route during the route construction stage, which almost always results in two nodes being visited by a separate route. The route elimination stage is in this case often not able to eliminate this small route, and the steepest descent nature of the improvement stage also prevents finding the optimal solution.

	time	obj.		time	obj.		time	obj.	
GD	-95.2%	+.4%	(R)	-98.6%	+.0%	(C)	-98.3%	+.2%	(RC)
HL	-98.5%	+.0%	(S)	-97.7%	+.4%	(L)			
LOTW	-76.7%	+.1%	(s)	-98.5%	+.3%	(l)			
LPP	-79.8%	+.1%	(A)	-98.6%	+.3%	(B)			
ULTPR	-98.0%	+.0%	(a*)	-98.5%	+.2%	(b*)	-95.3%	+.4%	(c*)

Table 10: Comparison of the objective values and runtimes of solutions obtained with the heuristic with respect to the exact solutions of Formulation III on instances with 16 parcels.

Something worth noticing is the relatively low values for percentage decrease in runtime for instances with short time windows and option A for the locations per package factor in Table 10. However, this is not because of the fact that the heuristic's performance is worse on these instances, but because of the fact that the exact formulation is usually able to solve these instances within a very small amount of time. The same can be said for some very high values in the column that contains the percentage difference in runtimes between the exact solution method and our heuristic of the table in Appendix C, where the heuristic's runtime is higher than the exact formulation's runtime. However, this is something that only occurs for some instances with a small amount of packages (in this case 16), for which the heuristic will rarely be applied in practice since those instances can quickly be solved to optimality. For instances with a higher amount of packages, for which the heuristic has been developed, the runtime of the exact formulation will become substantially higher than the heuristic's runtime.

From Appendix C we also find that the runtime of the heuristic is very robust in comparison to the runtime of Formulation III: While the runtime of the exact solution method fluctuates anywhere between 200 and 1.5 million milliseconds, the heuristic always returns its best solution within 150 to 9500 milliseconds.

6.2.2 Performance on FDPTW benchmark instances with 50 parcels

In order to analyze the heuristic's performance on large instances, for which the heuristic has been developed, we have applied the heuristic on the 72 benchmark instances with 50 packages. The results are summarized in Table 11, and extensive results can be found in Table 16 in Appendix B. We find that our idea of editing the way the route elimination procedure is used to create a more robust heuristic is successful. The average objective value over the 72 instances has been reduced to 864.0, which is lower than all average values found in Section 5.2. The heuristic does so in a total of 117 minutes.

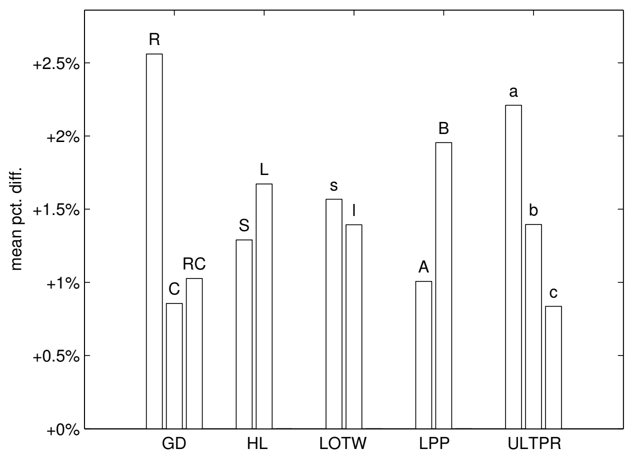

Figure 11: The average percentage gap between the best know solution for an instance and the solution that was found by the final heuristic per factor.

It is very difficult to draw conclusions on the solution quality of the heuristic on large instances, since we have no information on the optimal solutions. However, we are able to draw some conclusions on the influence of several factors on the difficulty of the problem if we make the assumption that the heuristic finds solutions with costs that are close to the costs of the optimal solution. This assumption can be justified by recalling the good performance of the heuristic on smaller instances. Although this does not necessarily guarantee a good performance on large problem instances, this does strengthen our assumption. Also, we can use the information obtained in Section 5.2 to assess the performance of the heuristic, where we obtained 18 solutions for every instance in our search for the best combination of heuristic settings in a total of 23 hours and 50 minutes. For every instance, we can compare the best of these 18 solutions with the solution that the final version of the heuristic finds. We can then calculate the average gap of the heuristic's solutions with the best solution known for each factor of the benchmark instances. The average gaps per factor of the benchmark instances are displayed in Figure 11.

The average gap over all instances between the heuristic's solution and the best known solution is 1.48%. All percentage differences are below 5% except for the instances RSsBa and RLsBa, where the gaps are 11.65% and 14.75% respectively. In the first case, the best known solution is 667.3, while the solution found has total costs of 744.9, and in the second case the best known costs are 566.45 while the costs found are 649.7. In both cases, almost all of the 18 solutions found during the search for the best heuristic settings are solutions with a gap larger than 10% in comparison to the best solution known, and another solution with costs within 5% of the best known solution is only encountered once. It is difficult to point to a single cause for this behavior, but a likely explanation is that this is due to the fact that these instances have a lot of local optima, from which it is difficult to get to a better local optimum. This in turn is caused by the fact that our route improvement procedure only allows the removal of one node from a route at a time. However, since a lot of packages are delivered at the same location in an instance with a low unique locations to package ratio, especially when combined with the second option for the locations per package factor, the exchange of a single node between routes does often not lead to a reduction in costs. This is because the vehicle still has to drive to the location where the parcel was previously delivered to deliver the other packages at that location, and therefore the considered exchange is not executed.

We shall now consider the influence of the factors of the benchmark instances on the problem and the heuristic's performance one by one, using Table 11 and Figure 11.

	time (s)	obj.	time (s)	obj.	time (s)	obj.
GD	58.6	873.6 (R)	90.5	848.4 (C)	60.9	896.7 (RC)
HL	6.6	1045.8 (S)	133.5	700.0 (L)		
LOTW	36.9	891.7 (s)	103.2	854.2 (l)		
LPP	46.4	946.2 (A)	93.7	799.7 (B)		
ULTPR	75.0	811.7 (a)	63.2	874.0 (b)	71.8	933.0 (c)

Table 11: average objective values and average runtimes of the heuristic on instances with 50 parcels per factor of the benchmark instances

Geographical distribution The influence of the geographical distribution is as one would expect. The more clustered the locations are, the higher the total runtime of the heuristic becomes. This can be explained by noticing that this increases the computational complexity of the route improvement procedure: A node that is located within a cluster that is considered for removal can usually be replaced with more nodes than a node that is not in a cluster, since it has a larger amount of nearby nodes. The performance of the heuristic is however more consistent on instances that are clustered than on instances in which the locations are placed randomly.

Horizon length Another thing that should be noticed is the significant influence of the length of the horizon on the objective value. As one expects, the costs for instances with short horizons are much higher than the costs for instances with long horizons, since more vehicles are needed and each vehicle has to depart from and return to the depot. In the case of short horizons, the solutions usually contain 8 or 9 vehicles, which means that a vehicle can carry on average between 5 or 7 packages. In the case of long horizons a vehicle carries between 12 and 17 packages on average. This increase has a very large influence on the runtime of the heuristic. An instance with a long planning horizon takes on average 20 times as long to solve as an instance with a short horizon. This large difference is explained by looking at the structure of the route improvement procedure. With long planning horizons, there are a lot more nodes that are considered for removal from a route before an exchange is executed. Also, with long planning horizons there are more feasible insertions possible for each node, since nodes can be inserted into routes that are further away. Although there is a large difference in the required time to solve an instance between short and long planning horizons, the average gap with respect to the best known solution does not seem to differ significantly.

Length of time windows The length of the time windows affects the problem's computational complexity in a similar way as the geographical distribution and the horizon length. With short time windows, there are less feasible insertion places and exchanges for each node than is the case when the time windows are longer. This results in slightly more expensive solutions but a much shorter runtime for instances with short time windows. However, the heuristics performance with respect to the best known solution is similar in both cases.

Locations per package Possibly the most important conclusion we can draw is the fact that in these theoretical benchmark instances a significant reduction in costs can be achieved by allowing parcels to be delivered at multiple locations with different time windows, by looking at the 'locations per package'-row of Table 11. In practical applications, this indicates that companies have the potential to increase profits by providing customers the possibility to specify multiple delivery locations. However, we can make no statement regarding the size of the savings in transport costs in using this policy. This is both due to the theoretical nature of our benchmark instances and due to the fact that the potential reduction in costs is heavily dependent on a lot of practical factors, such as the width of the customer's time windows, the number of potential delivery locations a customer is allowed to specify and the distance between those locations. We will further investigate the size of the potential savings by solving the urban benchmark instances in Section 6.2.4. As mentioned before, the performance with respect to the best known solution becomes worse for instances where a lot of packages can be delivered at multiple locations, which is likely caused by the fact that the route improvement procedure has difficulty with escaping from local optima in these cases.

Unique locations to package ratio The unique locations to package ratio does not seem to have a significant effect on the runtime of the heuristic. However, instances for which this ratio is low tend to have lower costs than instances for which this ratio is high, since packages have a higher chance of sharing a geographical delivery location in the first case. This provides the possibility to deliver the packages simultaneously at a location, which can lead to solutions with lower costs. The effect on the performance with respect to the best known solution is similar to that of the locations per package factor, since a lower unique locations per package ratio causes more packages to be delivered at the same location in a solution.

6.2.3 Performance on FDPTW benchmark instances with 50 parcels with a target runtime per instance

. Until now, we have used the same parameter values for solving each instance of the FDPTW benchmark instances. However, due to the difference in the complexity of the various instances, this causes the runtimes to lie anywhere in the interval of 3 to 470 seconds, which causes the total time needed to solve all 72 instances to be lower than two hours. In practice, there may be more time to solve an instance. A good heuristic should have the property that it is able to provide better solutions when it is allowed to have a longer runtime, provided that the optimal solutions have not been found yet. To verify that our heuristic has this property we give the heuristic a common maximum runtime for each instance. We assume that thirty minutes is a reasonable runtime for solving instances with 50 parcels, so we will use that as the target runtime. Since the route improvement procedure is the most time consuming stage of the heuristic, one can make good estimations of the runtime by looking at the number of solutions that should undergo this procedure. Since we already know how long it takes to improve 25 solutions for each instance from Section 6.2.2, we can make estimates for the parameters *solutions* and x for every instance to make sure the runtime will be close to thirty minutes. First, we determine the maximum number of solutions that can be created if $x = 50\%$ is used for this instance with the following equation:

$$y = \frac{1800}{time} * 50 \tag{56}$$

Here, *time* is the time it took the heuristic to solve the instance with the parameter values $solutions = 50$ and $x = 50\%$, and we assume the runtime of route construction and elimination stages to be negligible. However, the route construction stage has a limited amount of unique solutions it can provide and it makes little sense to create a lot of duplicate solutions. We find that in general, the number of duplicates in the initial solution pool is still low when creating 500 initial solutions. Therefore, we use the following decision rule to determine the final parameter values for the instance:

$$\begin{aligned} solutions = y, x = 50\% \quad & \text{if} \quad y \leq 500 \\ solutions = 500, x = 50\% + \frac{y - 500}{500} \times 50\% \quad & \text{if} \quad 500 < y < 1000 \\ solutions = 500, x = 100\% \quad & \text{if} \quad y \geq 1000 \end{aligned} \tag{57}$$

The results of applying the heuristic with the parameter values obtained with the method described above can be found in Table 17 in Appendix B. The total time needed to solve the 72 FDPTW benchmark instances is 19.5 hours, slightly more than an average of fifteen minutes per instance. The average objective value of the solutions over all 72 instances is 852.9, a 1.3% decrease in costs with respect to the average costs of 864.0 obtained with the fixed parameter values $solutions = 50$ and $x = 50\%$. The heuristic finds solutions with costs equal to or even below the best known costs for each instance much more often, with the largest percentage gap being 2.0%. The performance of the heuristic on the instances on which the performance of the heuristic was very bad in Section 6.2.2, RSsBa and RLsBa, has also been improved drastically. Where the percentage gaps with respect to the costs of the best known solution were +11.65% and +14.75% respectively with the default settings for the heuristic, the heuristic now finds solutions with costs 0.5% lower and 2% higher than the best known solutions for these two instances. The average percentage gap over all 72 instances has been improved from +1.48% to +0.04%.

Of course, it is usually difficult to obtain good estimates for the parameters *solutions* and x, since the instance might not have been solved before and there are therefore no total runtimes known. However, estimates can easily be obtained by creating a very small set of initial solutions and solving those to get an indication of the time it takes to improve a solution. The structure of the heuristic also makes a more dynamic implementation possible. One could first create a large set of solutions in a very short amount of time. Then, the solutions can be improved one by one, and the heuristic can be stopped once a certain time has passed. The final solution of the heuristic is then simply the best solution encountered up until that time.

6.2.4 Performance on the urban benchmark instances

In order to strengthen the conclusion that giving customers the option to specify their whereabouts during the day can lead to a reduction in cost for the supplier, we will now use our heuristic to solve the urban instances created in Section 4.3. Since the driving distances in the urban instances are different than those in the FDPTW benchmark instances, we have to adjust some of the parameter values found in Section 5.1. Since the length of the area of the urban instances is $\frac{1}{2}$ the length of the area of the theoretical benchmark instances, we multiply the values for the parameters $\bar{c}$ and $\bar{d}$ by $\frac{1}{2}$ to account for this difference. Furthermore, we found a way of improving the performance of our heuristic in solving instances in which there are a few locations where a lot of parcels can be delivered, while very few parcels can be delivered at most others. This is the case with pick-up points in the urban instances. In these cases, the human eye can easily see that it is very likely to be beneficial to deliver parcels at a location where a lot of parcels can be delivered. However, the heuristic does not recognize this, and it is possible that a route constructed in the route construction stage does not visit such a location, but rather drives to every single location in the entire suburb. Due to its steepest descent nature, it is unlikely that the route improvement stage is able to find this shortcoming, since driving to the location where a lot of parcels can be delivered is likely to increase the total costs of the solution if only a single parcel is delivered there.

In instances with the aforementioned property, this shortcoming can be easily be overcome by adding a parameter α_4 in the route construction stage, and modifying the cost function, Equation (52), by adding a term $-\alpha_4 g_u$. Here, g_u is the total number of parcels that can be delivered at node u. The weight α_4 should be relatively low, we find that choosing a value of 3/8 provides good results. In doing so, the heuristic is able to prefer the pick-up points above other nodes that lie very close in the route construction stage, while not giving them to much priority.

Next to the modification of the parameter values to account for the difference in distances and the introduction of the parameter α_4, we make two slight alterations to the heuristic. Firstly, the route elimination proves to become computationally too complex due to the exponential growth in the possible ejection chains in some instances and therefore we decide not to use it. Secondly, we are able to speed up the heuristic a lot by not considering parcels that are delivered at a pick-up point or office for elimination from a route during the route improvement stage, since it is very unlikely that this will lead to a reduction in costs.

The results of solving the urban instances with the slightly modified heuristic can be found in Table 12. We have included the percentage savings in costs of the versions A, B and C with respect to version D of each instance in a separate table for clarity, which is Table 13. When we compare row A with row D, it becomes immediately clear that the potential savings in costs caused by providing multiple

	instance	1	2	3	4	5
	p_{office}	0.3	0.2	0.5	0.7	0.3
	$p_{pick-up}$	0.2	0.3	0.5	0.3	0.7
A	obj.	483.15	515.58	436.75	425.30	412.24
	vehicles	4	5	4	4	4
	time (m)	106.8	52.6	170.6	139.6	92.4
	home	67/100	74/100	25/100	34/100	29/100
	office	21/30	12/16	20/52	51/74	9/36
	pick-up	12/15	14/23	55/58	15/25	62/64
B	obj.	550.58	560.15	531.19	558.42	444.32
	vehicles	5	5	6	5	5
	time (m)	28.7	15.1	36.0	15.6	33.7
	home	88/100	79/100	43/100	80/100	37/100
	office	0/0	0/0	0/0	0/0	0/0
	pick-up	12/15	21/23	57/58	20/25	63/64
C	obj.	483.23	516.44	489.95	437.31	510.40
	vehicles	4	5	4	4	5
	time (m)	72.6	32.95	58.3	112.2	40.8
	home	73/100	88/100	58/100	43/100	71/100
	office	27/30	12/16	42/52	57/74	29/36
	pick-up	0/0	0/0	0/0	0/0	0/0
D	obj.	566.86	545.59	544.46	573.35	646.20
	vehicles	5	5	5	6	6
	time (m)	18.7	12.8	10.6	11.7	10.3
	home	100/100	100/100	100/100	100/100	100/100
	office	0/0	0/0	0/0	0/0	0/0
	pick-up	0/0	0/0	0/0	0/0	0/0

Table 12: Results of solving the set of urban instances with the heuristic. For each instance, the objective value, the number of vehicles in the solution and the runtime in minutes are reported, along with the total number of parcels that was delivered at home, at an office or at a pick-up point as a fraction of the total number of packages that could be delivered at home, at an office or at a pick-up point

potential delivery locations are very high, reaching values between 5% and 36% in these instances. In the first instance, a total savings in costs of 14.8% can already be realized if just 30% of the customers allow their parcel to be delivered at an office in the city center. The best solutions for versions A and D of the instance with $p_{office} = 0.3$ and $P_{pick-up} = 0.7$ are displayed in Figures 12 and 13 in Appendix F. The total costs of the solution in Figure 12, where there are no pick-up points or offices, are 646.20. The total costs of the solution in Figure 13 are 412.40, a decrease in total costs of 36.2% as a result of the introduction of the offices and pick-up points. What also becomes clear from these figures is that notable savings are obtained by eliminating the need to visit some poorly located nodes. In Figures 12 and 13, these nodes are the top-left node and the nodes between the top-right and bottom-right clusters. The fact that these nodes do not have to be visited likely contributes to the fact that there are less vehicles needed in the solution. This is also a credible

instance	1	2	3	4	5
p_{office}	0.3	0.2	0.5	0.7	0.3
$p_{pick-up}$	0.2	0.3	0.5	0.3	0.7
A	-14.8%	-5.5%	-19.8%	-25.8%	-36.2%
B	-2.9%	+2.7%	-2.4%	-2.6%	-31.2%
C	-14.8%	-5.3%	-10.0%	-23.7%	-21.0%

Table 13: The percentage savings that the heuristic was able to achieve for the various versions of each instance with respect to version D, where all homes have to be visited.

explanation for the large differences in the reported savings in Table 13: It is not only the amount of parcels that can be delivered at an office or pick-up point that counts, but moreover which parcels have this opportunity.

Overall, the heuristic seems to be good at making use of the pick-up points and offices in order to reduce the total costs. However, in version B of the second instance, the heuristic's preference for making use of the pick-up points, which we have provided by adding the parameter α_4, actually causes a slightly worse performance. In this case, better solutions can be acquired by not making use of the pick-up points at all. This is likely to be a shortcoming of the introduction of the parameter α_4. By looking at Table 13, we find that in general, larger savings can be achieved by the introduction of a relatively small amount of customers that allow their parcels to be delivered at offices than by the introduction of a small amount of customers that allow their parcels to be delivered at pick-up points. It is difficult to find a single cause for this phenomenon, since we cannot say decisively whether the cause lies in the structure of the instances or if the performance of the heuristic is bad on some of the instances. However, we found a possible explanation by examining the solutions more thoroughly. It seems to be the case that the cause lies in the design of the instances. Whenever there is a small amount of offices available, the section above and to the right of the depot of the instance can usually be serviced by a single vehicle since a few parcels can be delivered by another vehicle at the offices. However, whenever there is a small to moderate amount of parcels that can be delivered at pick-up points, two vehicles are needed to service that same section due to the capacity constraints, since there are two pick-up points.

From the results of this section, we can conclude that there clearly is a potential to save costs by introducing the possibility to specify multiple delivery locations to customers. However, it is difficult to draw solid conclusions on the size of the savings that could be achieved in doing so. This is partially because we have no information on the optimal solutions of the instances on which we base our conclusions and partially because we find that the savings that can be achieved are very dependent on the problem instance.

7 Conclusion and Discussion

In this thesis, we have developed exact formulations and a heuristic in order to solve instances of the Flexible Delivery Problem with Time Windows (FDPTW). We have developed various sets of benchmark instances to analyze the performance of these formulations and heuristics, based on which we will make our concluding remarks in this section.

7.1 On the quality of the heuristic

As mentioned in our introduction, we think that the following set of properties constitute a set which accurately measures the quality of a heuristic: *Quality of the solutions, runtime, simplicity, flexibility and robustness* (Barr et al. (1992), Cordeau et al. (2002)). Since to the best of our knowledge there are no other heuristics for the FDPTW with which we can compare our results, it is not easy to objectively measure the degree to which our heuristic satisfies some of these criteria. However, we will attempt to do so based on the results of our analysis in Section 6.

In solving the 72 benchmark instances with 16 parcels, the heuristic finds solutions that have costs that are on average 0.23% above the costs of the optimal solutions within 2% of the runtime that the exact formulation needed. The heuristic also performs well on the 72 FDPTW instances with 50 parcels, finding solutions with costs that are only 0.04% above the best known solutions with an average runtime of 15 minutes per instance. Based on these results, we think that the heuristic provides a decent trade-off between *quality of the solutions* and *runtime*.

Due to the easy to understand route construction and improvement structure of the heuristic, it also has a high degree of *simplicity*. However, the implementation of the search for Or-exchanges is quite complex.

Cordeau et al. (2002) state that a good VRP heuristic should also be flexible, i.e. it should be flexible enough to accomodate the various side-constraints encountered in a majority of real-life applications. Using this definition, the heuristic scores high in terms of *flexibility* due to its simple route construction and route improvement structure. It is easy to imagine how one could implement for example maximum driving-times for a vehicle or how one could enforce certain packages to be delivered by certain vehicles (e.g. packages that should be kept at a certain temperature) by modifying the functions that assert whether the insertion of a node into a route is feasible.

Robustness is the criterion on which the heuristic would score the lowest. Although the heuristic found solutions that had costs that were within 2% of the costs of the best known solution for each instance of the FDPTW benchmark instances in Section 6.2.3, the runtime can vary greatly between instances with the same number of packages to be delivered. Partially, this is caused by the structure of FDPTW instances; an instance with 100 packages to be delivered can have 100 potential delivery locations, but it can also have 300 delivery locations. The route improvement stage of the heuristic is the most time consuming stage, while the runtime of the route elimination procedure is usually negligible. The length of the planning horizon is the factor that has the largest influence on the runtime of the heuristic. This is partially because longer planning horizons cause nodes to have more feasible insertion places in a solution, and partially because a vehicle can carry more parcels in the case of a long planning horizon, which causes the route improvement procedure to become slower. Another factor that influences the runtime strongly is the length of the time windows. Again, this is likely to be because of the fact that a node has more feasible insertion places in the case of long time windows.

7.2 Discussion and further research

In this thesis, we have laid some solid groundwork for further research in developing solution methods for the FDPTW. In Section 4.3 we have shown that large savings in costs can be achieved by giving customers the opportunity to specify multiple delivery locations, so the development of good solution methods could have a large impact on various potential practical applications of the FDPTW. However, more research is needed to obtain more solid conclusions regarding the size of these savings. This might prove to be very difficult, since there are a lot of factors that influence this, such as the amount of delivery locations per parcel, the locations of these delivery locations, but also the amount of parcels per delivery location. The size of the potential savings is thus very dependent on the application for which it is used. Therefore, we think that research in this area should focus on real world data. For example, a package delivery company could add a short survey to their deliveries, in which customers could specify whether they would make use of the possibility to specify multiple delivery locations, and in which they are asked to provide the addresses of these locations and their estimated time windows. The solutions of the instances without these additional locations can then be compared to the solutions of the instances where all additional delivery locations are added to find out if the savings in costs are significantly large.

Future research regarding solution methods could focus on the development of new heuristics, but there are also still various ways in which the current heuristic could potentially improve which would be interesting to investigate. One important aspect in which improvement is possible is the speed of the heuristic on large instances. The largest instances that we have solved with the heuristic contained 100 parcels to be delivered, with an average maximum of 25 parcels per vehicle. The heuristic can solve the most difficult of the instances that we created within three hours. When multi-threaded, the heuristic should be able to solve this instance in 45 minutes, which can be considered a reasonable time for an instance of that size. Depending on the application for which the heuristic is used however, the number of packages and the number of packages per vehicle may be higher.

If one would want to make the heuristic more suitable for solving very large instances, there are various ways to decrease the runtime of the heuristic. The main focus of these approaches should be to decrease the runtime of the route improvement procedure, since that is currently the most time-consuming stage. One way in which this could be achieved is by applying some form of preprocessing to reduce the computational complexity of the route improvement stage. For example, one could identify nodes that have to be visited in any solution, because the parcel only has a single delivery location. One could then identify any nodes that lie within close proximity and have time windows such that these nodes can be visited in quick succession of each other. The optimal route within this cluster of nodes can be determined, and this cluster can then be used as a single node within the problem, thereby removing all other nodes for any of the parcels that are delivered within this cluster.

Another way of speeding up the route improvement stage in order to make it more suitable for solving very large instances would be to decrease the number of ejection chains considered for each node before an exchange is executed. Currently, we only look at the costs of the finished ejection chains, but one could also incorporate the costs of the ejection chain after two ejections and one insertion. If the costs of the ejection chain are positive at this point, there is no use finishing this ejection chain since it will not lead to a reduction in costs.

Another area for investigation that might prove to be rewarding is trying to make the route improvement procedure more 'intelligent' by making it recognize parcels that are delivered at the same geographical location. Currently, the route improvement procedure does not use this information, and it treats every node on a route similarly. However, the route improvement procedure could likely use information on the delivery of multiple parcels at a location to its advantage. For example, when a very large quantity of parcels is delivered at a single location, there is little use in trying to relocate nodes from this location, as we mentioned in Section 6.2.4. Also, if just a few parcels are delivered at a single location, these nodes could be merged into a single node with adjusted time windows and the cumulative weight of the parcels. This node could then be treated as any other node by the route improvement procedure.

References

R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An Exact Algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52(5):723–738, 2004.

R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218:1–6, 2012.

M. Balinski and R. Quandt. On an integer program for a delivery problem. *Operations Research*, 12:300–304, 1964.

R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende, and W.R. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1:9–32, 1992.

O. Bräysy. Fast local searches for the vehicle routing problem with time windows. *INFOR*, 40:319–330, 2002.

O. Bräysy. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368, 2003.

O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part I: route construction and local search algorithms. *Transportation Science*, 39(1): 104–118, 2005a.

O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005b.

J.F. Cordeau, M. Gendreau, G. Laporte, J.Y. Potving, and J. Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53: 512–522, 2002.

G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science*, 6:80, 1959.

G. Finke, A. Claus, and E. Gunn. A two-commodity network flow approach to the traveling salesman problem. *Congress. Numerantium*, 41:167–178, 1984.

M.L. Fisher, K.O. Jörnsten, and O.B.G. Madsen. Vehicle routing with time windows: Two optimization algorithms. *Operations Research*, 45(3):448–492, 1997.

W.W. Garvin, H.W. Crandall, J.B. John, and R.A. Spellman. Applications of vehicle routing in the oil industry. *Management Science*, 3:407–430, 1957.

B. Gavish and S.C. Graves. The travelling salesman problem and related problems. Working Paper 7905, Graduate School of Management, University of Rochester, Rochester, NY, 1979.

B. Gavish and S.C. Graves. Scheduling and routing in transportation distribution systems: Formulations and new relaxations. Working paper, Graduate School of Management, University of Rochester, Rochester, NY, 1982.

R.A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18–21, 1973.

I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking*. PhD thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1976.

M.W.P. Savelsbergh. The vehicle routing problem with time windows: minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154, 1992.

M.M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35:234–265, 1987.

M.M. Solomon and J. Desrosiers. Time window constrained routing and scheduling problems. *Transportation Science*, 22:1–13, 1988.

Thuiswinkel Marktmonitor. Growth online shopping, 2013. URL http://www.thuiswinkel.org/groei-online-winkelen.

U.S. Department of Commerce. Quarterly retail e-commerce sales, 2013. URL https://www.census.gov/retail/mrts/www/data/pdf/ec_current.pdf.

A Proof of the concatenation theorem

Concatenation Theorem If two feasible paths $(r_i, \ldots r_j)$ and $(r_k, \ldots r_h)$ with associated forward time slacks $F_{r_i}^{(r_i,\ldots,r_j)}$ and $F_{r_k}^{(r_k,\ldots,r_h)}$ for the first vertices are concatenated, the forward time slack for the first vertex of the resulting path is given by:

$$F_{r_i}^{(r_i,\ldots,r_j,r_k,\ldots,r_h)} = \min\{F_{r_i}^{(r_i,\ldots,r_j)}, F_{r_k}^{(r_k,\ldots,r_h)} + \sum_{i<p\leq j} W_{r_p} + D_{r_k} - (D_{r_j} + t_{r_j,r_k} + s_{r_k})\} \tag{58}$$

Proof:

$$\begin{aligned}
F_{r_i}^{(r_i,\ldots,r_j,r_k,\ldots,r_h)} &= \min\{\min_{i\leq m\leq j}\{l_{r_m} - (D_{r_i} + \sum_{i\leq p<j} t_{r_p,r_{p+1}} + \sum_{i<p\leq j} s_{r_p})\}, \\
&\min_{k\leq m\leq h}\{l_{r_m} - (D_{r_i} + \sum_{i\leq p<j} t_{r_p,r_{p+1}} + \sum_{i<p\leq j} s_{r_p} + t_{r_j,r_k} + \sum_{k\leq p<m} t_{r_p,r_{p+1}} + \sum_{k\leq p\leq m} s_{r_p})\}\} \\
&= \min\{F_{r_i}^{(r_i,\ldots,r_j)}, \min_{k\leq m\leq h}\{l_{r_m} - (D_{r_k} + \sum_{k\leq p<m} t_{r_p,r_{p+1}} + \sum_{k<p\leq m} s_{r_p})\} \\
&+ D_{r_k} - t_{r_j,r_k} - s_{r_k} - (D_{r_i} + \sum_{i\leq p<j} t_{r_p,r_{p+1}} + \sum_{i<p\leq j} s_{r_p})\} \\
&= \min\{F_{r_i}^{(r_i,\ldots,r_j)}, \min_{k\leq m\leq h}\{l_{r_m} - (D_{r_k} + \sum_{k\leq p<m} t_{r_p,r_{p+1}} + \sum_{k<p\leq m} s_{r_p})\} \\
&+ D_{r_k} - t_{r_j,r_k} - s_{r_k} - (D_{r_j} - \sum_{i<p\leq j} W_{r_p})\} \\
&= \min\{F_{r_i}^{(r_i,\ldots,r_j)}, F_{r_k}^{(r_k,\ldots,r_h)} + D_{r_k} - (D_{r_j} + t_{r_j,r_k} + s_{r_k}) + \sum_{i<p\leq j} W_{r_p}\}
\end{aligned}$$

B Formulation comparison table

	Formulation I				Formulation II				Formulation III			
instance	obj.	time	LP gap	LP time	obj.	time	LP gap	LP time	obj.	time	LP gap	LP time
RSsAa	478.68	109	73.6%	4	478.68	107	27.9%	342	478.68	152	47.2%	17
RSsAb	326.62	188	71.7%	5	326.62	127	15.1%	410	326.62	63	47.8%	5
RSsAc	516.34	107	45.0%	5	516.34	460	26.9%	552	516.34	143	27.5%	6
RSsBa	252.37	348	68.7%	7	252.37	5353	34.0%	1234	252.37	773	32.9%	12
RSsBb	369.98	8826	80.4%	7	369.98	13364	24.3%	1363	369.98	993	41.0%	12
RSsBc	301.89	308	82.7%	14	301.89	1176	22.2%	1307	301.89	351	55.8%	11
RSlAa	438.00	47507	72.3%	4	438.00	3794	17.2%	326	438.00	1895	39.1%	5
RSlAb	297.99	859	45.2%	6	297.99	931	7.7%	326	297.99	148	28.2%	5
RSlAc	405.58	557	66.3%	4	405.58	1240	29.9%	241	405.58	346	37.9%	5
RSlBa	245.78	42398	97.4%	14	245.78	597811	18.2%	1334	245.78	9489	44.6%	12
RSlBb	236.81	32504	77.4%	17	236.81	9146	6.8%	1397	236.81	2584	28.6%	11
RSlBc	299.26	27838	75.3%	12	299.26	151936	14.4%	1338	299.26	3420	35.0%	11
RLsAa	253.55	950	100.0%	4	253.55	162	14.4%	324	253.55	545	82.9%	6
RLsAb	445.63	112	61.5%	2	445.63	383	27.0%	222	445.63	137	53.5%	4
RLsAc	470.51	21	48.4%	2	470.51	92	18.7%	181	470.51	41	42.0%	5
RLsBa	245.59	146	98.7%	6	245.59	2475	9.8%	1382	245.59	1238	80.9%	10
RLsBb	358.38	245	70.5%	5	358.38	1241	21.1%	1342	358.38	291	62.2%	13
RLsBc	382.86	160	55.6%	4	382.86	2040	24.5%	885	382.86	481	49.3%	11
RLlAa	386.74	125	68.8%	4	386.74	760	11.9%	334	386.74	255	53.4%	5
RLlAb	353.25	3550	61.5%	2	353.25	2219	8.1%	250	353.25	1066	52.7%	5
RLlAc	324.09	243	41.2%	1	324.09	346	5.9%	182	324.09	265	38.2%	5
RLlBa	282.72	8394	91.4%	5	282.72	79212	7.3%	1466	282.72	5437	79.1%	12
RLlBb	254.37	2492	86.0%	3	254.37	95251	13.7%	1405	254.37	10199	72.3%	14
RLlBc	259.93	581	74.5%	3	259.93	2754	13.1%	1185	259.93	1369	65.6%	13
CSsAa	374.36	1071	89.8%	4	374.36	552	26.3%	294	374.36	123	30.4%	5
CSsAb	469.00	60	66.7%	2	469.00	74	27.2%	174	469.00	39	34.5%	3
CSsAc	580.64	438	70.4%	3	580.64	596	31.8%	981	580.64	85	36.5%	4
CSsBa	221.19	36351	100.0%	24	221.19	1348	1.8%	2128	221.19	663	23.9%	16
CSsBb	364.04	33760	90.0%	19	364.04	12169	25.4%	1950	364.04	247	33.9%	9
CSsBc	443.02	10573	87.1%	10	443.02	1957	25.9%	1959	443.02	1686	45.0%	13
CSlAa	432.55	302019	81.2%	4	432.55	7269	24.3%	250	432.55	1326	37.2%	5
CSlAb	383.94	1863	81.8%	4	383.94	2632	17.7%	236	383.94	327	35.5%	4
CSlAc	490.88	4975	66.7%	2	490.88	4864	25.5%	94	490.88	590	42.1%	3
CSlBa	330.22	3600003	98.2%	12	330.22	3600003	33.5%	2076	330.22	3143	33.7%	14
CSlBb	320.81	55988	90.6%	12	320.81	194381	21.0%	2160	320.81	4510	38.8%	13
CSlBc	387.07	30540	79.9%	6	387.07	124187	14.8%	1531	387.07	4963	37.4%	16
CLsAa	339.53	54	77.1%	1	339.53	99	3.9%	225	339.53	67	61.8%	3
CLsAb	346.00	186	70.4%	2	346.00	89	7.3%	80	346.00	181	58.9%	3
CLsAc	412.96	127	59.3%	3	412.96	481	18.4%	121	412.96	298	50.7%	3
CLsBa	264.98	82113	91.7%	12	264.98	5682	7.2%	2318	264.98	2169	72.8%	16
CLsBb	368.71	378	89.8%	10	368.71	1083	15.1%	2020	368.71	343	71.2%	12
CLsBc	422.28	387	76.8%	6	422.28	1565	17.5%	1432	422.28	830	61.3%	13
CLlAa	288.95	17180	93.1%	4	288.95	326	1.0%	171	288.95	1676	76.3%	3
CLlAb	313.99	541	70.7%	4	313.99	538	6.2%	142	313.99	372	56.3%	4
CLlAc	416.27	2582	62.6%	2	416.27	1134	4.4%	192	416.27	1269	50.9%	4
CLlBa	272.75	1134038	98.0%	20	272.75	372973	3.5%	2341	272.75	9067	72.3%	17
CLlBb	327.62	75676	92.6%	14	327.62	32121	6.0%	2540	327.62	11022	73.5%	13
CLlBc	372.66	71113	86.4%	12	372.66	6061	13.3%	2062	372.66	7799	69.9%	13
RCSsAa	447.06	97	69.6%	3	447.06	160	23.3%	349	447.06	55	40.2%	4
RCSsAb	485.70	172	70.6%	4	485.70	131	29.0%	250	485.70	99	48.8%	5
RCSsAc	552.14	228	73.4%	6	552.14	357	35.4%	309	552.14	91	46.9%	5
RCSsBa	387.08	11321	100.0%	15	387.08	8228	32.4%	2208	387.08	1542	60.7%	18
RCSsBb	466.63	1178	83.3%	6	466.63	3563	31.9%	1890	466.63	339	52.8%	16
RCSsBc	409.11	814	69.5%	21	409.11	3695	28.6%	2038	409.11	363	37.2%	13
RCSlAa	411.00	108673	86.0%	5	411.00	48954	26.0%	313	411.00	2900	50.8%	5
RCSlAb	466.40	10778	69.6%	6	466.40	10091	24.3%	266	466.40	662	44.3%	4
RCSlAc	468.45	19509	56.2%	4	468.45	12658	22.4%	356	468.45	4109	40.1%	5
RCSlBa	285.31	2648429	100.0%	20	285.31	2215035	14.1%	2353	285.31	7309	50.2%	14
RCSlBb	359.83	21811	92.3%	15	359.83	184898	8.6%	2208	359.83	4672	40.6%	16
RCSlBc	412.30	156114	78.8%	14	412.30	782005	30.8%	2205	412.30	20721	40.6%	17
RCLsAa	374.30	167	62.3%	2	374.30	277	17.7%	241	374.30	459	52.5%	4
RCLsAb	440.75	635	53.5%	1	440.75	399	26.8%	182	440.75	142	58.6%	4
RCLsAc	455.27	430	76.0%	4	455.27	120	19.7%	283	455.27	457	61.5%	4
RCLsBa	317.95	126705	100.0%	16	317.95	155429	30.8%	2227	317.95	9859	83.2%	19
RCLsBb	369.17	586	88.5%	12	369.17	12927	26.2%	2141	369.17	964	76.4%	16
RCLsBc	342.19	209	79.0%	4	342.19	1764	29.5%	2229	342.19	573	68.2%	13
RCLlAa	378.16	246	76.9%	3	378.16	243	13.2%	333	378.16	104	67.5%	4
RCLlAb	316.59	156	55.8%	2	316.59	315	2.9%	262	316.59	254	44.0%	4
RCLlAc	385.47	149	57.3%	2	385.47	199	16.6%	202	385.47	104	47.7%	15
RCLlBa	302.72	888888	100.0%	5	302.72	809012	21.4%	2308	302.72	14919	76.3%	18
RCLlBb	388.18	311218	73.8%	11	388.18	1451018	22.4%	2159	388.18	39318	62.6%	16
RCLlBc	364.82	5905	83.0%	12	364.82	34211	22.2%	2051	364.82	8003	64.0%	14

Table 14: Comparison of the solving times and the LP-relaxations of the three formulations on solving problem instances with 12 packages. For each formulation, the first two columns contain the objective value of solving the problem instance and the corresponding runtime. The third column shows the percentage difference between the objective value of the LP relaxation and the objective value of the problem instance, and the fourth column contains the runtime of the LP relaxation. All times reported are in milliseconds with a maximum runtime of one hour.

C Results of the heuristic on benchmark instances with 16 parcels

	Formulation III			Heuristic				
instance	objective	vehicles	time (ms)	objective	vehicles	time (ms)	pct. obj.	pct. time
RSsAa	686.41	5	719	686.41	5	658	+0%	-8.5%
RSsAb	309.71	3	890	309.71	3	742	+0%	-16.6%
RSsAc	590.22	4	229	590.22	4	239	+0%	+4.4%
RSsBa	333.04	3	4455	333.04	3	346	+0%	-92.2%
RSsBb	392.63	3	13305	392.63	3	632	+0%	-95.2%
RSsBc	426.60	3	6179	426.60	3	494	+0%	-92.0%
RSlAa	424.40	3	8894	424.40	3	708	+0%	-92.0%
RSlAb	450.43	3	1782	450.43	3	462	+0%	-74.1%
RSlAc	537.26	3	1643	537.26	3	217	+0%	-86.8%
RSlBa	422.06	3	73663	422.06	3	601	+0%	-99.2%
RSlBb	457.73	4	468027	457.77	3	481	+0%	-99.9%
RSlBc	400.09	3	25330	400.09	3	673	+0%	-97.3%
RLsAa	360.29	3	243	360.29	3	995	+0%	+309.5%
RLsAb	470.84	2	1130	470.84	2	1461	+0%	+29.3%
RLsAc	554.64	2	244	554.64	2	1295	+0%	+430.7%
RLsBa	279.21	2	4484	279.21	2	2216	+0%	-50.6%
RLsBb	316.23	2	2920	316.23	2	4294	+0%	+47.1%
RLsBc	347.96	3	2470	362.62	2	3189	+4.2%	+29.1%
RLlAa	325.16	1	50280	325.16	1	9352	+0%	-81.4%
RLlAb	454.65	2	13970	476.91	3	1745	+4.9%	-87.5%
RLlAc	436.90	2	916	437.04	2	2924	+0%	+219.2%
RLlBa	385.63	2	383195	385.63	2	4908	+0%	-98.7%
RLlBb	348.79	1	36757	348.79	1	9372	+0%	-74.5%
RLlBc	417.56	2	31471	417.56	2	5934	+0%	-81.1%
CSsAa	335.35	2	2542	335.35	2	568	+0%	-77.7%
CSsAb	567.53	4	416	567.53	4	273	+0%	-34.4%
CSsAc	575.32	4	233	575.32	4	284	+0%	+21.9%
CSsBa	471.85	4	9034	471.85	4	1286	+0%	-85.8%
CSsBb	412.46	3	7344	414.28	3	542	+0.4%	-92.6%
CSsBc	478.40	4	16095	478.40	4	492	+0%	-96.9%
CSlAa	539.27	4	2333	539.27	4	811	+0%	-65.2%
CSlAb	413.31	3	2732	413.31	3	486	+0%	-82.2%
CSlAc	422.40	3	1342	422.40	3	663	+0%	-50.6%
CSlBa	349.65	3	11985	349.65	3	2801	+0%	-76.6%
CSlBb	417.10	3	14098	417.10	3	668	+0%	-95.3%
CSlBc	436.52	3	124113	436.52	3	829	+0%	-99.3%
CLsAa	292.17	2	1870	292.17	2	848	+0%	-54.7%
CLsAb	498.67	3	150	498.67	3	605	+0%	+303.3%
CLsAc	464.58	2	242	464.58	2	643	+0%	+165.7%
CLsBa	278.31	2	16694	278.31	2	1457	+0%	-91.3%
CLsBb	344.22	2	10600	344.22	2	1499	+0%	-85.9%
CLsBc	454.81	2	8262	454.81	2	1805	+0%	-78.2%
CLlAa	284.23	2	3870	284.23	2	1652	+0%	-57.3%
CLlAb	364.46	2	5447	364.46	2	4014	+0%	-26.3%
CLlAc	392.91	2	3637	392.91	2	2131	+0%	-41.4%
CLlBa	291.30	2	1591121	291.30	2	6240	+0%	-99.6%
CLlBb	385.06	2	1181848	385.06	2	7089	+0%	-99.4%
CLlBc	381.78	2	22817	381.78	2	4197	+0%	-81.6%
RCSsAa	347.54	3	462	347.54	3	279	+0%	-39.6%
RCSsAb	599.28	4	497	599.28	4	184	+0%	-63.0%
RCSsAc	571.44	3	380	571.44	3	153	+0%	-59.7%
RCSsBa	339.07	3	5550	339.07	3	531	+0%	-90.4%
RCSsBb	400.11	3	2811	400.11	3	383	+0%	-86.4%
RCSsBc	501.73	3	8100	502.68	3	386	+0.2%	-95.2%
RCSlAa	510.75	3	3050	510.75	3	288	+0%	-90.6%
RCSlAb	490.19	3	3665	490.19	3	300	+0%	-91.8%
RCSlAc	503.74	3	5087	503.74	3	253	+0%	-95.0%
RCSlBa	437.64	3	50701	437.64	3	1035	+0%	-98.0%
RCSlBb	479.78	3	55969	479.78	3	772	+0%	-98.6%
RCSlBc	526.25	3	482535	529.77	3	528	+0.7%	-99.9%
RCLsAa	597.80	3	681	597.80	3	598	+0%	-12.2%
RCLsAb	456.89	3	228	456.89	3	738	+0%	+223.7%
RCLsAc	477.72	2	155	477.72	2	821	+0%	+429.7%
RCLsBa	328.82	2	17583	328.82	2	2323	+0%	-86.8%
RCLsBb	426.37	2	9346	426.37	2	2263	+0%	-75.8%
RCLsBc	377.30	2	5090	377.30	2	2185	+0%	-57.1%
RCLlAa	345.40	2	22799	345.40	2	1386	+0%	-93.9%
RCLlAb	411.24	2	66213	411.24	2	3610	+0%	-94.5%
RCLlAc	491.68	2	9801	491.68	2	1910	+0%	-80.5%
RCLlBa	332.92	2	53165	332.92	2	4427	+0%	-91.7%
RCLlBb	361.22	2	1159736	361.22	2	3397	+0%	-99.7%
RCLlBc	369.57	1	20121	391.65	2	4526	+6.0%	-77.5%

Table 15: The results of solving benchmark instances with 16 parcels with Formulation III and with our heuristic are shown in this table. The first seven columns contain information on the instances and the performance of the two solution methods on these instances. The last two columns express the objective value and the runtime of the heuristic's solutions as a percentage of the solutions obtained with the exact method.

D Results of the heuristic on benchmark instances with 50 parcels (1)

instance	construction	objective	pct. diff.	vehicles	time(s)	best construction	pct. diff.2	best known	pct. diff with best
RSsAa	1197.4	1082.2	-9.6%	8	4.7	1149.8	-5.9%	1060.1	+2.1%
RSsAb	1268.5	1079.7	-14.9%	8	3.9	1225.8	-11.9%	1062.2	+1.7%
RSsAc	1605.2	1241.3	-22.7%	8	3.4	1349.8	-8.0%	1228.0	+1.1%
RSsBa	1103.3	744.9	-32.5%	9	15.0	877.8	-15.1%	667.3	+11.6%
RSsBb	1144.2	892.6	-22.0%	8	8.8	997.2	-10.5%	892.5	+0.0%
RSsBc	1057.6	961.2	-9.1%	8	6.0	1057.6	-9.1%	946.2	+1.6%
RSlAa	1272.3	1068.1	-16.1%	9	11.4	1143.8	-6.6%	1040.1	+2.7%
RSlAb	1276.0	1041.5	-18.4%	8	9.0	1115.9	-6.7%	1041.5	+0.0%
RSlAc	1263.4	1139.7	-9.8%	8	6.3	1260.0	-9.5%	1133.4	+0.6%
RSlBa	1003.8	774.9	-22.8%	10	19.8	948.8	-18.3%	757.1	+2.3%
RSlBb	1171.0	978.4	-16.5%	9	10.5	1118.8	-12.6%	938.2	+4.3%
RSlBc	1190.1	918.4	-22.8%	8	16.1	1030.9	-10.9%	899.6	+2.1%
RLsAa	1167.3	922.3	-21.0%	5	22.6	1040.2	-11.3%	914.6	+0.8%
RLsAb	1319.0	843.0	-36.1%	4	38.2	1001.1	-15.8%	833.6	+1.1%
RLsAc	1029.2	873.0	-15.2%	3	35.3	1004.0	-13.1%	873.0	-0.0%
RLsBa	950.1	649.7	-31.6%	4	91.5	817.5	-20.5%	566.4	+14.7%
RLsBb	982.5	715.2	-27.2%	5	112.9	924.8	-22.7%	691.3	+3.5%
RLsBc	978.3	680.5	-30.4%	3	107.9	828.1	-17.8%	680.5	+0.0%
RLlAa	934.9	664.4	-28.9%	3	141.3	767.0	-13.4%	664.4	+0.0%
RLlAb	801.3	697.1	-13.0%	3	154.3	788.0	-11.5%	697.1	+0%
RLlAc	1013.7	789.4	-22.1%	3	124.1	886.9	-11.0%	782.8	+0.8%
RLlBa	723.9	570.4	-21.2%	4	360.2	723.5	-21.2%	540.6	+5.5%
RLlBb	825.9	622.2	-24.7%	4	227.3	751.5	-17.2%	597.6	+4.1%
RLlBc	860.9	608.4	-29.3%	3	394.2	777.4	-21.7%	602.1	+1.0%
CSsAa	1199.5	1035.9	-13.6%	8	7.0	1093.2	-5.2%	1033.8	+0.2%
CSsAb	1344.2	1230.9	-8.4%	10	6.3	1306.4	-5.8%	1207.4	+2.0%
CSsAc	1254.0	1133.3	-9.6%	8	3.6	1174.6	-3.5%	1120.5	+1.1%
CSsBa	979.4	918.8	-6.2%	8	29.9	971.0	-5.4%	902.6	+1.8%
CSsBb	1267.9	1093.6	-13.7%	9	9.8	1189.2	-8.0%	1088.6	+0.5%
CSsBc	1164.3	986.5	-15.3%	8	14.6	1086.8	-9.2%	986.5	+0%
CSlAa	1120.0	1020.6	-8.9%	8	13.4	1059.8	-3.7%	996.6	+2.4%
CSlAb	1220.4	1120.1	-8.2%	9	12.4	1208.2	-7.3%	1119.6	+0.0%
CSlAc	1262.5	1139.6	-9.7%	8	5.8	1165.7	-2.2%	1137.4	+0.2%
CSlBa	1041.5	882.7	-15.2%	8	58.7	938.6	-5.9%	882.7	+0%
CSlBb	1296.6	1050.4	-19.0%	8	32.3	1121.9	-6.4%	1050.4	+0%
CSlBc	1292.9	1133.0	-12.4%	8	16.5	1210.4	-6.4%	1127.4	+0.5%
CLsAa	762.4	633.2	-17.0%	4	108.8	700.8	-9.7%	633.2	+0%
CLsAb	746.1	698.3	-6.4%	3	78.9	729.3	-4.3%	676.5	+3.2%
CLsAc	1033.9	776.9	-24.9%	4	50.3	867.6	-10.5%	776.8	+0.0%
CLsBa	645.2	570.9	-11.5%	4	374.3	642.3	-11.1%	567.5	+0.6%
CLsBb	707.0	526.3	-25.6%	3	107.9	564.2	-6.7%	526.3	-0.0%
CLsBc	814.1	705.0	-13.4%	4	87.7	806.5	-12.6%	705.0	+0%
CLlAa	698.5	624.0	-10.7%	4	259.1	665.7	-6.3%	622.8	+0.2%
CLlAb	754.6	600.7	-20.4%	3	315.6	669.5	-10.3%	584.9	+2.7%
CLlAc	890.1	751.3	-15.6%	4	189.5	862.1	-12.9%	746.2	+0.7%
CLlBa	560.2	515.3	-8.0%	4	410.2	553.9	-7.0%	510.6	+0.9%
CLlBb	592.7	495.1	-16.5%	3	433.0	562.9	-12.0%	492.4	+0.6%
CLlBc	808.8	679.4	-16.0%	4	467.2	720.4	-5.7%	658.6	+3.2%
RCSsAa	988.3	839.5	-15.1%	9	5.6	930.6	-9.8%	839.5	+0.0%
RCSsAb	1364.8	1135.1	-16.8%	9	5.6	1292.5	-12.2%	1135.1	+0.0%
RCSsAc	1497.2	1237.4	-17.4%	8	3.5	1321.0	-6.3%	1237.4	+0.0%
RCSsBa	1051.4	882.8	-16.0%	9	14.3	959.7	-8.0%	882.8	+0%
RCSsBb	1286.8	962.6	-25.2%	7	6.7	1100.1	-12.5%	946.5	+1.7%
RCSsBc	1443.3	1112.5	-22.9%	9	7.5	1243.8	-10.6%	1112.5	+0.0%
RCSlAa	1430.3	1303.0	-8.9%	9	11.3	1378.8	-5.5%	1299.9	+0.2%
RCSlAb	1263.7	1034.5	-18.1%	7	7.4	1135.4	-8.9%	1019.6	+1.5%
RCSlAc	1255.5	1088.0	-13.3%	8	8.7	1172.7	-7.2%	1088.0	+0%
RCSlBa	1141.9	1085.5	-4.9%	9	26.0	1141.9	-4.9%	1051.1	+3.3%
RCSlBb	1181.0	1029.6	-12.8%	8	28.0	1121.4	-8.2%	1029.6	+0.0%
RCSlBc	1287.1	1036.5	-19.5%	9	24.6	1188.7	-12.8%	1028.3	+0.8%
RCLsAa	862.5	743.9	-13.7%	4	51.9	825.6	-9.9%	733.2	+1.5%
RCLsAb	1376.6	986.0	-28.4%	5	29.8	1066.9	-7.6%	972.9	+1.3%
RCLsAc	1141.4	1007.4	-11.7%	5	34.1	1141.4	-11.7%	1004.2	+0.3%
RCLsBa	671.3	459.9	-31.5%	4	156.9	597.3	-23.0%	459.0	+0.2%
RCLsBb	898.6	652.4	-27.4%	3	115.0	881.6	-26.0%	645.3	+1.1%
RCLsBc	999.5	727.2	-27.2%	4	97.7	887.4	-18.1%	707.0	+2.9%
RCLlAa	771.2	619.1	-19.7%	3	209.9	704.3	-12.1%	606.8	+2.0%
RCLlAb	968.5	797.3	-17.7%	3	134.9	864.5	-7.8%	777.8	+2.5%
RCLlAc	1013.0	790.7	-21.9%	4	121.3	889.1	-11.1%	764.4	+3.4%
RCLlBa	751.6	591.1	-21.4%	3	240.7	700.1	-15.6%	591.1	+0%
RCLlBb	742.8	587.7	-20.9%	3	230.1	644.2	-8.8%	577.2	+1.8%
RCLlBc	842.7	621.2	-26.3%	3	456.9	708.7	-12.4%	621.2	+0%

Table 16: Results of solving the 72 FDPTW benchmark instances with 50 parcels with the heuristic with parameter values *solutions* = 50 and $x = 50\%$. Columns two to four contain the objective values of the best solution after the route construction stage and the route improvement stage, and the percentage difference between these two. The number of vehicles in the solution is shown in the fifth column and the sixth column shows the runtime in seconds. The seventh column contains the best solution that resulted from the route construction and route elimination stage, which can be seen as the best solution if the heuristic did not make use of a route improvement procedure. The next column shows the percentage difference between the values in columns three and seven, which is the extra savings obtained by introducing the route improvement procedure. The penultimate and the last column show the best solution known for the instance and the percentage difference in costs with the currently found solution.

E Results of the heuristic on benchmark instances with 50 parcels (2)

instance	*solutions*	x	construction	objective	pct. diff.	vehicles	time(s)	best known	pct. diff with best
RSsAa	500	100%	1329.9	1058.9	-20.4%	8	89.7	1060.1	-0.1%
RSsAb	500	100%	1423.1	1059.6	-25.5%	8	83.5	1062.2	-0.2%
RSsAc	500	100%	1683.2	1228.0	-27.0%	9	70.5	1228.0	+0%
RSsBa	500	100%	1066.0	663.9	-37.7%	9	247.9	667.3	-0.5%
RSsBb	500	100%	1191.6	886.2	-25.6%	8	177.1	892.5	-0.7%
RSsBc	500	100%	1189.0	945.1	-20.5%	8	143.7	946.2	-0.1%
RSlAa	500	100%	1241.0	1024.9	-17.4%	9	193.7	1040.1	-1.5%
RSlAb	500	100%	1250.0	1041.5	-16.7%	8	174.2	1041.5	+0%
RSlAc	500	100%	1360.2	1133.8	-16.6%	8	133.6	1133.4	+0.0%
RSlBa	500	100%	1162.4	764.2	-34.3%	10	337.3	757.1	+0.9%
RSlBb	500	100%	1093.1	946.1	-13.5%	9	221.9	938.2	+0.8%
RSlBc	500	100%	1112.1	896.2	-19.4%	8	305.7	899.6	-0.4%
RLsAa	500	100%	1278.6	914.6	-28.5%	4	558.2	914.6	+0%
RLsAb	500	100%	1225.0	833.6	-32.0%	4	925.8	833.6	+0%
RLsAc	500	100%	1095.4	873.0	-20.3%	3	788.1	873.0	+0%
RLsBa	500	98%	1037.0	577.7	-44.3%	3	1995.8	566.4	+2.0%
RLsBb	500	80%	955.7	691.4	-27.7%	4	1782.0	691.3	+0%
RLsBc	500	83%	1012.8	680.5	-32.8%	3	1844.0	680.5	+0%
RLlAa	500	64%	934.9	664.4	-28.9%	3	1949.5	664.4	+0%
RLlAb	500	58%	910.4	697.1	-23.4%	3	1855.0	697.1	+0%
RLlAc	500	73%	1029.3	786.7	-23.6%	4	1781.2	782.8	+0.5%
RLlBa	250	50%	817.8	540.3	-33.9%	3	2844.5	540.6	-0.1%
RLlBb	396	50%	878.8	601.0	-31.6%	3	1875.1	597.6	+0.6%
RLlBc	228	50%	757.4	602.1	-20.5%	3	1734.4	602.1	+0%
CSsAa	500	100%	1156.8	1033.4	-10.7%	8	141.5	1033.8	-0.0%
CSsAb	500	100%	1393.6	1207.4	-13.4%	10	110.3	1207.4	+0%
CSsAc	500	100%	1203.8	1120.6	-6.9%	8	73.0	1120.5	+0%
CSsBa	500	100%	1000.8	906.9	-9.4%	8	443.5	902.6	+0.5%
CSsBb	500	100%	1253.9	1088.6	-13.2%	9	200.2	1088.6	+0%
CSsBc	500	100%	1164.3	986.5	-15.3%	8	245.3	986.5	+0%
CSlAa	500	100%	1104.1	992.9	-10.1%	7	295.8	996.6	-0.4%
CSlAb	500	100%	1334.5	1119.9	-16.1%	9	229.4	1119.6	+0.0%
CSlAc	500	100%	1325.2	1137.4	-14.2%	8	122.8	1137.4	+0%
CSlBa	500	100%	969.9	887.2	-8.5%	8	848.9	882.7	+0.5%
CSlBb	500	100%	1237.4	1059.6	-14.4%	8	493.5	1050.4	+0.9%
CSlBc	500	100%	1243.2	1126.8	-9.4%	9	318.4	1127.4	-0.0%
CLsAa	500	83%	762.4	633.2	-17.0%	4	1948.2	633.2	+0%
CLsAb	500	100%	778.4	676.5	-13.1%	3	1485.3	676.5	+0%
CLsAc	500	100%	1005.9	776.8	-22.8%	4	1146.1	776.8	+0%
CLsBa	240	50%	688.6	567.5	-17.6%	4	1749.0	567.5	-0.0%
CLsBb	500	83%	739.1	526.3	-28.8%	3	1893.0	526.3	+0%
CLsBc	500	100%	932.7	705.0	-24.4%	4	2163.6	705.0	+0%
CLlAa	347	50%	673.6	622.8	-7.5%	4	1972.2	622.8	+0%
CLlAb	285	50%	747.7	584.9	-21.8%	3	2038.6	584.9	+0%
CLlAc	475	50%	968.2	746.2	-22.9%	4	2087.1	746.2	+0%
CLlBa	219	50%	647.9	515.3	-20.5%	4	1972.8	510.6	+0.9%
CLlBb	208	50%	702.4	493.5	-29.7%	3	1773.0	492.4	+0.2%
CLlBc	193	50%	785.6	658.6	-16.2%	4	1829.0	658.6	+0%
RCSsAa	500	100%	1033.2	839.5	-18.7%	9	116.0	839.5	+0%
RCSsAb	500	100%	1489.2	1135.1	-23.8%	9	112.2	1135.1	+0%
RCSsAc	500	100%	1478.4	1248.5	-15.6%	8	73.9	1237.4	+0.9%
RCSsBa	500	100%	1127.5	883.5	-21.6%	9	225.3	882.8	+0.1%
RCSsBb	500	100%	1080.7	941.3	-12.9%	7	144.7	946.5	-0.6%
RCSsBc	500	100%	1549.9	1097.2	-29.2%	8	143.9	1112.5	-1.4%
RCSlAa	500	100%	1577.5	1298.5	-17.7%	9	200.2	1299.9	-0.1%
RCSlAb	500	100%	1326.0	1017.0	-23.3%	7	165.4	1019.6	-0.3%
RCSlAc	500	100%	1361.8	1076.6	-20.9%	8	159.1	1088.0	-1.0%
RCSlBa	500	100%	1425.6	1055.9	-25.9%	9	442.3	1051.1	+0.5%
RCSlBb	500	100%	1194.5	1029.6	-13.8%	8	494.3	1029.6	+0%
RCSlBc	500	100%	1361.9	1034.2	-24.1%	9	443.3	1028.3	+0.6%
RCLsAa	500	100%	864.4	737.1	-14.7%	4	1185.3	733.2	+0.5%
RCLsAb	500	100%	1345.4	972.9	-27.7%	6	714.8	972.9	+0%
RCLsAc	500	100%	1225.2	1004.4	-18.0%	4	777.9	1004.2	+0.0%
RCLsBa	500	57%	682.8	459.0	-32.8%	4	1947.2	459.0	+0%
RCLsBb	500	78%	1057.4	645.3	-39.0%	3	1851.8	645.3	+0%
RCLsBc	500	92%	947.2	706.6	-25.4%	4	1883.7	707.0	-0.1%
RCLlAa	429	50%	771.2	606.8	-21.3%	3	1861.3	606.8	+0%
RCLlAb	500	67%	957.8	777.8	-18.8%	3	1735.8	777.8	+0%
RCLlAc	500	74%	841.7	764.4	-9.2%	4	2082.7	764.4	+0%
RCLlBa	374	50%	774.7	591.1	-23.7%	3	1914.2	591.1	+0%
RCLlBb	391	50%	827.5	575.4	-30.5%	3	2145.4	577.2	-0.3%
RCLlBc	197	50%	891.4	624.5	-29.9%	3	1906.2	621.2	+0.5%

Table 17: Results of solving the 72 FDPTW benchmark instances with 50 parcels with the heuristic with the parameter values based on the decision rules from Section 6.2.3. Columns two and three contain the parameter values used in solving the instance. The fourth to sixth column contain the objective values of the best solution after the route construction stage and the route improvement stage, and the percentage difference between these two. The number of vehicles in the solution is shown in the seventh column and the eight column shows the runtime in seconds. The last two columns show the best solution known for the instance and the percentage difference in costs with the currently found solution.

F Example of vehicle routing before and after the introduction of offices and pick-up points

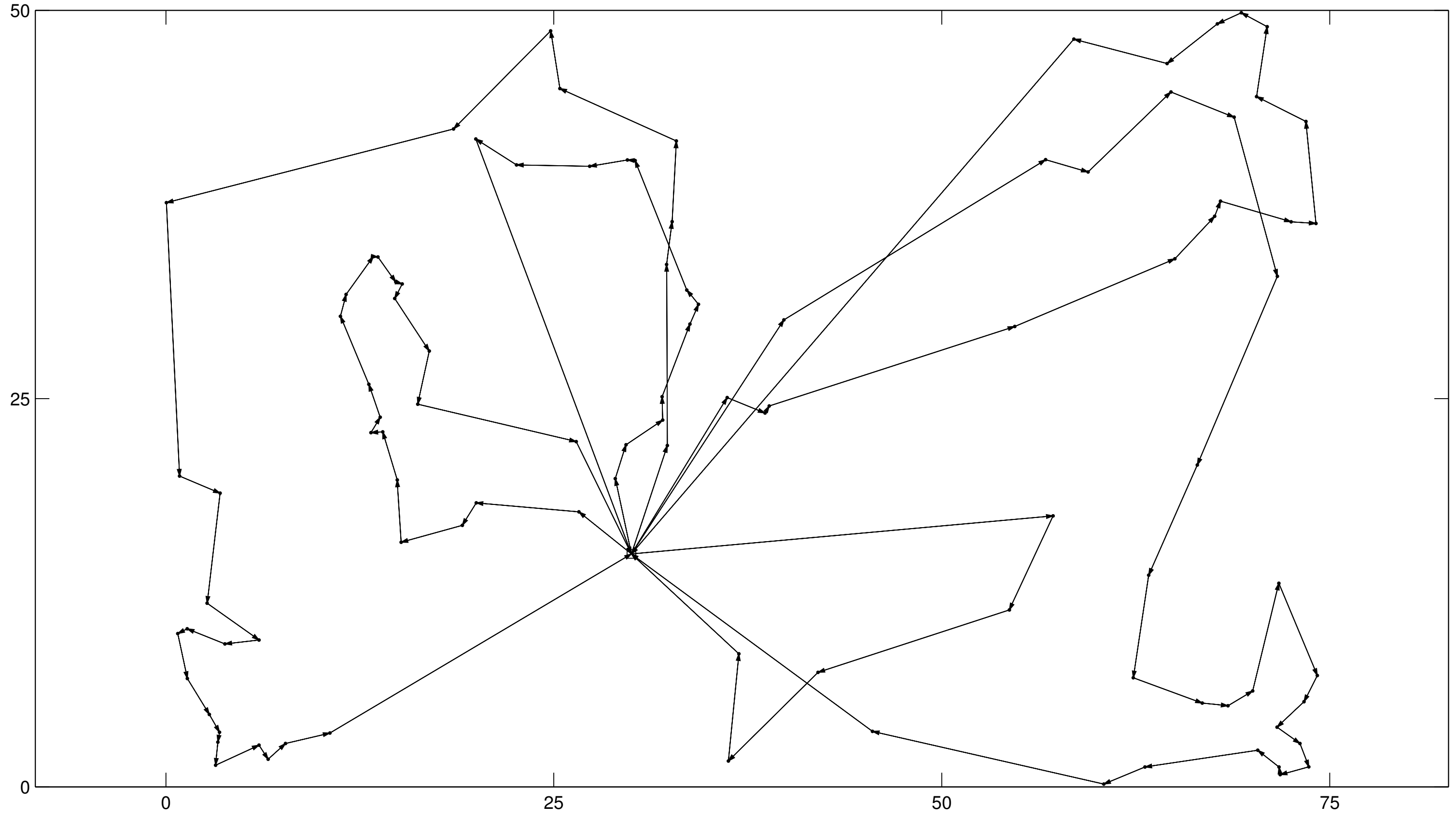

Figure 12: This is the map displaying the routes of the best solution found for version D of the instance with $p_{office} = 0.3$ and $P_{pick-up} = 0.7$ corresponding to Table 12 in Section 6.2.4. As one can observe, in this version of the instance all offices and pick-up points have been eliminated and every single home has to be visited. The total costs of this solution are 646.20.

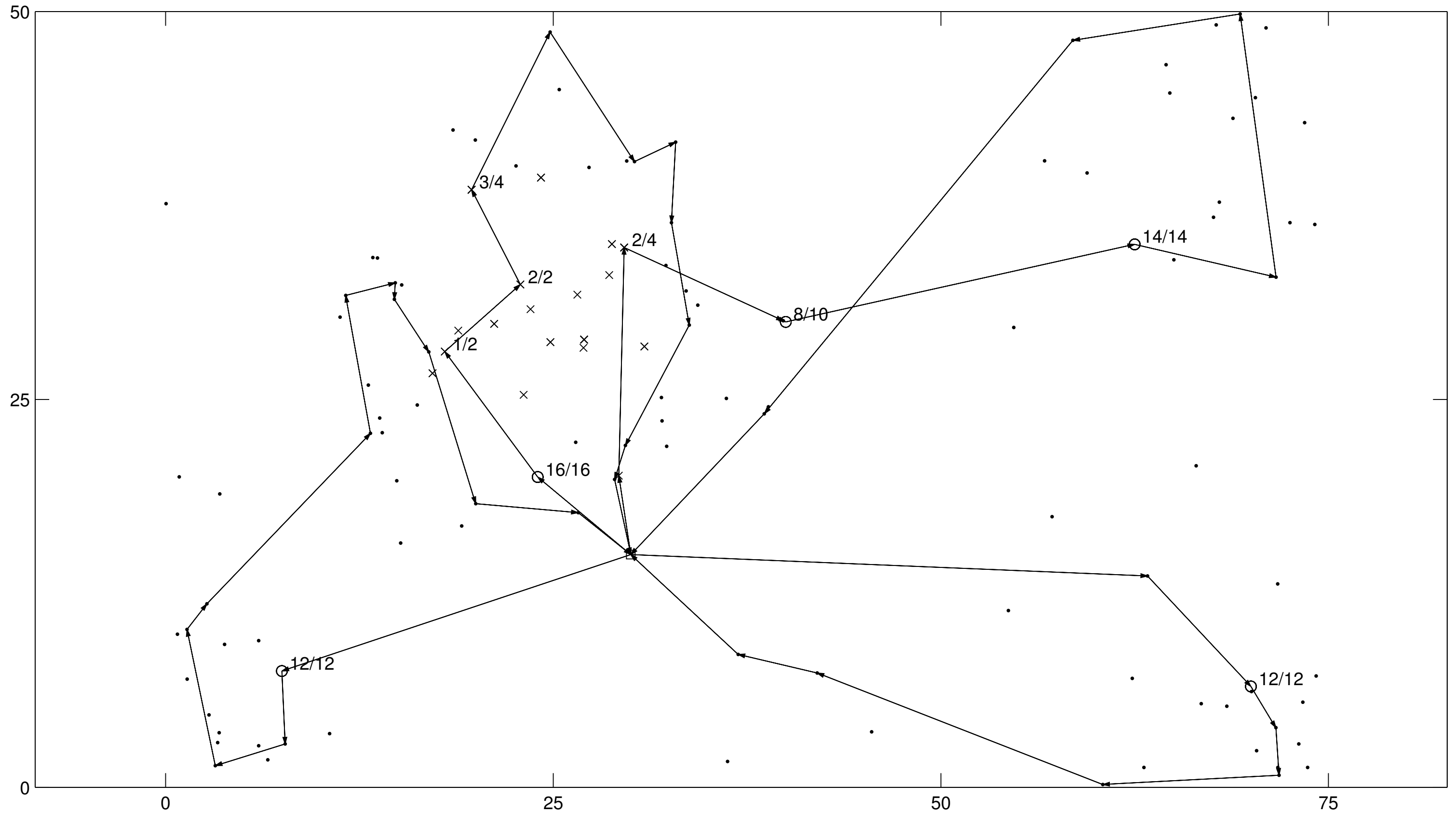

Figure 13: This is the map displaying the routes of the best solution found for version A of the instance with $p_{office} = 0.3$ and $P_{pick-up} = 0.7$ corresponding to Table 12 in Section 6.2.4. The numbers above the offices and pick-up points indicate the number of parcels delivered there, and the maximum number of parcels that can be delivered at that location. The total costs of this solution are 412.40.