The Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and Recharging Stations

Gerhard Hiermann[1], Jakob Puchinger[1] and Richard F. Hartl[2]

[1]Mobility Department, Dynamic Transportation Systems,
AIT Austrian Institute of Technology GmbH, Austria
gerhard.hiermann.fl@ait.ac.at, jakob.puchinger@ait.ac.at

[2]Department of Business Administration
University of Vienna, Austria
richard.hartl@univie.ac.at

Working Paper, March 2014

Abstract

Due to new regulations and further technological progress in the field of electric vehicles, the research community faces the new challenge of incorporating the electric energy based restrictions into vehicle routing problems. One of these restrictions is the limited battery capacity which makes detours to recharging stations necessary, thus requiring efficient tour planning mechanisms in order to sustain the competitiveness of electric vehicles compared to conventional vehicles. We introduce the Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and recharging stations (E-FSMVRPTW) to model decisions to be made with regards to fleet composition and the actual vehicle routes including the choice of recharging times and locations. The available vehicle types differ in their transport capacity, battery size and acquisition cost. Furthermore, we consider time windows at customer locations, which is a common and important constraint in real-world routing and planning problems. We propose to solve this problem using a hybrid heuristic, which combines an Adaptive Large Neighbourhood Search (ALNS) with an embedded local search and labelling procedure for intensification. By solving a newly created set of benchmark instances for the E-FSMVRPTW as well as benchmarks of related problems we show the effectiveness of the proposed approach.

Introduction

Current research in sustainable and energy efficient mobility is strongly motivated by increasing concerns about climate change and rising green house gas emissions. The introduction of electrically powered vehicles is one of the major directions taken in order to address these concerns. One of the main operational challenges for introducing electric vehicles in transport applications is their limited range and long recharging times. Besides acquisition cost, the take-up of electric vehicles in the transportation business will strongly depend on methods alleviating the range and recharging limitations. Selecting the right vehicles for specific transport requirements while minimizing overall cost is therefore of crucial importance. We propose to address this task by introducing a new optimization problem, the so-called *Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and Recharging Stations* (E-FSMVRPTW). It combines and subsumes the well known Fleet Size Mix Vehicle Routing Problem with Time Windows (FSMVRPTW) and the recently defined Electric Vehicle Routing Problem with Time Windows and Recharging Stations (E-VRPTW).

Related Work

Solving and optimizing tour assignments of vehicles is a well known and well studied field of research, known as *Vehicle Routing Problem* (VRP) (Toth and Vigo 2001). The basic problem variant is the *Capacitated VRP* (CVRP), where each customer has a given demand that has to be satisfied, with respect to a maximum vehicle capacity. The *VRP with Time Windows* (VRPTW) extends the CVRP by adding time windows to the depot and the customers. A survey on this problem class is provided in Bräysy and Gendreau (2005a,b).

The classical VRP has been extended in various ways to account for additional real world aspects. These more complicated problems are often called "rich VRPs" or multi-attribute VRPs; see Vidal et al. (2013b). Our research combines two such streams of research, (1) the use of electrical/zero-emission vehicles in "green" VRPs and (2) the analysis of VRPs with heterogeneous fleet. In the first stream, Erdoğan and Miller-Hooks (2012) extend the CVRP to the *Green VRP* where tours for Alternative Fuel Vehicles are optimized. The uneven distribution of *Alternative Fuelling Stations* (AFS) leads to the problem of deciding when a vehicle has to visit AFS during its tour (possibly multiple times) in order to minimize the distance travelled but avert to run out of fuel. Two construction heuristics are presented: A Clarke and Wright Savings (Clarke and Wright 1964) heuristic that is extended to include AFS nodes during the merge process, and a Density Based Clustering exploiting spatial properties of the problem. Both approaches terminate after creating a solution containing a feasible set of routes, which is then improved by means of local search. The methods were tested on a randomly generated test set as well as a real world case study considering up to 500 (randomly located) customers and 21 existing AFS. The presented computational results demonstrate that the proposed methods are able to find feasible solutions for large instances. For smaller random instances the presented methods obtain solutions that are, on average, less then 10% worse then the best known solutions obtained with CPLEX.

Schneider, Stenger, and Goeke (2014) adapted the Green VRP to electric vehicles (EV) and added time window constraints, introducing the *Electric VRPTW with Recharging Stations* (E-VRPTW). The aim is to find tours satisfying charge constraints (the state of charge may never fall below zero) and time window constraints. The recharging process complicates the time calculations, since the required recharging time depends on the state of the charge. The problem is solved by a *Variable Neighbourhood Search* (VNS) approach using *Tabu Search* (TS) as local optimization technique. The proposed approach was tested on a new benchmark set based on the traditional Solomon instances for the VRPTW that have been extended with recharging stations as well as the instances of Erdoğan and Miller-Hooks (2012) for the GreenVRP. In addition, the presented approach has been adapted to solve instances of the related *Multi Depot VRP with Inter-depot routes* (MDVRPI) (Crevier, Cordeau, and Laporte 2007, Tarantilis, Zachariadis, and Kiranoudis 2008), where vehicles can visit depots between customers to restock in order satisfy the demand of the customers. The presented methods were able to improve upon previous results for the GreenVRP and new best solutions have been obtained for the MDVRPI. Based on the proposed benchmark set, smaller instances allowing a direct comparison with CPLEX have been created. The comparison shows, that the VNS/TS approach is able to find optimal solutions (where known).

A different electric vehicle routing problem has been presented in Conrad and Figliozzi (2011), the so-called *Recharging VRP* (RVRP). Instead of using dedicated recharging stations the authors assume that a set of customers provides the option to recharge at their location. The EV can perform a recharging operation to a certain percentage of the maximum capacity, which needs a fixed amount of time. Recharging operations and service at the customer can be performed simultaneously. Different problem instances are proposed and solved with various parameter settings using a modified iterative construction and improvement algorithm. The paper focuses on the analysis of the instance parameters and their contribution to the solutions obtained to generate meaningful solution bounds for the average tour distance.

The second stream of research related to our work is that of VRPs with heterogeneous fleet. The *Mixed Fleet* or *Heterogenous VRP* considers problems where different types of vehicles are available. It was first introduced in Golden et al. (1984). Baldacci, Battarra, and Vigo (2008) identifies five major subclasses differing in the number of vehicles available (limited, unlimited), whether a fixed cost per vehicle is considered or not and if the routing cost depend on the vehicle type. The original formulation by Golden et al. (1984) considers an unlimited number of vehicles with fixed acquisition costs and vehicle type independent routing costs, which is classified as a *Fleet Size and Mix VRP with Fixed costs* (FSMF) (Baldacci, Battarra, and Vigo 2008).

Liu and Shen (1999) reformulate the FSMF to consider time windows, creating the *Fleet Size and Mix Vehicle Routing Problem with Time Windows* (FSMVRPTW). The so-called *En Route* time, i.e., the time between departing from and returning to the depot minus the cumulative service time at the customers in the respective route is considered as routing cost. The proposed approach was applied to a new benchmark set based on the well known Solomon instances for the VRPTW. This benchmark extends the 56 VRPTW instances by providing three classes of vehicle type settings (A,B,C) varying from 3 to 6 vehicle types with different cost and capacity margins, resulting in 168 problem instances in total.

Bräysy et al. (2008) propose a three phase *multi-start deterministic annealing* metaheuristic (MSDA) to solve the FSMVRPTW. A threshold acceptance criterion was used where the maximum threshold of accepting a worse solution is reduced every set iteration until no worsening is allowed. The solution itself is created using a systematic and deterministic multi-phase approach, starting with a modified *Clarke and Wright* savings heuristic, followed by a route elimination procedure and a systematic local search where three operators are applied every single, second or third iteration. The proposed algorithm shows a very good performance when run for a similar amount of time compared to previous approaches. Furthermore, with longer run-times new best solutions for almost every instance are obtained.

An *Adaptive Memory Program* (AMP) was proposed by Repoussis and Tarantilis (2010). A memory of good solution features (route assignments and orderings) is maintained within an *Iterated Local Search* (ILS) using TS as local improvement procedure. A specialized construction heuristic uses the information of the memory to create new solutions for the following run of the ILS which is repeated until the predefined maximum runtime has been reached. The approach was able to find new best solutions for over a half of the instances and found comparable results for the others.

Most recently Vidal et al. (2013a) presented a hybrid evolutionary algorithm using population management mechanisms and a generalized solution representation with modular evaluation which is able to solve a large class of VRPs, including the FSMVRPTW. The approach was able to find new best solutions for many instance in a reasonable amount of time.

Contributions of this article

We combine the streams of research on the EVRPTW and the FSMVRPTW and introduce a new vehicle routing problem: the Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and recharging stations (E-FSMVRPTW). This is not an artificial problem but a natural planning problem within the increasingly important area of route optimization for electric vehicles.

The motivation comes from a real world project.

To define the problem precisely, we provide a mathematical formulation as a MIP model and – after some preprocessing and symmetry breaking – solve small instances with CPLEX to have benchmarks for heuristic approaches.

In order to tackle problem instances of realistic size, we propose a metaheuristic approach based on Adaptive Large Neighbourhood Search (ALNS) with embedded local search and labelling procedures.

We describe a sophisticated move evaluation process, which enables us to calculate the change in the objective value using common moves in constant time. We propose to compute optimal positions of recharging stations using a labelling procedure with label extension and elimination rules for the shortest path problem with time and energy resources.

The presented computational experiments show that our approach is able to find optimal solutions for small instances (where we have been able to obtain optimal solutions using CPLEX) and that high quality feasible solutions can be found for the larger instances.

Furthermore we demonstrate the competitiveness of our approach by comparing it to the state of the art for solving the basic problems, E-VRPTW and FSMVRPTW. For the E-VRPTW new best known solutions are obtained.

Overview of the article

In Section 1, a mixed-integer linear programming formulation is given alongside detailed formal notation. Section 2 describes the modules of our ALNS approach in detail. Results of our experiments for the new benchmark as well as the FSMVRPTW and E-VRPTW instances are presented in Section 3. Section 4 provides a short summary and a conclusion of our work.

1 Problem Description and Model

The *Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and recharging stations* (E-FSMVRPTW) consists of finding admissible tours for vehicles of different types such that every customer is covered by exactly one tour while minimizing acquisition cost and the total distance travelled. Our formulation is based on Bräysy et al. (2008) for the FSMVRPTW and Schneider, Stenger, and Goeke (2014) for the E-VRPTW.

1.1 Mixed Integer Programming Model

An instance of the E-FSMVRPTW consists of a set of customers C, a set of recharging stations F, a depot node and k different vehicle types. We define N as set of nodes $N = C \cup F'$ consisting of a set of customers C and a set of dummy nodes F' representing the recharging stations of F multiple times in order to allow for multiple visits. We use u_0 and u_{n+1} to represent the start and end depot nodes respectively. N_0 (N_{n+1}) denotes the set of nodes with the start depot node (end depot node) whereas $N_{0,n+1}$ addresses $N \cup \{u_0, u_{n+1}\}$. A binary variable x_{ij}^k is used for indicating whether a vehicle of type k visits node j after visiting node i.

The maximum capacity of vehicle type k is defined in Q^k, and p_i is the amount of capacity needed to serve node i. Variables q_i^k hold the current load of a vehicle of type k in node i. Similar to the load capacity, Y^k is the maximum energy capacity of vehicle type k and y_i^k is the variable holding the current energy level of a vehicle of type k in node i. Furthermore r^k represents the energy consumption per distance unit travelled of a vehicle of type k and g^k corresponds to the energy recharging rate per time unit. For each node i a range $[e_i, l_i]$ is defined representing the time windows, where e_i marks the earliest and l_i the latest begin of service. The service time of each node is represented by s_i and the actual start of service time is stored in variable τ_i. The distance and the travel time between two nodes i and j are denoted as d_{ij} and t_{ij} respectively.

The E-FSMVRPTW can be modelled as follows:

$$min \quad \sum_{k\in V}\sum_{j\in N} f^k x_{0j}^k + \sum_{k\in V}\sum_{i\in N_0, j\in N_{n+1}, i\neq j} d_{ij}x_{ij}^k \tag{1.1}$$

$$s.t. \quad \sum_{k\in V}\sum_{j\in N_{n+1}, i\neq j} x_{ij}^k = 1 \qquad \forall i\in C \tag{1.2}$$

$$\sum_{k\in V}\sum_{j\in N_{n+1}, i\neq j} x_{ij}^k \leq 1 \qquad \forall i\in F' \tag{1.3}$$

$$\sum_{i\in N_{n+1}, i\neq j} x_{ji}^k - \sum_{i\in N_0, i\neq j} x_{ij}^k = 0 \qquad \forall j\in N, \forall k\in V \tag{1.4}$$

$$e_j \leq \tau_j \leq l_j \qquad \forall j\in N_{0,n+1} \tag{1.5}$$

$$\tau_i + (t_{ij}+s_i)x_{ij}^k - l_0(1-x_{ij}^k) \leq \tau_j \qquad \forall k\in V, \forall i\in C_0, \forall j\in N_{n+1}, i\neq j \tag{1.6}$$

$$\tau_i + t_{ij}x_{ij}^k + g^k(Y^k - y_i^k) - (l_0 + g^kY^k)(1-x_{ij}^k) \leq \tau_j \qquad \forall k\in V, \forall i\in F', \forall j\in N_{n+1}, i\neq j \tag{1.7}$$

$$q_j^k \leq q_i^k - p_i x_{ij}^k + Q^k(1-x_{ij}^k) \qquad \forall k\in V, \forall i\in N_0, \forall j\in N_{n+1}, i\neq j \tag{1.8}$$

$$0 \leq q_j^k \leq Q^k \qquad \forall k\in V, \forall j\in N_{0,n+1} \tag{1.9}$$

$$0 \leq y_j^k \leq y_i^k - (r^k d_{ij})x_{ij}^k + Y^k(1-x_{ij}^k) \qquad \forall k\in V, \forall i\in C, \forall j\in N_{n+1}, i\neq j \tag{1.10}$$

$$0 \leq y_j^k \leq Y^k - (r^k d_{ij})x_{ij}^k \qquad \forall k\in V, \forall i\in F_0', \forall j\in N_{n+1}, i\neq j \tag{1.11}$$

$$y_0^k = Y^k \qquad \forall k\in V \tag{1.12}$$

$$x_{ij}^k \in \{0,1\} \qquad i\in N_0, j\in N_{n+1}, i\neq j, \forall k\in V \tag{1.13}$$

The problem is a minimization problem with an objective function (1.1) consisting of two parts. The first part is the sum of the costs of all vehicles used, i.e., if a vehicle is driving from the depot to any other node than the depot (indicated by a value greater than zero) the corresponding acquisition cost (f^k) is added. The second part counts the total distance travelled by each car. Equation (1.2) ensures that every customer is visited by any vehicle exactly once while (1.3) covers the fact that a recharge station does not need to be used in a solution at all. Furthermore we restrict each dummy node representing a recharge station to be visited at most once by any vehicle to ensure correct assignments to node specific variables like the start of service time τ_i. Equation (1.13) forces the value of x_{ij}^k to be either zero or one.

The timing constraints are covered by the constraints (1.5) - (1.7). The constraint described in (1.5) ensures that the start of service time τ_i has to be inside the time window $[e_i, l_i]$ of node i. In (1.6) we ensure that the starting time of the next node τ_j has to consider the start time τ_i plus the service time s_i of the previous node i in addition to the travel time t_{ij} in case that node i is a customer node or the start depot. If the previous node is a recharging station, i.e., $i \in F'$, constraint (1.7) considers the recharge time instead of a service time. The recharging time depends on the remaining energy when arriving at the recharging station y_i^k, the maximum energy capacity of the vehicle Y^k and the recharging rate g^k. As assumed in Schneider, Stenger, and Goeke (2014) a vehicle is always recharged to full capacity every time it visits a recharging station.

To ensure that the demand of the customers can be fulfilled by the assigned vehicle k, constraint (1.8) ensures that the load of the vehicle in the next node q_j^k depend on the load of the previous node q_i^k plus the demand p_i. Equation (1.9) restricts the load q_i^k never to exceed the maximum capacity Q^k of the vehicle k as well as ensures a positive value.

As the problem also considers resource consumption, in our case energy, we have to model this as well. Constraint (1.10) restricts the current available energy y_j^k to be smaller than the previous y_i^k and to consider the energy consumption per km/mile r^k when travelling from a customer node $i \in C$ to any other node $j \in N_{n+1}$. As we assume that a vehicle is recharged to the maximum capacity when visiting a recharge station, constraint (1.11) ensures that the current load is assumed to be the maximum minus the consumed amount. Both constraints also ensure that the available energy is always positive in any node $i \in N_{0,n+1}$.

This model can also solve the FSMVRPTW and the E-VRPTW by setting the energy consumption to zero or initializing the available set of vehicles with only one vehicle type.

1.2 Preprocessing and Additional Inequalities

In order to increase the efficiency when solving the proposed model using a generic MIP solver such as CPLEX, we introduce some preprocessing steps and valid inequalities.

Preprocessing. As a preprocessing step we identify and forbid the use of an arc (i, j) by a vehicle of type k if one of the following equations is violated (based on Schneider, Stenger, and Goeke (2014)):

$$p_i + p_j \leq Q^k \qquad \forall i, j \in C, \forall k \in V \tag{1.14}$$

$$e_i + s_i + t_{ij} \leq l_j \qquad \forall i \in C, \forall j \in N_{n+1} \tag{1.15}$$

$$e_i + s_i + t_{ij} + s_j + t_{jn+1} \leq l_{n+1} \qquad \forall i, j \in C \tag{1.16}$$

$$r^k(d_{vi} + d_{ij} + d_{jw}) \leq Y^k \qquad \forall i, j \in C, \forall v, w \in F', \forall k \in V \tag{1.17}$$

The first equation (1.14) removes direct arcs between two nodes if their cumulative demand is greater than the maximal transport capacity of the vehicle type. Equations (1.15) and (1.16) ensures that the time window constraint can be fulfilled whereas equation (1.17) prohibits any detour visiting any two customer leading to a violation of the energy constraint.

As arcs between dummy nodes representing the same recharging station are not necessary we have to remove them from the set of possible arcs. The following equations use a function $orig(v) = w$ which is a direct mapping of the dummy node $v \in F'$ to the actual recharging station

$w \in F$.

$$x_{ij}^k = 0 \qquad \forall i, j \in F', orig(i) = orig(j), \forall k \in V \tag{1.18}$$

$$x_{0i}^k = 0 \qquad \forall i \in F', orig(i) = u_0, \forall k \in V \tag{1.19}$$

$$x_{in+1}^k = 0 \qquad \forall i \in F', orig(i) = u_{n+1}, \forall k \in V \tag{1.20}$$

Equations (1.18)-(1.20) are used to handle identical nodes. Equation (1.18) prohibits the use of an arc between two dummy nodes of recharging stations, if the original recharging station of these nodes is the same, whereas Equation (1.19) and (1.20) constrain the use of depot nodes as recharging stations if they are visited right after (before) leaving from (returning to) the depot.

Additional inequalities. They are introduced to avoid symmetries and unnecessary assignments resulting from the use of dummy nodes.

$$\sum_{u \in N_0} x_{ui}^k \geq \sum_{v \in N_0} x_{vj}^k \qquad \forall i, j \in F', i < j, orig(i) = orig(j), \forall k \in V \tag{1.21}$$

To avoid symmetries Equation (1.21) prohibits the use of a dummy recharging station $j \in F'$ by vehicle k if a dummy recharging station $i \in F'$ of the same original recharging station ($orig(i) = orig(j)$) but with a lower index is currently not used by vehicle k (i.e., no vehicle of type k is using the recharging station i).

$$\sum_{k \in V} \sum_{i \in N} x_{0i}^k Q^k \geq \sum_{i \in N} p_i \tag{1.22}$$

$$\sum_{k \in V} \sum_{i \in N} x_{0i}^k \leq |C| \tag{1.23}$$

As additional constraints to strengthen the LP relaxation we added equations (1.22) and (1.23). Equation (1.22) ensures that the cumulative capacity of the fleet we use covers the demand of all nodes whereas (1.23) bounds the maximum number of vehicles use to the number of customers served.

2 Adaptive Large Neighbourhood Search

In order to be able to solve benchmark instances of realistic sizes, we developed an *Adaptive Large Neighbourhood Search* (ALNS) based on Ropke and Pisinger (2006) and Pisinger and Ropke (2007). We combine the ALNS with a local search procedure for intensification, as well as a labelling procedure to optimize the position of recharging stations within the routes at certain points during search. We first describe the data structures for representing, evaluating and improving solutions. Then we present the local search and labelling procedures. Finally the ALNS including the used destroy and repair operators is discussed.

2.1 Representation and Evaluation

We base our approach of evaluating solutions on the work of Vidal et al. (2013a,b, 2014), where a sequence-based evaluation approach is introduced. It exploits the fact that solutions resulting

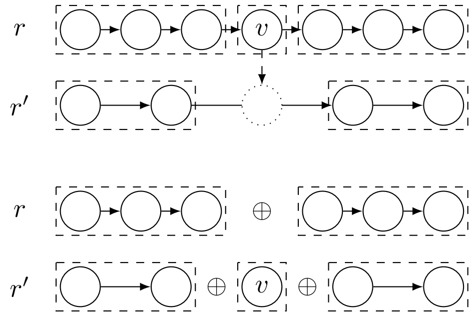


Figure 2.1: Example of a shift (relocate) move from r to r' (dashed line) using data of preprocessed sequences (dashed boxes) and the concatenation operator $\oplus$.

from a bounded number of edge exchanges and node relocations performed on a solution can also be achieved using a recombination of a bounded number of sequences of visits of the same solution (see Kindervater and Savelsbergh (1997), Vidal et al. (2013a)). Vidal et al. (2014) further showed that a large number of objective values and constraints can be evaluated using information of sub-sequences in $O(1)$. To achieve these bounds it is required to pre-compute all necessary sub-sequences of each route prior to the local search call and to update the sub-route information of changed routes after a move is performed.

Figure 2.1 shows an example of a move evaluation by concatenation of preprocessed sequences. Starting with two routes, as shown at the top, we can evaluate the changed state (at the bottom) using three concatenation operations.

2.1.1 Concatenation operators from the literature.

According to the categorization introduced by Vidal et al. (2013b), our problem consists of several *EVAL* attributes attributes, i.e., capacity, distance, time but also charge, and a single *ASSIGN* attribute, the vehicle type. For the concatenation of the capacity and distance attribute – and the time attibute to some extent – we used the definitions also presented in Vidal et al. (2014). The general battery charge concatenation is based on the work of Schneider, Stenger, and Goeke (2014) for the E-VRPTW, but was adapted for our sequence-based evaluation approach. In the following we will describe the concatenation operations in more detail.

Capacity and distance. Both, capacity $Q(\sigma)$ and distance $Dist(\sigma)$ can be concatenated in a straight forward way, only relying on constant operations:

$$Q(\sigma_1 \oplus \sigma_2) = Q(\sigma_1) + Q(\sigma_2) \tag{2.1}$$

$$Dist(\sigma_1 \oplus \sigma_2) = Dist(\sigma_1) + d_{\sigma_1\sigma_2} + Dist(\sigma_2) \tag{2.2}$$

The load capacity of a concatenation is simply the sum of both capacities. The distance of a concatenation is the sum of the individual distances plus the distance from the last node of the first sequence to first node of the second sequence. The latter distance is denoted by $d_{\sigma_1\sigma_2}$.

State of charge. In order to compute the information concerning the state of charge, three values for each sequence are required. First, the energy required to reach the first recharging station in the sequence is stored in $\overleftarrow{Y}(\sigma)$. Second, $\overrightarrow{Y}(\sigma)$ stores the energy needed to get from the last recharging station to the last node in the sequence. In sequences with no recharging station both values are set to the total energy needed to travel through the sequence. Third, in $EY(\sigma)$ we store the overall violation of charge in a sequence, i.e., how much extra energy would be additionally needed to satisfy the constraint.

$$\overleftarrow{Y}(\sigma_1 \oplus \sigma_2) = \begin{cases} \overleftarrow{Y}(\sigma_1) & \text{if } \exists f \in \sigma_1, f \in F' \\ \overleftarrow{Y}(\sigma_1) + r \cdot d_{\sigma_1\sigma_2} + \overleftarrow{Y}(\sigma_2) - \Delta_Y & \text{otherwise} \end{cases} \tag{2.3}$$

$$\overrightarrow{Y}(\sigma_1 \oplus \sigma_2) = \begin{cases} \overrightarrow{Y}(\sigma_2) & \text{if } \exists f \in \sigma_2, f \in F' \\ \overrightarrow{Y}(\sigma_1) + r \cdot d_{\sigma_1\sigma_2} + \overrightarrow{Y}(\sigma_2) - \Delta_Y & \text{otherwise} \end{cases} \tag{2.4}$$

$$EY(\sigma_1 \oplus \sigma_2) = EY(\sigma_1) + EY(\sigma_2) + \Delta_Y \tag{2.5}$$

where $\Delta_Y = max\{\overrightarrow{Y}(\sigma_1) + r \cdot d_{\sigma_1\sigma_2} + \overleftarrow{Y}(\sigma_2) - Y, 0\}$. For simplicity we omit the the index of the vehicle type for the the energy consumption r in these equations.

For the concatenation process we first have to calculate the additional violation in the energy we would face when combining two sequences. This value, denoted as Δ_Y, is calculated using the information of the charge needed from the last charging station of the first sequence to the first charging station of the to the second sequence. If such an additional violation exists ($\Delta_Y > 0$), the amount is added to the total violation of the new sequence; see (2.5). To avoid counting violations multiple times throughout concatenations we have to subtract Δ_Y in Equation (2.3) and (2.4) in the cases where the other sequence does not contain a recharging station. This also implies that neither $\overleftarrow{Y}(\sigma)$ nor $\overrightarrow{Y}(\sigma)$ will ever exceed the maximum battery charge.

Time windows. An effective way to relax the time window constraint and efficiently calculate the time window violation was presented by Nagata, Bräysy, and Dullaert (2010) and was enhanced by Schneider, Sand, and Stenger (2013) in the context of VRPTW. A time window penalty (TW) counting the cumulative time needed to travel back in time in order to satisfy the time window constraints is introduced. By 'repairing' the time using this time window penalty, violations in one location do not propagate to later locations. The concept of slack variables presented by Kindervater and Savelsbergh (1997) to evaluate moves efficiently in constant time is used.

Proper concatenation operators for this approach have been presented in Vidal et al. (2014). In addition to the duration $Dur(\sigma)$ and the amount of time window violation $TW(\sigma)$, the earliest $E(\sigma)$ and latest departure $L(\sigma)$ from the first node are stored.

$$Dur(\sigma_1 \oplus \sigma_2) = Dur(\sigma_1) + Dur(\sigma_2) + t_{\sigma_1\sigma_2} + \Delta_{WT} \tag{2.6}$$

$$TW(\sigma_1 \oplus \sigma_2) = TW(\sigma_1) + TW(\sigma_2) + \Delta_{TW} \tag{2.7}$$

$$E(\sigma_1 \oplus \sigma_2) = max\{E(\sigma_2) - \Delta, E(\sigma_1)\} - \Delta_{WT} \tag{2.8}$$

$$L(\sigma_1 \oplus \sigma_2) = min\{L(\sigma_2) - \Delta, L(\sigma_1)\} + \Delta_{TW} \tag{2.9}$$

where $\Delta = Dur(\sigma_1) - TW(\sigma_1) + t_{\sigma_1\sigma_2}$, $\Delta_{WT} = max\{E(\sigma_2) - \Delta - L(\sigma_1), 0\}$ and $\Delta_{TW} = max\{E(\sigma_1) + \Delta - L(\sigma_2), 0\}$

For each concatenation of two sequences σ_1, σ_2, we calculate the additional waiting time Δ_{WT} and the additional time window violation Δ_{TW} introduced by the concatenation. These are used to compute the concatenation values of (2.6)-(2.9) properly. We use Δ as an auxillary variable for the time needed to reach the first node of the second segment σ_2.

With this operator definition we can calculate the time window violations, but we still need to consider charging times and its effect on the time calculation. In Schneider, Stenger, and Goeke (2014) a slack-based approach was presented which we used as a starting point for our new concatenation operator definition. Here the authors utilized the concept of time travelling and showed that for combining two partial routes, $\{u, \ldots, v\}$ and $\{w, \ldots, f, \ldots, h\}$, we only have to recalculate a part of the second route, i.e., until the first recharging station f is encountered. After recalculating the additional charge needed at the recharging station as well as the new arrival time we can check if there is a violation in the time constraints using the new time values. In case there is no recharging station in the second part we do not have to recalculate any data.

2.1.2 Our efficient concatenation operator for the E-VRPTW.

To adapt the modified time window constraint calculation by Schneider, Stenger, and Goeke (2014) for the sequence-based evaluation approach of Vidal et al. (2013a) we need to store additional data in our sequences in order to handle the changed recharging times efficiently. In our approach we store the two sub-sequences resulting from splitting a sequence right after the first recharging station encountered. I.e., for a sequence with at least one recharging station $\sigma = \{u, \ldots, f, v, \ldots, w\}$ we store $\overleftarrow{\sigma} = \{u, \ldots, f\}$, $\overrightarrow{\sigma} = \{v, \ldots, w\}$, where $f \in F'$ is the first recharging station in σ. Furthermore we introduce $\overleftrightarrow{\sigma}$ representing the tupel $\{\overleftarrow{\sigma}, \overrightarrow{\sigma}\}$ for the purpose of easier understanding. For sequences $\sigma = \{u, \ldots, w\}$ without a recharging station we define $\overleftrightarrow{\sigma} = \{\{u, \ldots, w\}, \{\}\}$.

When we concatenate two sequences $\overleftrightarrow{\sigma} = (\overleftrightarrow{\sigma_1} \oplus \overleftrightarrow{\sigma_2})$ we can distinguish between four cases, depending on whether recharging stations exists in the sequences:

- **no recharging station in both sequences.** We can simply link both sequences in a single concatenation operation $\overleftrightarrow{\sigma} = \{\overleftarrow{\sigma_1} \oplus \overleftarrow{\sigma_2}, \{\}\}$
- **no recharging station in σ_1, but one in σ_2.** In this case we consider the additional distance and time travelled in σ_1 and calculate the changes when combining this information with the first part of σ_2, i.e., $\overleftrightarrow{\sigma} = \{\overleftarrow{\sigma_1} \oplus \overleftarrow{\sigma_2}, \overrightarrow{\sigma_2}\}$
- **a recharging station in σ_1 but not in σ_2.** Although the recharging time of the first recharging station in σ_1 will not change we have to update the information of the second part $\overrightarrow{\sigma}$ to account for the reduced energy capacity available after travelling the additional distance of σ_2. The concatenation of both sequences is therefore $\overleftrightarrow{\sigma} = \{\overleftarrow{\sigma_1}, \overrightarrow{\sigma_1} \oplus \overleftarrow{\sigma_2}\}$
- **a recharging station in both sequences.** If both sequences contain recharging stations we need to evaluate the changes in σ_2 using two concatenations. First we calculate the changes in distance and time for σ_2 by concatenating $\overrightarrow{\sigma_1}$ with $\overleftarrow{\sigma_2}$. With the final distance and time bounds stored we evaluate the actual recharging time by concatenating $\overrightarrow{\sigma_1} \oplus \overleftarrow{\sigma_2}$ with $\overrightarrow{\sigma_2}$. This newly acquired information is also the second part of the new sequence, i.e, $\overleftrightarrow{\sigma} = \{\overleftarrow{\sigma_1}, \overrightarrow{\sigma_1} \oplus \overleftarrow{\sigma_2} \oplus \overrightarrow{\sigma_2}\}$.

In summary, the concatenation operation for time windows with recharging are as follows:

$$\overleftrightarrow{\sigma_1} \oplus \overleftrightarrow{\sigma_2} = \begin{cases} \{\overleftarrow{\sigma_1} \oplus \overleftarrow{\sigma_2}, \{\}\} & \text{if } \nexists f \in \sigma_1, \nexists g \in \sigma_2, f, g \in F' \\ \{\overleftarrow{\sigma_1} \oplus \overleftarrow{\sigma_2}, \overrightarrow{\sigma_2}\} & \text{if } \nexists f \in \sigma_1, \exists g \in \sigma_2, f, g \in F' \\ \{\overleftarrow{\sigma_1}, \overrightarrow{\sigma_1} \oplus \overleftarrow{\sigma_2}\} & \text{if } \exists f \in \sigma_1, \nexists g \in \sigma_2, f, g \in F' \\ \{\overleftarrow{\sigma_1}, \overrightarrow{\sigma_1} \oplus \overleftarrow{\sigma_2} \oplus \overrightarrow{\sigma_2}\} & \text{if } \exists f \in \sigma_1, \exists g \in \sigma_2, f, g \in F' \end{cases} \tag{2.10}$$

Equation (2.10) shows that we can build the constraint data of a combination of two sequences using at most two concatenation operations. To calculate the actual values we have to combine the data from $\overleftarrow{\sigma}$ and $\overrightarrow{\sigma}$, i.e., perform an additional concatenation operation $\overleftarrow{\sigma} \oplus \overrightarrow{\sigma}$). This leads to a total of three constant time operations per concatenation. By storing additional information we can evaluate inter-route moves as described in Schneider, Stenger, and Goeke (2014) regardless of whether recharging stations are present or not in constant time without the need to recalculate a subsequence. Furthermore, this approach is not limited to moves of single nodes but sequences of nodes.

2.1.3 Feasibility.

To check the feasibility of a route we can use the data calculated for the corresponding sequence:

$$\mathit{isFeasible}(\sigma^k) = Q(\sigma) \leq Q^k \wedge EY(\sigma^k) \leq 0 \wedge TW(\sigma^k) \leq 0 \tag{2.11}$$

where $k \in V$ is the vehicle type used to calculate the penalties. We note that the calculation of $Q(\sigma)$ is independent of the vehicle type, but the charge and time window penalty are not. Although we assume identical travel times for each vehicle, the time window penalty calculation is influenced by the battery charge constraint, which itself depends on the vehicle type.

Penalizing infeasiblility. For our approach we decided to work with infeasible solutions using penalty values for violations of the constaints (thus relaxing the problem) in the objective function:

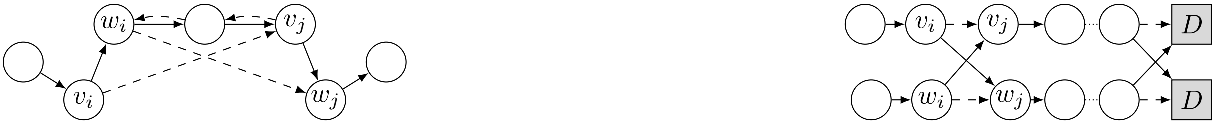
$$\mathit{obj}(\sigma^k) = \mathit{Dist}(\sigma^k) + \rho_Q(Q(\sigma^k) - Q^k) + \rho_{EY} EY(\sigma^k) + \rho_{TW} TW(\sigma^k) \tag{2.12}$$

Each constraint has a corresponding penalty weight ρ which is multiplied with the amount of violation for each of the three relaxed constraints and added to the total distance travelled. We use ρ_Q to weight load capacity violations, ρ_{EY} for energy capacity violations and ρ_{TW} for time window violations.

2.2 Construction and Insertion

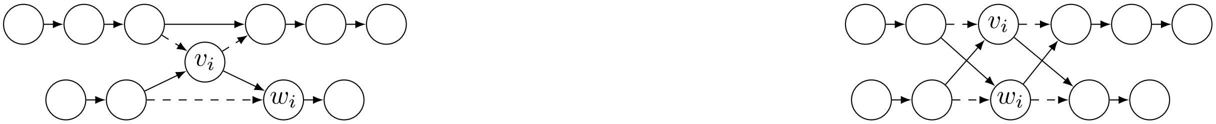
For the construction of an initial solution and later also in the repair step of each iteration of the ALNS, we have to extend partial solutions, i.e., solutions with at least one customer not being assigned to any route, in order to create a complete solution. This is done using different insertion based approaches, i.e., inserting nodes into existing or newly created routes until all nodes have been assigned. To diversify our search, we do not insert the nodes with the lowest cost but select the node and insertion position in each step in a probabilistic way. We use the concept of a *Restricted Candidate List* (RCL) introduced by Hart and Shogan (1987) to reduce the number of positions considered to be selected. Each position in the RCL is selected based on its contribution to the sum of all insertion costs in the RCL.

As recharging stations do not have to be part of a route and can be inserted multiple times, we have implemented a special procedure to insert recharging stations during the insertion and construction phase. Besides trying to insert a node v at position i of a route r, we perform three additional attempts, where we insert a recharging station f right before v, a station g right after v, or both. To determine $f, g \in F'$ we select the recharging station which is reachable with the available energy capacity and which has the smallest additional detour. This approach is myopic



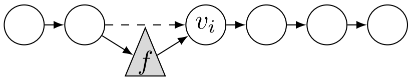
(a) **2-Opt.** Inverting a sequence of at least two node. This example shows the inversion of the sequence between v_j and w_i

(b) **2-Opt*.** Reconnecting a substring of one route at node v_i with a substring of another route at w_j and vice versa.



(c) **Shift.** Moves a single node v_i from the assigned route prior to w_i of another route.

(d) **Swap.** Swaps the two nodes v_i and w_j of different routes.



(e) **InsertRemoveIF.** Inserts or removes a recharging station f prior to v_i.

Figure 2.2: List of all moves used. Solid lines shows the path of the routes after the move is performed and dashed lines the removed edges.

in the sense that only just a subset of recharging stations is considered, and that only the direct effect on the value of the route (and not the effect on future inserts). Furthermore, all previously inserted recharging stations are kept unchanged.

Similar to the work of Paraskevopoulos et al. (2008) and Repoussis and Tarantilis (2010) we implemented a construction heuristic where in each iteration a route for each vehicle type is created using only unassigned nodes until the capacity constraint is violated. These routes are constructed independently, i.e., nodes can be used in multiple routes, and the route with the best lowest cost is added to the current partial solution. We use the *Average Cost per Unit Transfered* (ACUT_k) value defined in Paraskevopoulos et al. (2008) to measure the cost of a tour. This value represents the relative costs of serving the demand of a route σ of vehicle type k (see Equation 2.13).

$$ACUT_k(\sigma) = (f^k + Dist(\sigma))/(Q(\sigma)) \tag{2.13}$$

The remaining routes are dismissed and the next iteration is performed until no further unassigned nodes are available.

2.3 Neighbourhoods and Local Search

Our approach combines the general search methodology of ALNS with a local search method to intensify the search in each iteration. We use well-known as well as problem specific neighbourhoods. First we present the moves defining these neighbourhods followed by a description of the local search method applied.

2-Opt. Figure 2.2a. This intra-route move (i.e., operation on a single route) optimizes a route by inverting a subsequence. As shown in the figure, this move involves the deletion and reinsertion of two edges as well as the inversion of the directions between the nodes of the selected sequence. To further reduce the neighbourhood size and thus the runtime we consider only moves including subsequences of size two at maximum.

2-Opt*. Figure 2.2b. The 2-Opt* move removes two edges of two distinct routes and reconnects the remaining sequences as shown in the figure. An additional cross is made at the end of the route to ensure that the depots are assigned to the same node after the move is performed.

Shift. Figure 2.2c. In this move we shift a single node from one route in between nodes of a different route.

Swap. Figure 2.2d. This moves switches the position of two nodes assigned to different routes.

InsertRemoveIF. Figure 2.2e. In this move we try to insert a recharging station prior to a node v_i in order to improve the value of a route due to repairing violations of the battery capacity constraint, or to remove an existing recharging station from the route. This problem specific move is the only one capable to insert and/or remove an optional recharging station in our local search procedure.

Resize. In all moves described so far, we only changed node assignments or positions but never the vehicle type assignment. The Resize move changes the vehicle assigned without changing the route itself.

ShiftAndResize. This move works like the *Shift* move described before, but with an additional procedure performed after shifting a node. If a node v is moved from a route r_i to r_j we try to change the vehicle type to r_i and r_j at the same time in order to benefit from either lower costs from selecting a cheaper vehicle type or reducing the penalty of constraint violations due to the change to a vehicle type with more capacities.

2.3.1 Local Search.

We use a *Cyclic Neighbourhood Search* procedure as described in Hiermann et al. (2013). Using a list of neighbourhoods, we traverse it in a cyclic manner (i.e, after reaching the end of the list, the search starts again from the beginning), where each one is searched until no further improvement is found. The procedure terminates if a local optimum is reached in each neighbourhood (i.e., no improvement can be found for any neighbourhood). This search approach is similar to the so-called *token-ring search* by Di Gaspero and Schaerf (2002). As improvement strategy we use a *best of 50*-policy, where we apply the best of the first 50 improving moves encountered during a single search iteration. In this way the search terminates faster than with a *best improvement*-policy, but it still provides higher quality solutions than *first improvement*. Preliminary results showed that shuffling the list of neighbourhoods at the beginning of the local search call improves the overall performance.

2.3.2 MakeFeasible.

In our approach we use varying penalty costs to guide our search through the infeasible search space. To reach a feasible solution using our existing tools we use a similar approach as described in Vidal et al. (2014). In our procedure we try to find a feasible solution twice by multiplying each penalty costs by 100 and call the local search procedure. If the solution is still not feasible, the penalty costs are further multiplied by ten each and the local search is called again. No further attempts are made to make the solution feasible, i.e. solutions might still be infeasible after applying this repair operator. However, using this approach we are often able to find feasible regions close to the infeasible solution at low costs.

2.3.3 Reducing neighbourhood size.

Our approach depends heavily on the embedded local search procedure which in turn uses a number of neighbourhoods to improve the search. As the number of neighbours is large and computationally expensive, we use a pruning approach as described in Vidal et al. (2013a) to reduce the number of neighbours considered. In this approach we calculate a set of so-called *promising* arcs for each customer node. This set is then used to prohibit introducing arcs into the solution that are not in this set. Equation (2.14) is used to calculate the so-called *customer correlations* measure.

$$\begin{aligned}\gamma(u,v) = d_{uv} + \gamma^{WT} \cdot max(e_v - s_u - t_{uv} - l_u, 0) \\ + \gamma^{TW} \cdot max(e_u + s_u + t_{uv} - l_v, 0)\end{aligned} \tag{2.14}$$

For a customer v_i not only the direct distance, but fractions of the waiting (γ^{WT}) and time window violation (γ^{TW}) are considered as well. The final set of *promising* arcs $\Gamma(u)$ consists of the $|\Gamma|$ closest customers v with respect to the correlation measure $\gamma(u,v)$. We used the same settings as in Vidal et al. (2013a), setting $|\Gamma| = 60$, $\gamma^{WT} = 0.2$, and $\gamma^{TW} = 1.0$.

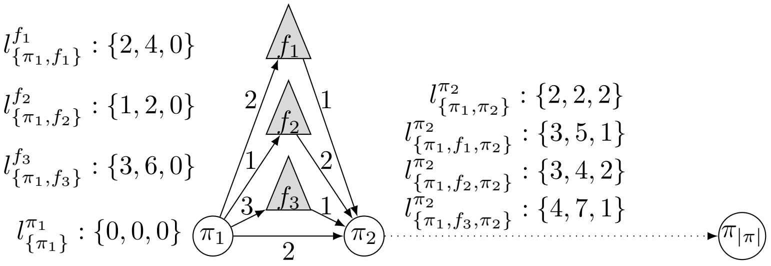
2.4 Labelling Algorithm

We decided to handle recharging stations explicitly, i.e., we add stations as entries in our routes directly. As our construction heuristics have a rather narrow view on the positioning of recharging stations in order to keep the runtime low, we use a post-processing procedure to improve the selection and positioning of recharging stations in a route of fixed visiting orders.

The method we implemented is a labelling algorithm similar to the work of Desrochers, Desrosiers, and Solomon (1992) for a *Shortest Path Problem with Resource Constraints* (SPPRC) in the context of VRP. Feillet et al. (2004) used the more restricted formulation of an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC), presenting a modified labelling algorithm with strong dominance checks. To work with a replenishable resource, Akca et al. (2010) modified the ESPPRC labelling in order to solve the VRP with multiple trips. The replenishable resource is the load capacity which can be replenished in the depot. In our problem we look at the state of charge as replenishable resource which can be replenished in any recharging station. However, this recharging process needs an amount of the available time resource based on the energy consumed thus far (see Section 1).

The general labelling algorithm is applied on a graph where labels are assigned and extended to other nodes. After each extension, a dominance check is applied to remove dominated labels. This is done iteratively until no label can be extended or a terminating condition is reached.

Figure 2.3: Labelling Graph


$l^{f_1}_{\{\pi_1,f_1\}}: \{2,4,0\}$
$l^{f_2}_{\{\pi_1,f_2\}}: \{1,2,0\}$
$l^{f_3}_{\{\pi_1,f_3\}}: \{3,6,0\}$
$l^{\pi_1}_{\{\pi_1\}}: \{0,0,0\}$
$l^{\pi_2}_{\{\pi_1,\pi_2\}}: \{2,2,2\}$
$l^{\pi_2}_{\{\pi_1,f_1,\pi_2\}}: \{3,5,1\}$
$l^{\pi_2}_{\{\pi_1,f_2,\pi_2\}}: \{3,4,2\}$
$l^{\pi_2}_{\{\pi_1,f_3,\pi_2\}}: \{4,7,1\}$
f_1
f_2
f_3
π_1
π_2
$\pi_{|\pi|}$
2
1
1
2
3
1
2

The labelling algorithm for finding the optimal assignment of recharging stations in a given route $\pi = \pi_1, \pi_2, \ldots, \pi_{|\pi|}$ is illustrated in Figure 2.3. In this example, we define the service times in all nodes to be zero, the travel times equal to the travel distances with a recharging and consumption rate of $r = g = 1$. The maximum energy level is set to a value such that it does not affect the example, i.e., $Y \geq 3$.

2.4.1 Label definition.

A label $\{R_{dist}, R_t, R_y\}$ consists of three resources. The distance covered since the start of the tour is counted in R_{dist}. R_t contains the earliest departure time whereas R_y stores the information about the already consumed energy capacity. Each consecutive node in the route is linked by an arc. Between each pair of nodes (π_i, π_j) we add nodes $f_k \in F_{\pi_i,\pi_j}$ representing recharging stations in F and connect them by adding two arcs per station, i.e., (π_i, f_k) and (f_k, π_j), to the graph. The labelling algorithm starts from the first node in the route π_1 with a single label initialized with zero distance and time (or the earliest possible start time) and zero energy consumption. We use the example shown in Figure 2.3 to explain the procedure in more detail.

2.4.2 Label extension.

In the first iteration we now extend this label along all connected arcs using following rules: A possible successor of a label l^i_p is node j if the path is resource feasible, i.e.,

- $R^i_t + t_{ij} \leq l_j$
- $R^i_y + d_{ij} \cdot r \leq Y$

If we can extend the label to node j we create a new label $l^j_{p'}$ with updated resources and cost values and store it in the open list of node j. The resources are updated as follows depending whether node j

- is a recharge station
  - $R^j_{dist} = R^i_{dist} + d_{ij}$
  - $R^j_t = R^i_t + t_{ij} + g \cdot (R^i_y + d_{ij} \cdot r)$
  - $R^j_y = 0$
- or a customer

- $R^j_{dist} = R^i_{dist} + d_{ij}$
- $R^j_t = \max\{R^i_t + t_{ij}, e_j\} + s_j$
- $R^j_y = R^i_y + d_{ij} \cdot r$

Labels are then extended iteratively throughout the graph until a termination condition is reached.

2.4.3 Label elimination.

To reduce the number of labels we use label elimination methods based on *dominance checks* as described in Feillet et al. (2004). By identifying and removing dominated labels we can reduce the number of labels significantly which directly impacts the overall performance of the algorithm. We define that a label l^i_p dominates label $l^i_{p'}$, if the distance travelled R_{dist}, the earliest departure time R_t and the energy consumption R_y of l^i_p are less than or equal to the corresponding value for $l^i_{p'}$ and at least one of those values is strictly smaller. In the example of Figure 2.3 label $l^{\pi_2}_{\{\pi_1,\pi_2\}}$ dominates $l^{\pi_2}_{\{\pi_1,f_j,\pi_2\}}$ but not $l^{\pi_2}_{\{\pi_1,f_k,\pi_2\}}$ as the energy consumed is not smaller. However $l^{\pi_2}_{\{\pi_1,f_k,\pi_2\}}$ is dominated by $l^{\pi_2}_{\{\pi_1,f_i,\pi_2\}}$ and will be removed.

As we have a graph with a special structure (i.e., a directed simple path) we simplify the extension and dominance checks by using a bucket list of size $\kappa|\pi| - 1$, which is traversed. Each bucket holds a set of labels where bucket i contains the labels of node $\pi_{\lfloor i/2 \rfloor + 1}$, if i is odd or the labels of recharging stations reachable from $\pi_{\lfloor i/2 \rfloor + 1}$ otherwise. Dominance checks are only applied on adding a label to a bucket representing a node of the route (i.e., a bucket with an odd index).

When we reach the last bucket $2|\pi| - 1$ we terminate, select the label in the bucket with the lowest objective value (see Section 2.1) and return the corresponding route if it is better than the route we wanted to optimize. Otherwise the unmodified route is returned. We note that the original route can have a better objective value, as our labelling algorithm ensures the construction of a route satisfying the battery charge constraint. Due to our penalty function the weighted time windows violation cost might dominate the possible gain of satisfying the battery charge constraint.

For our approach we assume that only a single recharging station is ever needed between two consecutive nodes. We are aware that this assumption might prohibit the labelling procedure to reach an optimum. However, preliminary experiments with implementations permitting two or three recharging stations between nodes in π showed that the additional computation time leads to no improvement in the overall performance.

This simple but effective procedure is performed for each route after construction, reinsertion and local search.

2.5 Adaptive Large Neighbourhood Search

We solve the E-FSMVRPTW using *Adaptive Large Neighbourhood Search* (ALNS) as proposed by Ropke and Pisinger (2006), but extended by an intensification mechanism in form of an embedded *Local Search* procedure. When speaking of *Large Neighbourhood Search*, most papers refer to the work of Shaw (1998), which corresponds of the class of *Very Large Scale Neighbourhood search* (VLSN). A general introduction to LNS and its variants and extensions can be found in Pisinger and Ropke (2010). An outline of our approach is shown in Algorithm 1.

Algorithm 1: ALNS main loop

```
Input: initial solution $s$
Output: best feasible solution found $s^*_f$
$s^* \leftarrow s$;
if $isFeasible(s)$ then $s^*_f \leftarrow s$;
$i, i^{lastImp} \leftarrow 0$;
while $i < \eta_{max}$ and $(i - i^{lastImp}) < \eta_{maxNoImp}$ do
    $s' \leftarrow DestroyAndRepair(s)$;
    $s' \leftarrow LocalSearch.improve(s')$;
    $s' \leftarrow Labelling.optimize(s')$;
    if $isBetter(s', s)$ then
        $s \leftarrow s'$;
        if $isBetter(s', s^*)$ then $s^* \leftarrow s'$;
        $s'_f \leftarrow MakeFeasible(s')$;
        if $isFeasible(s'_f)$ and $isBetter(s'_f, s^*_f)$ then $s^*_f \leftarrow s'_f$;
        $i^{lastImp} \leftarrow i$;
    updateScore(s');
    updatePenalty(s');
    if $0 \equiv (i - i^{lastImp} + 1) \mod \eta_R$ then
        $s \leftarrow s^*$;
    if $0 \equiv (i + 1) \mod \eta_L$ then
        $adaptSelectionScore()$;
    $i \leftarrow i + 1$;
return $s^*_f$;
```

In Shaw (1998) the author presents a search procedure for a larger neighbourhood defined by a *destroy* and *repair* operation. A neighbour is a solution which is reachable by removing some nodes using the destroy operator, followed by the repair operator.

Ropke and Pisinger (2006) extend this by using learning mechanisms to bias a selection of a variety of destroy and repair operators effectively. For each phase (destroy or repair) the corresponding operator i is selected using a roulette-wheel based on so-called weights π_i. These weights π_i are changed to adapt the algorithm based on the performance of the selected operators for the last couple of iterations. This adaption is done using exponential smoothing of the observed scores $\bar{\pi}_i$. Similar to Ropke and Pisinger (2006) we increase the score of an operator in the following cases, if the newly generated solution

- is an overall best solution
- has not been accepted before and is better than the current solution
- has not been accepted before and is worse, but was accepted in this iteration

After a learning period (of η_L iterations) this score $\bar{\pi}_i$ is used to update the selection probabilities of operator i using the following equation (cf. Ropke and Pisinger 2006)

$$\pi_{i,j+1} = \phi(\frac{\bar{\pi}_{i,j}}{a_i}) + (1 - \phi)\pi_{i,j} \tag{2.15}$$

The weight $\pi_{i,j+1}$ of heuristic i in the next period $j+1$ is calculated using iteration (2.15). The counted score $\bar{\pi}_{i,j}$ for the last period is divided by the number of times the heuristic has been used a_i, which influences the new score for period $j+1$. The step size $\phi = [0;1]$ measures the influence of the new observation compared to the past. If $\phi = 1$ the score reflects the performance of the previous iteration only.

As described in Section 2.1.3, we relax our constraints during the search in order to also search in infeasible regions, but penalizing infeasible solutions. The penalties are managed using an adaption mechanism as presented in Cordeau, Laporte, and Mercier (2001). However, instead of adapting the penalty weights at each iteration of the embedded local search we control these values within the ALNS iteration. Our embedded intensification procedure is therefore used to improve the newly generated solution towards a local optimum for the given penalty settings. As we move rather slowly towards feasible solutions, we use the repair approach described in Section 2.3.2 to find these earlier in the search.

In our search we keep two types of best solutions. On the one hand we store the best feasible solution and on the other hand we also store the best solution with the current penalty setting. Every time we change the penalty values we also adapt the value of the best solution. This might lead to an objective value for the best solution which is worse than the value for the best feasible solution. In this case we replace the best with the best feasible solution. Similar to Cordeau, Laporte, and Mercier (2001) we multiply the penalty weight by w_i if the constraint i was violated at least once in the last b iterations or divide it by the same value if it was always satisfied.

To avoid searching too deep in infeasible regions and to escape possible local optima we implemented a restarting mechanism. After a certain number of iterations (η_R) with no overall improvement the current solution is reset to the best overall solution found so far. In contrast to other restarting mechanisms, we only change the current solution and no other values (e.g., no change in the penalty values).

We started with an SA approach to handle the acceptance of new solutions generated in an iteration. However, preliminary experiments showed that using an 'accept only improvements' policy yields better results for the given problems. This might be due to the use of adaptive penalties and the restart mechanism, which already contributes to the diversification process sufficiently.

An important aspect of ALNS is the number and functionality of the operators used. In the remainder of this section we will list and describe the important operators implemented and used as well as provide and discuss them in detail.

2.5.1 Destroy operators.

We destroy a solution by removing at least q nodes from the current solution, where q is a random number between $[\zeta_{max}, \zeta_{min}]$.

The ***RandomRemoval*** operator simply removes q nodes from the current solution randomly with equal probability.

RandomAndRelatedRemoval operator is based on the definition of Shaw (1998) where we iteratively select and remove a node at random (with equal probability). Following that we select a node similar node using a RCL of five nodes and roulette-wheel selection and remove it as well. The similarity of nodes is calculated using the relatedness measure (2.14). We continue until at least q nodes have been removed in total.

A similar operator is the ***WorstAndRelatedRemoval***. Here, however, we first select and remove a node based on the detour we need to travel in the solution induced by this node. This is done using a RCL of the five worst nodes where a roulette-wheel is used based on the detour cost. Then up to $q-1$ additional nodes are removed based on their relatedness to the selected node. The relatedness is again calculated using (2.14).

With the ability to remove whole routes the ***InefficientRouteAndNeighbourRemoval*** starts by selecting and removing a route at random using roulette-wheel based on the ACUT (2.13). Then neighbouring routes are selected and removed iteratively – starting from the route with the smallest maximum distance between any pair of nodes – until at least q nodes have been removed in total.

The ***TargetRemoval*** operator is similar to the target operator used by Dell'Amico et al. (2007) but simplified. Instead of trying different combinations of a target route, we only consider a single node, the so-called *target node*. We select the node with the highest contribution to the total distance of its assigned tour. Then we select and remove routes based on the minimum distance from any node of a route to the target node. Starting from the route with the highest minimum distance we destroy them until at least $q-1$ nodes have been removed.

2.5.2 Insertion Based Repair.

The repair operators are all based on insertion heuristics with the ability to insert recharging stations as described in Section 2.2.

The first basic **SequentialInsertion** heuristic we implemented is a myopic but fast approach. Nodes are processed in a sequential manner. One after another we first calculate the insertion cost of each route and position, followed by the selection using a RCL containing the five best options. To handle different vehicle types, we change the vehicle type assignment if necessary (when a constraint is violated) and add the resulting costs to the insertion costs. This is also done in the other insertion heuristics. When used as a repair operator in the ALNS, the sequential order of the insertion depends on removal order of the nodes .

The second heuristic is a **ParallelRegretInsertion** approach. As basic parallel greedy insertion heuristics tend to place 'difficult' nodes late in the process where we do not have many possibilities left. To avoid this behaviour, the regret heuristic uses a so-called *regret* value which represents the expected costs of inserting a node not in this iteration but in a future iteration. Such heuristics have been used by Potvin and Rousseau (1993) for the VRPTW and by Trick (1992) for the Generalized Assignment Problem. The regret value for a node v is obtained as follows. For each route σ_i in the current partial solution we calculate the position and cost c_i of the best insertion (based on the objective value) of v into σ_i. Let $l \in L$ be the index of the routes where L is sorted based on c_l, i.e., $l \leq l'$ if $c_l \leq c_{l'}$. We calculate the regret value considering the k best insertion costs using following equation:

$$\mathit{regret}(v) = \sum_{i=2}^{k} (c_{l_i} - c_{l_1}) \tag{2.16}$$

Our **SemiParallelConstruction** heuristic we used to create our initial solution can be easily modified to work as a repair operator too. Instead of creating a full solution from scratch this approach starts with the partial (destroyed) solution and creates new routes without considering existing ones.

A variant we use, the **SemiParallelInsertion** procedure, extends the previous approach as also existing routes are considered. Instead of creating only new routes, existing routes are extended by inserting unassigned nodes until the capacity constraint is violated.

iterations		ALNS		penalty	
η_{max}	2000	η_R	200	$\rho_Q, \rho_{EY}, \rho_{TW}$	10
		η_L	50	$\rho_Q^{min}, \rho_{EY}^{min}, \rho_{TW}^{min}$	0.1
		ϕ	0.8	$\rho_Q^{max} \rho_{EY}^{max}, \rho_{TW}^{max}$	5000
		$[\zeta_{min}, \zeta_{max}]$	[0.05,0.15]	w_i	1.1

Table 3.1: Parameters of the ALNS with embedded LS/labelling approach

3 Computational Results

In this section we present computational results of the proposed ALNS approach for solving the E-FSMVRPTW. Since this problem has not been solved before, we further present results on benchmark instances of two related problems (E-VRPTW and FSMVRPTW) where algorithms and computational results have been published in the literature. This is to show that the proposed algorithm does not only work well on a new problem for which it was designed but where no benchmark exists (except for CPLEX results for very small instances), but that it is highly competitive with the state-of-the-art for solving related simpler problems without any or much adaptation.

Our experiments were run on a single core of a cluster system with an Intel Core2 Quad CPU Q6600 with 2.40 GHz where a memory of 4 GB RAM is shared between 4 cores, operating on the Linux distribution openSuse 12.1 (Asparagus) 64 Bit. The ALNS with embedded local search and labelling is a single-thread implementation in Java 7 and was run using the Java Runtime Environment 1.7, Update 25 (JRE 7u25). Ten test runs were performed using the parameter setting as shown in Table 3.1 if not stated otherwise.

3.1 Experiments on the E-FSMVRPTW

Benchmark Instances. Our benchmark instances are based on the modified Solomon instances of Schneider, Stenger, and Goeke (2014) and the description of the vehicle type classes of Liu and Shen (1999). The instances of Schneider, Stenger, and Goeke (2014) consists of six data sets varying in the distribution of the customers, i.e., whether they are clustered (C), randomly distributed (R) or a combination of both (RC). Furthermore they are divided in instances with a shorter (1) or a longer (2) scheduling horizon. In Liu and Shen (1999), the vehicle types are defined for each of these instance sets, differing in their acquisition cost. These vehicle types are extended for our problem by energy consumption per km/mile, battery size and recharging rate. The consumption and recharging rate is directly taken from the instances of Schneider, Stenger, and Goeke (2014) and set as the same value for each vehicle type. For the battery size we use the value defined in the E-VRPTW instances as base value and scale it upon the vehicle types rank. Given an instance with $|V|$ vehicle types defined (FSMVRPTW) and the battery size Y for the E-VRPTW, we set the battery size of vehicle type k for the E-FSMVRPTW as follows:

$$Y^k = (1.0 + (s \cdot (k/|V|) - (s/2)) \cdot Y, \tag{3.1}$$

where $s = 0.1 \cdot |V|$. Using this approach we encourage the use of larger vehicles due to their larger battery size which leads to a different fleet composition compared to the non-electrical FSMVRPTW. Although a different choice of the recharging and energy consumption factors may result in more realistic scenarios, we note that this could be easily included in our model and

		CPLEX				init.	ALNS				
type	name	obj	F	mix	t[m]	obj	$\overline{obj}$	obj	F	mix	t[m]
A	c101c5	**857,75**	600	A^2	720,00	867,33	**857,75**	**857,75**	600	A^2	0,28
	c103c5	**476,05**	300	A^1	1,06	493,15	**476,05**	**476,05**	300	A^1	0,21
	c206c5	**1261,88**	1000	A^1	88,93	2262,46	1262,19	**1261,88**	1000	A^1	0,22
	c208c5	**1164,34**	1000	A^1	0,36	**1164,34**	**1164,34**	**1164,34**	1000	A^1	0,18
	r104c5	**327,25**	190	A^1C^1	720,00	393,52	**327,25**	**327,25**	190	A^1C^1	0,21
	r105c5	**346,08**	190	A^1C^1	720,00	422,92	**346,08**	**346,08**	190	A^1C^1	0,19
	r202c5	**609,18**	450	A^1	3,32	618,86	**609,18**	**609,18**	450	A^1	0,16
	r203c5	**645,63**	450	A^1	8,00	1095,63	**645,63**	**645,63**	450	A^1	0,20
	rc105c5	**511,96**	270	A^2B^1	720,00	524,99	**511,96**	**511,96**	270	A^2B^1	0,14
	rc108c5	721,15	360	A^1B^2	720,00	640,51	**532,04**	**532,04**	210	A^1B^1	0,18
	rc204c5	**496,74**	300	A^2	720,00	**496,74**	**496,74**	**496,74**	300	A^2	0,12
	rc208c5	**328,89**	150	A^1	7,51	502,91	**328,89**	**328,89**	150	A^1	0,15
B	c101c5	**377,75**	120	A^2	6,88	387,33	**377,75**	**377,75**	120	A^2	0,23
	c103c5	**236,05**	60	A^1	0,34	287,11	**236,05**	**236,05**	60	A^1	0,15
	c206c5	**461,88**	200	A^1	79,16	662,46	**461,88**	**461,88**	200	A^1	0,16
	c208c5	**364,34**	200	A^1	0,52	**364,34**	**364,34**	**364,34**	200	A^1	0,11
	r104c5	**175,25**	38	A^1C^1	1,61	233,52	**175,25**	**175,25**	38	A^1C^1	0,12
	r105c5	**194,08**	38	A^1C^1	2,08	**194,08**	**194,08**	**194,08**	38	A^1C^1	0,13
	r202c5	**249,18**	90	A^1	3,47	258,86	**249,18**	**249,18**	90	A^1	0,11
	r203c5	**285,63**	90	A^1	5,91	407,45	**285,63**	**285,63**	90	A^1	0,11
	rc105c5	**295,96**	54	A^2B^1	139,02	332,99	**295,96**	**295,96**	54	A^2B^1	0,15
	rc108c5	**313,93**	30	B^1	140,32	432,35	**313,93**	**313,93**	60	B^2	0,17
	rc204c5	**255,55**	70	B^1	89,32	260,35	255,79	**255,55**	70	B^1	0,17
	rc208c5	**208,89**	30	A^1	1,51	247,06	**208,89**	**208,89**	30	A^1	0,14
C	c101c5	**317,75**	60	A^2	3,64	340,04	**317,75**	**317,75**	60	A^2	0,17
	c103c5	**206,05**	30	A^1	0,21	233,84	**206,05**	**206,05**	30	A^1	0,15
	c206c5	**361,88**	100	A^1	43,18	445,98	**361,88**	**361,88**	100	A^1	0,16
	c208c5	**264,34**	100	A^1	0,72	265,55	**264,34**	**264,34**	100	A^1	0,15
	r104c5	**156,25**	19	A^1C^1	0,69	213,52	**156,25**	**156,25**	19	A^1C^1	0,14
	r105c5	**175,08**	19	A^1C^1	1,08	**175,08**	**175,08**	**175,08**	19	A^1C^1	0,13
	r202c5	**204,18**	45	A^1	2,88	222,43	**204,18**	**204,18**	45	A^1	0,11
	r203c5	**240,63**	45	A^1	2,94	263.46	**240,63**	**240,63**	45	A^1	0,13
	rc105c5	**268,96**	27	A^2B^1	29,75	308,99	**268,96**	**268,96**	27	A^2B^1	0,13
	rc108c5	**283,93**	30	B^2	73,78	408,35	**283,93**	**283,93**	30	B^2	0,17
	rc204c5	**220,55**	35	B^1	27,47	228,06	220,86	**220,55**	35	B^1	0,16
	rc208c5	**193,89**	15	A^1	2,73	212,31	**193,89**	**193,89**	15	A^1	0,13
avg. %							-2,43	-2,44			

Table 3.2: Results for the E-FSMVRPTW instances with 5 customers compared to CPLEX

algorithms. However, changes in the battery size and other energy related factors have to be weighted against each other to avoid vehicle type dominations.

Results on small instances. For the exact approach we solve the smaller sets created by Schneider, Stenger, and Goeke (2014) using the commercial solver CPLEX 12.2 with the model and modifications described in Section 1. The size of these instances are either 5, 10 or 15 customers with 2 to 8 recharging stations. We assumed from the results of Schneider, Stenger, and Goeke (2014) that some of the instances will require very large runtimes, thus we set a time limit of 12 hours.

Tables 3.2-3.4 present the comparison of the proposed ALNS with CPLEX 12.2. We show the objective value (*obj*) and the included sum of the acquisition costs (F) for the fleet composition specified in the column *mix* as well as the runtime in minutes ($t[m]$). The fleet composition is presented as in Repoussis and Tarantilis (2010), where the vehicle type is referenced using capital letters (starting with the cheapest vehicle type using the letter 'A') and the number of vehicles used shown superscript after the corresponding letter. The objective of the initial solution (i.e., the

		CPLEX				init.	ALNS				
type	name	obj	F	mix	t[m]	obj	$\overline{obj}$	obj	F	mix	t[m]
A	c101c10	-	-		720,00	**1302,15**	**1302,15**	**1302,15**	900	A^3	0,24
	c104c10	1424,28	1100	A^1B^1	720,00	917,07	**902,71**	**902,71**	600	A^2	0,30
	c202c10	3327,39	3000	A^1C^1	720,00	2268,00	1493,84	**1304,32**	1000	A^1	0,25
	c205c10	3931,97	3700	A^1D^1	720,00	2693,16	**2631,97**	**2631,97**	2400	A^1B^1	0,24
	r102c10	1058,12	650	A^3E^1	720,00	787,67	**595,34**	**595,34**	320	$A^2B^1C^1$	0,23
	r103c10	546,45	330	A^1C^2	720,00	651,73	478,23	**478,07**	270	$A^1B^1C^1$	0,24
	r201c10	1665,36	900	A^2	720,00	1156,54	1140,45	**1138,38**	900	A^2	0,27
	r203c10	2745,94	2500	D^1	720,00	1424,84	956,50	**952,16**	700	B^1	0,23
	rc102c10	-	-		720,00	1190,05	**1100,27**	**1100,27**	660	$A^1B^2C^1$	0,27
	rc108c10	-	-		720,00	816,19	**693,27**	**693,27**	240	A^4	0,35
	rc201c10	688,12	300	A^2	720,00	842,74	684,88	**684,63**	300	A^2	0,30
	rc205c10	1495,48	1100	C^2	720,00	1246,55	1067,44	**1064,59**	700	A^1C^1	0,29
B	c101c10	-	-		720,00	648,14	**582,15**	**582,15**	180	A^3	0,25
	c104c10	**422,71**	120	A^2	720,00	534,28	**422,71**	**422,71**	120	A^2	0,31
	c202c10	1038,96	480	A^1B^1	720,00	669,77	548,61	**504,32**	200	A^1	0,24
	c205c10	**711,97**	480	A^1B^1	720,00	972,74	**711,97**	**711,97**	480	A^1B^1	0,24
	r102c10	378,42	58	A^1B^3	720,00	507,67	**325,19**	**325,19**	76	$A^1B^1D^1$	0,24
	r103c10	**262,07**	54	$A^1B^1C^1$	720,00	335,48	**262,07**	**262,07**	54	$A^1B^1C^1$	0,29
	r201c10	440,10	180	A^2	720,00	471,79	**418,38**	**418,38**	180	A^2	0,28
	r203c10	468,56	90	A^1	720,00	496,63	438,29	**392,16**	140	B^1	0,23
	rc102c10	-	-		720,00	626,05	**572,27**	**572,27**	132	$A^1B^2C^1$	0,23
	rc108c10	-	-		720,00	538,15	**422,12**	**422,12**	72	A^1B^2	0,26
	rc201c10	556,79	190	A^1D^1	720,00	476,30	**429,17**	**429,17**	90	A^3	0,45
	rc205c10	561,56	170	A^2C^1	720,00	575,21	**504,59**	**504,59**	140	A^1C^1	0,38
C	c101c10	698,22	165	A^1C^1	720,00	536,90	**492,15**	**492,15**	90	A^3	0,27
	c104c10	372,60	60	A^2	720,00	419.47	**362,71**	**362,71**	60	A^2	0,28
	c202c10	550,83	300	A^1C^1	720,00	449,44	413,84	**404,32**	100	A^1	0,24
	c205c10	**471,97**	240	A^1B^1	720,00	499,03	**471,97**	**471,97**	240	A^1B^1	0,17
	r102c10	**287,19**	38	$A^1B^1D^1$	720,00	472,67	**287,19**	**287,19**	38	$A^1B^1D^1$	0,20
	r103c10	**235,07**	27	$A^1B^1C^1$	720,00	299,73	**235,07**	**235,07**	27	$A^1B^1C^1$	0,34
	r201c10	584,62	210	A^2D^1	720,00	342,99	**328,38**	**328,38**	90	A^2	0,29
	r203c10	476,38	70	B^1	720,00	378,32	350,29	**322,16**	70	B^1	0,22
	rc102c10	-	-		720,00	548,05	**498,51**	**498,51**	75	B^3C^1	0,22
	rc108c10	454,83	33	A^3B^1	720,00	484,44	**386,12**	**386,12**	36	A^1B^2	0,25
	rc201c10	419,92	45	A^3	720,00	426,96	**384,17**	**384,17**	45	A^3	0,32
	rc205c10	434,59	70	A^1C^1	720,00	451,47	**429,69**	**429,69**	95	A^1D^1	0,27
avg. %							-29,98	-31,16			

Table 3.3: Results for the E-FSMVRPTW instances with 10 customers compared to CPLEX

improved solution after construction using LS and the labelling procedure) is shown in the column *init.* For the ALNS solutions we present these values for the best solution (obj) as well as the average objective function ($\overline{obj}$) of all ten runs.

CPLEX is able to find solutions for all of the instances with 5 customers (Table 3.2), although it is not able to prove optimality in six cases. Our ALNS approach is able to obtain optimal solutions in all cases where the optimum has been proven by CPLEX, and finds a better solution in one of the six instances where optimality is not proven. Note that these very small instances are apparently quite easy, so that even the average value of ALNS coincides with the optimal (or best known) solution in almost all cases, implying that in each of the ten runs this best known solution was found.

For the larger instances (Table 3.3 and Table 3.4), CPLEX is not able to prove optimality for any of its solutions, and our proposed ALNS did find equal or better solutions for all instances.

These results clearly demonstrate the quality of our proposed algorithm, since it is able to find proven optimal solutions (where available) with short computational effort and clearly outperform

		CPLEX				init.	ALNS				
type	name	obj	F	mix	t[m]	obj	$\overline{obj}$	obj	F	mix	t[m]
A	c202c15	6136,74	5700	$A^1C^1D^1$	720,00	3419,19	**2403,35**	**2403,35**	2000	A^2	0,64
	c208c15	-	-		720,00	2327,88	**2325,89**	**2325,89**	2000	A^2	0,53
	r102c15	-	-		720,00	943,28	854,99	**850,58**	390	A^3B^3	0,50
	r105c15	-	-		720,00	889,40	762,48	**760,13**	380	B^3C^1	0,41
	r202c15	4619,56	4150	$A^1C^1D^1$	720,00	1397,12	1316,84	**1311,24**	900	A^2	0,92
	r209c15	-	-		720,00	2029,22	**1033,50**	**1033,50**	700	B^1	0,64
	rc202c15	5667,56	5050	$A^2B^1D^1E^1F^1$	720,00	1434,09	**1101,61**	**1101,61**	700	A^1C^1	0,56
	rc204c15	-	-		720,00	925,38	**810,90**	**810,90**	500	A^1B^1	0,54
B	c202c15	-	-		720,00	1020,57	**803,35**	**803,35**	400	A^2	0,61
	c208c15	-	-		720,00	927,88	**725,89**	**725,89**	400	A^2	0,47
	r102c15	-	-		720,00	580,80	514,84	**511,55**	90	$A^3B^2C^1$	0,56
	r105c15	1228,94	604	$A^1B^1C^1D^3E^4$	720,00	566,99	**436,89**	**436,89**	98	B^3D^1	0,43
	r202c15	-	-		720,00	673,31	**591,24**	**591,24**	180	A^2	0,74
	r209c15	-	-		720,00	709,59	**473,50**	**473,50**	140	B^1	0,63
	rc202c15	1799,42	1150	$C^1D^2E^1F^1$	720,00	586,78	**541,61**	**541,61**	140	A^1C^1	0,51
	rc204c15	-	-		720,00	563.39	**410,90**	**410,90**	100	A^1B^1	0,59
C	c202c15	-	-		720,00	693,34	607,82	**603,35**	200	A^2	0,61
	c208c15	-	-		720,00	627,21	526,29	**525,89**	200	A^2	0,38
	r102c15	-	-		720,00	551,82	467,00	**465,94**	46	$A^2B^1C^2$	0,51
	r105c15	706,41	228	$C^2D^4E^2$	720,00	552,71	387,94	**387,89**	49	B^3D^1	0,38
	r202c15	-	-		720,00	568,08	**495,64**	**495,64**	115	A^1B^1	0,55
	r209c15	-	-		720,00	486,61	**403,50**	**403,50**	70	B^1	0,55
	rc202c15	-	-		720,00	567,12	**471,61**	**471,61**	70	A^1C^1	0,44
	rc204c15	-	-		720,00	585,51	**360,90**	**360,90**	50	A^1B^1	0,52
avg. %							-61,14	-61,15			

Table 3.4: Results for the E-FSMVRPTW instances with 15 customers compared to CPLEX

CPLEX for the instances where no optimal solutions have been found and proven.

Results on larger instances. In order to further evaluate the performance of our algorithm, we conduct experiments on the larger benchmark instances (as described earlier in this section). We test four settings of our approach denoted as

- $ALNS_{2000}$ as our default settings, with a limit of 2000 iterations,
- $ALNS^{800}_{1500}$ using a limit of 1500 iterations or 800 iterations without an improvement,
- $ALNS_{800}$ with a limit of 800 iterations and
- $\overline{ALNS}^{800}_{1500}$ where we use the same termination criteria as $ALNS^{800}_{1500}$, but without calling the labelling algorithm described in Section 2.4

Table 3.5 shows the average deviations from the best solutions we have been able to find in ten runs ($\overline{obj}$) as well as the average runtime in minutes ($t[m]$) grouped by instance type (C,R,RC).

As expected the best average performance is achieved using a setting with a higher iteration limit ($ALNS_{2000}$) which itself results in higher runtime (~25 minutes on average). However, terminating earlier ($ALNS^{800}_{1500}$) reduces the runtime by around one third while keeping the average solution quality on a high level. Although almost 2% worse than the best known solutions the fast variant ($ALNS_{800}$) is around 60% faster than the default variant, on average.

When comparing the fast approach $ALNS^{800}_{1500}$ including the labelling procedure with the version without labelling, one can observe that, on average, the labelling procedure increases solution

		$ALNS_{2000}$		$ALNS^{800}_{1500}$		$ALNS_{800}$		$\overline{ALNS}^{800}_{1500}$	
type	name	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]
A	C1	0,24	17,68	0,28	12,80	0,44	7,35	0,45	13.64
	C2	0,39	42,54	0,61	24,36	0,61	19,21	0,78	26,58
	R1	0,89	15,51	1,24	11,64	1,69	6,49	1,34	12,60
	R2	0,87	45,90	1,12	30,76	1,31	18,03	1,12	33,70
	RC1	1,23	14,92	1,39	11,18	2,21	6,35	1,74	12,08
	RC2	0,61	16,04	0,68	11,98	1,06	6,55	0,86	12,62
	avg.	0,66	25,68	0,84	17,38	1,15	10,70	1,00	18,84
B	C1	0,48	14,68	0,62	10,38	0,78	6,09	0,84	11,23
	C2	1,08	37,05	1,24	18,92	1,29	14,81	1,62	20,73
	R1	1,47	15,19	1,70	11,28	2,49	6,26	1,82	12,39
	R2	1,30	29,48	1,49	20,82	2,20	11,85	1,70	23,19
	RC1	1,40	14,57	1,79	10,73	2,37	6,08	1,92	11,65
	RC2	1,43	18,87	1,54	13,34	1,80	7,55	1,83	14,89
	avg.	1,17	21,47	1,38	14,32	1,82	8,71	1,59	15,77
C	C1	0,43	14,77	0,39	10,45	0,72	6,02	0,56	11,33
	C2	0,89	31,95	0,96	18,50	1,07	13,08	1,23	21,11
	R1	1,59	15,51	1,76	11,44	2,57	6,29	1,90	12,40
	R2	1,70	28,72	1,97	20,22	2,33	11,37	1,99	22,36
	RC1	1,40	14,80	1,76	10,90	2,43	6,08	1,65	11,60
	RC2	1,29	19,71	1,53	13,70	2,01	7,87	1,59	15,26
	avg.	1,23	20,83	1,40	14,26	1,90	8,41	1,49	15,72

Table 3.5: Average deviation from the best known solution and average runtime for all EFSM instances

quality while slightly reducing the overall runtime. This is due to the termination criterion of 800 iterations without improvement. Furthermore a detailed analysis of our test runs shows, that the contribution of the labelling procedure to the overall runtime is only around 1%.

For future reference, we present the results for each of the newly proposed instances in Tables 3.6, 3.7 and 3.8. The second to fourth column show the objective value (obj), the included acquisition costs (F) and the composition of the fleet (mix) of the best solution found throughout all experiments.

3.2 Solution of Related Problems

Although our ALNS shows excellent behaviour in small instances of the E-FSMVRPTW, where CPLEX results are available, it would be useful to further assess its competitiveness on larger instances. Since this is not possible for the E-FSMVRPTW, where no other benchmark results exist, we solve the benchmark results for the related but simpler classes E-VRPTW and FSMVRPTW. The aim is not to find new best solutions nor to establish a new state-of-the-art algorithm for these problem classes. Rather, we want to show that the ALNS, that we developed for the E-FSMVRPTW, is also competitive for these other classes without any modification (or with minimal modifications because of a different objective function) and without and tuning for the special structure of these subclasses.

3.2.1 Experiments on the E-VRPTW.

For the E-VRPTW, as formulated in Schneider, Stenger, and Goeke (2014), we have to make some small modifications, since it has a different objective. More precisely, it has a hierarchical objective function, where the number of vehicles used is optimized first before minimizing the overall distance travelled. Like in Ropke and Pisinger (2006), we adapted our approach accordingly using a two phase approach. First we introduce an additional constraint to bound the number of vehicles used. This constraint is relaxed such that violations are penalized in the objective value using a large number. In the first phase we set the number of available vehicles to the number of vehicles used in the initial solution minus one making the initial solution infeasible. Every time we find a new feasible solution (which has to have at most the defined upper bound of vehicles used) we decrease the number of vehicles available by one until the current feasible solution is infeasible again. This is done until we cannot find a feasible solution any more for a certain number of iterations. In our experiments, we use a value of 400 which was obtained as a good and robust choice in preliminary tests. At this point, we set the number of available vehicles to the number used in the last feasible solution found and start the second phase where we optimize the solution until a termination criteria is reached.

Results. Table 3.9 presents the results compared to the best results found in Schneider, Stenger, and Goeke (2014). We show the number of vehicles needed (m) and the corresponding objective value (obj) for the best solution obtained in ten runs as well as the average runtime ($\overline{t[m]}$). Furthermore we present the difference in the objective value of the best ($\Delta(obj)$) solution if the the same number of vehicles is used. For all but two instances (r105,r109) we were able to find a solution with the lowest known number of vehicles needed and for one instance (rc104) our approach was able find a solution with less vehicles than the previously best known solution. In summary, our approach was able to find twelve new best known solutions when considering the primary and secondary objective.

With this experiment, we have been able to demonstrate that our approach is not only very well suited for the problem we planned to solve originally, but is also competitive with the state-of-the-art for solving a related simpler problem without much adaptation. The only necessary change was using the two phase approach because of the different objective function, but no change in the operators or parameters is made.

3.2.2 Experiments on the FSMVRPTW.

To solve the FSMVRPTW we simply set the energy consumption of every edge to zero, therefore no recharging is needed. Furthermore, as the set of available recharging stations is empty, no attempts are made to insert such into a route by a construction heuristic or the labelling algorithm. However, no changes were made on the structure of the used neighbourhoods and on the move evaluation. This means that although we could have exploited a property of the FSMVRPTW – a change of the vehicle type does only require a re-evaluation of the capacity constraint, which can be done in constant time – we still use the neighbourhoods described in Section 2.3.

Results. The results in Table 3.10 show that our approach is competitive and performs especially well for the clustered instances (c-instances). We compared our results with the approaches of Bräysy et al. (2008) (BDHMG08), Repoussis and Tarantilis (2010) (RT10) and Vidal et al. (2014) (VCGP13). The best run of each instance is less than 1% worse than the best known solution. These results further underline the competitiveness of our approach, since it can reach high quality solutions without being especially tailored to solving the FSMVRPTW.

4 Conclusion

In this paper, we introduced a new fleet composition and routing problem using electric vehicles: the E-FSMVRPTW considers multiple vehicle types differing in their capacities and costs. Each vehicle has the possibility of recharging on tour using recharging stations. Each recharge operation consumes an amount of time depending on the distance travelled and the resulting battery charge. This adds additional complexity as time windows at customer locations are considered as well.

We proposed a hybrid solution method based on *Adaptive Large Neighbourhood Search* (ALNS) extended with an embedded intensification mechanism using intelligent local search for route improvement and efficient labelling procedures for the optimal placement of charging stations. For defining the problem rigorously, we also provided a MIP formulation and proposed some preprocessing tricks to make its solution easier for a commercial solver. We constructed a new benchmark set for the E-FSMVRPTW and performed extensive computational experiments with CPLEX and our extended ALNS. On small instances, where CPLEX was able to find proven optimal solutions, we were always able to find these optimal solutions with our extended ALNS in much shorter time. On all other smaller instances, where CPLEX only could find feasible solutions, the extended ALNS was always able to equal or outperform these. We also presented results of our algorithm on larger instances, comparable in size to those of benchmark instances published for related problems. Finally, we applied the proposed approach also to two related sub-problems, namely to the E-VRPTW and the FSMVRPTW. The results of our experiments demonstrated the competitiveness of our approach. We were able to discover twelve new best known solutions for the E-VRPTW. Future work will focus on extending and applying our model to real world cases, where additional constraints must be added.

Acknowledgments

Thanks are due to our colleagues at the Austrian Institute of Technology and the University of Vienna as well as the participants of conference VeRoLog 2013 and the workshop AWGM 2013 (where preliminary results were presented) for useful discussions and suggestions.

This work is partially funded by the Austrian Climate and Energy Fund within the "Electric Mobility Flagship Projects" programme under grant 834868 (project VECEPT).

References

Akca Z, Berger RT, Ralphs TK (2010) Solution methods for the multi-trip elementary shortest path problem with resource constraints. Technical report, COR@L Laboratory, Lehigh University, Bethlehem, Pennsylvania.

Baldacci R, Battarra M, Vigo D (2008) Routing a heterogeneous fleet of vehicles. Bruce Golden, Subramanian Raghavan, Edward Wasil, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges*, *Operations Research/Computer Science Interfaces*, vol. 43 (Springer, US), 3–27.

Bräysy O, Dullaert W, Hasle G, Mester DI, Gendreau M (2008) An effective multirestart deterministic annealing metaheuristic for the fleet size and mix vehicle-routing problem with time windows. *Transportation Science* 42(3):371–386.

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

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

Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* 12(4):568–581.

Conrad RG, Figliozzi MA (2011) The recharging vehicle routing problem. T Doolen, E Van Aken, eds. *Proceedings of the 2011 Industrial Engineering Research Conference.* (Reno, Nevada).

Cordeau J-F, Laporte G, Mercier A (2001) A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society* 52(8):928–936.

Crevier B, Cordeau J-F, Laporte G (2007) The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research* 176(2):756–773.

Dell'Amico M, Monaci M, Pagani C, Vigo D (2007) Heuristic approaches for the fleet size and mix vehicle routing problem with time windows. *Transportation Science* 41(4):516–526.

Desrochers M, Desrosiers J, Solomon M (1992) A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* 40(2):342–354.

Di Gaspero L, Schaerf A (2002) Multi-neighbourhood local search with application to course timetabling. Burke E, De Causmaecker P, eds. *Practice and Theory of Automated Timetabling IV* (Springer, DE), 262–275.

Dullaert W, Janssens GK, Sörensen K, Vernimmen B (2002) New heuristics for the fleet size and mix vehicle routing problem with time windows. *Journal of the Operations Research Society* 53(11):1232–1238.

Erdoğan S, Miller-Hooks E (2012) A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review* 48(1):100–114.

Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* 44(3):216–229.

Glover F (1996) Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics* 65(1-3):223–253.

Golden B, Assad A, Levy L, Gheysens F (1984) The fleet size and mix vehicle routing problem. *Computers & Operations Research* 11(1):49–66.

Hart JP, Shogan AW (1987) Semi-greedy heuristics: an empirical study. *Operations Research Letters* 6(3):107–114.

Hiermann G, Prandtstetter M, Rendl A, Puchinger J, Raidl GR (2013) Metaheuristics for solving a multimodal home-healthcare scheduling problem. *Central European Journal of Operations Research*, Online Published.

Kindervater GAP, Savelsbergh MWP (1997) Vehicle routing: Handling edge exchanges. Aarts E, Lenstra JK, eds., *Local Search in Combinatorial Optimization*. (John Wiley & Sons Ltd) 337–360.

Liu F-H, Shen S-Y (1999) The fleet size and mix vehicle routing problem with time windows. *The Journal of the Operational Research Society* 50(7):721–732.

Nagata Y, Bräysy O, Dullaert W (2010) A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research* 37(4):724–737.

Paraskevopoulos DC, Repoussis PP, Tarantilis CD, Ioannou G, Prastacos GP (2008) A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. *Journal of Heuristics* 14(5):425–455.

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. *Computers & Operations Research* 34(8):2403–2435.

Pisinger D, Ropke S (2010) Large neighborhood search. Gendreau M, Potvin J-Y, eds. *Handbook of Metaheuristics*, *International Series in Operations Research & Management Science*, vol. 146 (Springer US) 399–419.

Potvin J-Y, Rousseau JM (1993) A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research* 66(3):331 – 340.

Repoussis PP, Tarantilis CD (2010) Solving the fleet size and mix vehicle routing problem with time windows via adaptive memory programming. *Transportation Research Part C: Emerging Technologies* 18(5):695–712.

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

Schneider M, Sand B, Stenger A (2013) A note on the time travel approach for handling time windows in vehicle routing problems. *Computers & Operations Research* 40(10):2564 – 2568.

Schneider M, Stenger A, Goeke D (2014) The electric vehicle routing problem with time windows and recharging stations. *Transportation Science* (forthcoming).

Schrimpf G, Schneider J, Stamm-Wilbrandt H, Dueck G (2000) Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics* 159(2):139–171.

Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. Maher MJ, Puget JF, eds. *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*. CP '98, (Springer, UK) 417–431.

Solomon M (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* 35(2):254–265.

Tarantilis CD, Zachariadis EE, Kiranoudis CT (2008) A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *INFORMS Journal on Computing* 20(1):154–168.

Toth P, Vigo D (2001) *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, US.

Trick MA (1992) A linear relaxation heuristic for the generalized assignment problem. *Naval Research Logistics (NRL)* 39(2):137–151.

Vidal T, Crainic TG, Gendreau M, Prins C (2013a) A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research* 40(1):475–489.

Vidal T, Crainic TG, Gendreau M, Prins C (2013b) Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. *European Journal of Operational Research* 231(1):1–21.

Vidal T, Crainic TG, Gendreau M, Prins C (2014) A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research* 234(3):658-673.

	BKS			$ALNS_{2000}$		$ALNS^{800}_{1500}$		$ALNS_{800}$		$\overline{ALNS}^{800}_{1500}$	
name	obj	F	mix	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]
c101	7180,42	5700	A^{19}	7190,21	17,53	7199,54	12,67	7211,10	7,34	7212,68	13,24
c102	7150,78	5700	A^{19}	7162,24	17,95	7162,92	12,83	7173,29	7,59	7179,40	13,13
c103	7119,62	5700	A^{19}	7149,31	18,30	7144,56	12,99	7158,31	7,62	7164,42	13,94
c104	7100,22	5700	A^{19}	7110,43	17,75	7117,83	13,13	7124,70	7,36	7126,23	14,94
c105	7155,23	5700	A^{19}	7182,56	17,60	7176,08	12,72	7199,86	7,19	7197,98	11,94
c106	7146,88	5700	A^{19}	7168,94	17,69	7165,14	13,06	7186,77	7,39	7173,99	13,90
c107	7156,18	5700	A^{19}	7171,04	17,42	7181,60	12,47	7182,94	7,19	7188,87	13,40
c108	7139,12	5700	A^{19}	7153,77	17,53	7161,47	12,37	7166,20	7,23	7169,37	14,44
c109	7120,33	5700	A^{19}	7132,19	17,32	7141,76	12,95	7148,94	7,26	7146,34	13,82
c201	5737,57	5000	A^{5}	5757,53	39,28	5767,45	18,98	5750,87	15,75	5780,19	21,30
c202	5741,45	5000	A^{5}	5765,52	42,22	5790,78	20,72	5787,04	17,86	5806,13	20,20
c203	5717,36	5000	A^{5}	5751,99	47,69	5754,77	22,55	5750,62	20,38	5770,94	26,56
c204	5696,93	5000	A^{5}	5727,18	50,91	5742,70	28,83	5736,29	20,20	5727,46	29,96
c205	5703,48	5000	A^{5}	5725,41	46,16	5726,77	24,31	5729,38	18,30	5752,89	22,74
c206	5708,77	5000	A^{5}	5714,39	31,20	5732,56	25,46	5740,69	19,93	5741,86	29,28
c207	5697,99	5000	A^{5}	5713,44	32,58	5732,52	24,58	5744,51	20,19	5738,00	33,97
c208	5681,47	5000	A^{5}	5707,65	50,23	5715,82	29,45	5724,20	21,08	5723,91	28,59
r101	4415,73	2550	$A^{1}B^{12}C^{11}$	4465,51	14,47	4473,74	10,78	4504,93	6,06	4460,07	11,81
r102	4233,29	2610	$B^{5}C^{14}D^{1}$	4270,92	14,97	4287,24	11,27	4318,60	6,28	4291,37	12,27
r103	4095,80	2570	$A^{1}C^{18}$	4130,86	16,43	4144,65	11,92	4175,26	6,67	4139,84	12,68
r104	4000,32	2570	$A^{1}C^{18}$	4025,60	15,27	4054,87	11,26	4071,91	6,51	4063,41	12,22
r105	4181,80	2560	$B^{4}C^{16}$	4215,34	15,37	4237,02	11,58	4232,93	6,44	4231,93	12,75
r106	4120,23	2620	$A^{1}B^{1}C^{16}D^{1}$	4155,24	15,54	4174,52	11,62	4195,99	6,53	4183,71	12,92
r107	4057,06	2630	$C^{17}D^{1}$	4093,59	15,30	4102,02	11,70	4099,93	6,56	4112,52	12,89
r108	3992,57	2570	$A^{1}C^{18}$	4025,75	15,77	4030,02	12,01	4060,23	6,68	4048,06	13,12
r109	4067,14	2630	$C^{17}D^{1}$	4110,98	15,58	4110,82	12,14	4144,17	6,52	4135,18	12,26
r110	3994,37	2570	$A^{1}C^{18}$	4045,96	15,73	4052,47	12,09	4062,74	6,48	4062,98	12,65
r111	4011,99	2560	$B^{4}C^{16}$	4048,42	15,93	4060,56	11,81	4080,73	6,53	4065,45	12,94
r112	4001,07	2630	$C^{17}D^{1}$	4023,01	15,80	4052,07	11,58	4054,69	6,57	4035,77	12,68
r201	3400,34	2250	A^{5}	3432,83	42,20	3440,95	29,55	3442,07	16,63	3426,17	32,93
r202	3270,49	2250	A^{5}	3295,26	44,95	3313,26	30,40	3308,79	17,38	3319,22	32,83
r203	3136,47	2250	A^{5}	3169,97	49,40	3179,78	33,62	3194,28	20,04	3180,88	35,68
r204	3008,01	2250	A^{5}	3026,09	46,32	3039,57	28,21	3041,16	18,14	3040,51	32,64
r205	3234,26	2250	A^{5}	3261,16	40,89	3265,78	29,55	3274,90	15,47	3262,63	32,74
r206	3172,50	2250	A^{5}	3194,12	47,73	3203,69	32,56	3215,34	18,76	3210,65	34,25
r207	3074,95	2250	A^{5}	3099,52	46,87	3109,59	28,54	3115,36	18,41	3114,94	33,25
r208	3000,90	2250	A^{5}	3026,57	51,26	3029,02	33,82	3037,24	19,49	3045,40	35,16
r209	3142,16	2250	A^{5}	3161,57	45,06	3170,34	32,60	3172,92	17,96	3161,98	35,24
r210	3108,78	2250	A^{5}	3143,79	45,94	3145,32	32,20	3158,22	19,14	3138,76	34,57
r211	3041,93	2250	A^{5}	3079,24	44,29	3082,19	27,35	3082,58	16,88	3077,74	31,44
rc101	5285,41	3300	$A^{5}B^{12}C^{4}$	5346,49	14,13	5356,93	10,76	5401,83	6,13	5376,44	11,31
rc102	5113,03	3270	$A^{7}B^{7}C^{6}$	5180,03	14,63	5175,04	11,01	5249,26	6,24	5220,37	11,73
rc103	4938,98	3390	$A^{4}B^{5}C^{8}$	5007,37	14,53	5024,74	10,75	5032,46	6,25	5029,24	12,01
rc104	4804,00	3450	$B^{3}C^{10}$	4862,65	16,03	4887,62	11,65	4931,23	6,72	4892,75	13,11
rc105	5074,43	3390	$A^{4}B^{7}C^{7}$	5117,09	14,48	5140,53	10,90	5167,74	6,19	5136,26	11,62
rc106	5028,28	3420	$A^{2}B^{8}C^{7}$	5102,46	14,69	5100,85	11,14	5145,54	6,14	5135,71	11,56
rc107	4864,78	3420	$A^{2}B^{4}C^{9}$	4913,90	15,24	4911,14	11,52	4954,15	6,43	4936,09	12,41
rc108	4794,47	3420	$A^{2}B^{2}C^{10}$	4862,41	15,64	4862,96	11,67	4904,73	6,70	4872,28	12,92
rc201	4346,25	2950	$A^{8}B^{5}$	4361,17	14,27	4370,67	10,33	4388,20	5,80	4380,58	11,36
rc202	4262,12	3000	$A^{6}B^{6}$	4295,27	14,59	4291,40	11,03	4316,39	6,10	4299,75	12,21
rc203	4152,94	3000	$A^{7}B^{4}C^{1}$	4186,28	15,98	4183,12	12,41	4196,32	6,47	4190,14	12,61
rc204	4113,49	3150	$A^{3}B^{3}C^{3}$	4127,11	19,18	4142,42	13,92	4160,90	7,91	4139,02	14,08
rc205	4246,52	3000	$A^{6}B^{6}$	4273,59	14,86	4271,95	10,89	4293,49	6,20	4274,25	12,21
rc206	4237,75	3050	$A^{5}B^{5}C^{1}$	4270,25	15,09	4269,08	11,20	4280,67	6,17	4289,22	12,32
rc207	4174,59	3000	$A^{6}B^{6}$	4199,60	16,20	4206,98	12,19	4212,89	6,55	4203,67	13,42
rc208	4097,04	3200	$A^{2}B^{2}C^{4}$	4122,12	18,13	4125,42	13,84	4139,28	7,23	4142,91	12,74
avg.	4772,32			4803,80	25,68	4812,40	17,38	4827,42	10,70	4820,15	18,84
dev.				0,66		0,84		1,15		1,00	

Table 3.6: Results for the E-FSMVRPTW instances, vehicle type A

	BKS			$ALNS_{2000}$		$ALNS^{800}_{1500}$		$ALNS_{800}$		$\overline{ALNS}^{800}_{1500}$	
name	obj	F	mix	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]
c101	2495,00	1340	A^9B^5	2505,73	14,43	2510,03	9,89	2511,82	5,93	2521,93	10,98
c102	2445,99	1340	A^9B^5	2450,73	14,41	2459,10	10,95	2471,49	6,09	2463,39	11,09
c103	2432,80	1340	A^9B^5	2452,40	15,03	2454,83	10,41	2455,80	6,18	2456,42	11,92
c104	2402,62	1340	A^9B^5	2428,95	15,07	2432,06	11,04	2435,81	6,39	2426,59	10,98
c105	2472,93	1340	A^9B^5	2475,95	14,45	2483,49	10,58	2487,32	6,02	2491,52	11,26
c106	2462,03	1340	A^9B^5	2468,13	14,71	2471,19	10,52	2476,74	6,04	2474,89	11,28
c107	2457,89	1340	A^9B^5	2461,32	14,60	2473,08	9,76	2468,52	5,95	2476,14	11,49
c108	2450,17	1340	A^9B^5	2463,02	14,33	2458,76	10,23	2470,58	6,07	2466,86	11,53
c109	2434,32	1340	A^9B^5	2452,57	15,07	2447,36	10,03	2448,33	6,16	2460,38	10,54
c201	1730,41	1040	A^1B^3	1739,26	35,76	1742,86	17,35	1743,11	14,06	1747,30	18,01
c202	1730,41	1040	A^1B^3	1745,24	38,14	1745,51	23,03	1746,33	15,02	1757,06	22,90
c203	1716,29	1000	A^5	1742,76	39,38	1731,67	19,61	1747,97	15,63	1754,78	20,56
c204	1699,07	1040	A^1B^3	1709,43	37,02	1716,82	19,35	1732,55	14,76	1718,45	20,88
c205	1694,25	1000	A^5	1715,09	37,83	1725,49	17,46	1718,00	15,35	1724,49	19,77
c206	1687,96	1000	A^5	1712,38	36,78	1713,99	19,11	1707,63	14,30	1717,58	20,72
c207	1694,61	1000	A^5	1710,70	35,05	1717,25	18,39	1711,27	14,75	1724,56	22,66
c208	1681,47	1000	A^5	1707,11	36,46	1709,43	17,09	1704,07	14,65	1710,62	20,31
r101	2261,21	588	$A^2B^2C^{12}D^4$	2281,28	13,68	2283,72	10,46	2305,21	5,84	2282,12	11,51
r102	2065,81	584	$A^1B^1C^{11}D^5$	2095,87	14,45	2096,91	10,57	2114,41	5,87	2098,26	11,34
r103	1894,98	594	$B^2C^4D^9$	1927,52	15,11	1930,25	11,59	1942,42	6,24	1941,33	12,21
r104	1747,65	628	$C^1D^{10}E^1$	1775,33	15,28	1789,98	11,39	1797,29	6,52	1787,53	12,32
r105	2010,31	584	$B^2C^9D^6$	2030,12	15,08	2054,58	10,60	2061,56	6,08	2041,12	11,91
r106	1934,00	616	$B^2C^3D^{10}$	1963,88	14,51	1957,49	10,70	1980,71	5,97	1973,23	12,00
r107	1821,62	634	$C^3D^9E^1$	1844,20	15,50	1844,76	10,62	1855,17	6,29	1847,97	12,63
r108	1716,50	628	$C^1D^{10}E^1$	1753,09	16,27	1751,12	12,07	1778,66	6,56	1750,23	13,37
r109	1871,54	622	$B^1C^2D^{11}$	1904,12	15,37	1907,18	11,21	1917,77	6,46	1908,84	12,42
r110	1759,69	650	$D^{11}E^1$	1793,48	15,54	1791,53	12,00	1817,38	6,35	1803,09	12,72
r111	1786,97	644	$A^1C^3D^7E^2$	1808,36	15,66	1820,64	12,08	1823,90	6,40	1818,00	13,01
r112	1721,79	650	$D^{11}E^1$	1746,02	15,77	1748,98	12,10	1761,07	6,53	1751,11	13,26
r201	1593,17	450	A^5	1618,25	31,21	1620,45	20,20	1629,67	12,41	1612,41	23,28
r202	1462,46	450	A^5	1479,42	29,61	1485,16	21,12	1494,56	11,85	1485,89	23,31
r203	1327,56	450	A^5	1354,24	30,70	1346,21	20,28	1355,51	12,58	1354,66	25,94
r204	1203,17	450	A^5	1211,63	27,35	1213,70	19,73	1216,56	10,89	1210,71	22,03
r205	1430,70	450	A^5	1455,08	30,16	1448,86	20,59	1478,87	12,03	1463,68	21,94
r206	1357,52	450	A^5	1376,34	31,35	1374,49	23,29	1392,94	12,50	1382,01	23,59
r207	1256,22	450	A^5	1268,66	28,18	1276,85	20,11	1283,47	11,29	1287,18	22,02
r208	1198,39	450	A^5	1208,89	29,02	1211,94	21,01	1219,70	11,33	1207,32	22,38
r209	1328,28	450	A^5	1345,50	30,30	1353,57	21,51	1353,99	12,47	1345,85	23,29
r210	1306,68	450	A^5	1324,07	30,07	1326,36	22,57	1335,34	11,95	1335,80	25,29
r211	1231,38	450	A^5	1244,73	26,30	1257,37	18,62	1258,24	11,11	1259,73	22,05
rc101	2514,62	732	$A^1B^{10}C^7$	2560,33	13,71	2557,21	9,99	2566,60	5,72	2569,55	11,28
rc102	2315,41	726	$A^3B^3C^{10}$	2359,92	14,35	2370,89	10,69	2380,87	5,84	2367,44	11,50
rc103	2105,84	732	$A^1B^4C^7D^2$	2136,78	14,14	2138,79	10,76	2177,39	6,01	2146,20	11,20
rc104	1983,75	732	$A^1B^1C^7D^3$	2002,33	15,56	2018,56	11,13	2024,90	6,48	2011,81	12,21
rc105	2259,97	750	$B^7C^6D^2$	2287,95	13,87	2295,67	10,27	2307,31	5,73	2303,96	11,15
rc106	2202,33	750	$B^4C^9D^1$	2232,05	14,50	2239,22	10,59	2249,98	6,17	2253,34	12,19
rc107	2037,25	750	$C^{11}D^1$	2050,20	15,43	2075,83	11,92	2078,47	6,44	2067,78	12,10
rc108	1962,87	750	$B^2C^7D^3$	1995,41	14,99	1997,78	10,53	2008,07	6,24	1995,62	11,54
rc201	1899,99	620	$A^3B^6C^1$	1931,42	15,69	1933,10	11,02	1944,89	6,29	1940,81	11,67
rc202	1802,53	640	$A^2B^2C^4$	1825,07	16,19	1830,16	12,23	1836,89	6,68	1824,76	13,87
rc203	1642,43	650	$A^1B^1C^5$	1660,93	19,00	1667,71	12,79	1666,36	7,22	1682,69	15,30
rc204	1521,70	660	C^6	1543,04	22,13	1544,66	15,05	1547,57	8,61	1548,63	16,75
rc205	1751,87	660	C^6	1774,22	19,03	1779,34	14,12	1780,26	7,85	1789,09	15,42
rc206	1751,75	640	$A^2B^2C^4$	1767,75	17,79	1775,13	13,61	1782,07	7,26	1773,67	14,48
rc207	1604,60	660	C^6	1640,23	19,07	1629,17	13,63	1637,95	7,78	1635,54	16,13
rc208	1495,34	660	C^6	1520,76	22,03	1518,15	14,29	1517,05	8,75	1521,97	15,50
avg.	1854,07			1875,70	21,47	1879,58	14,32	1887,83	8,71	1883,46	15,77
dev.				1,17		1,38		1,82		1,59	

Table 3.7: Results for the E-FSMVRPTW instances, vehicle type B

	BKS			$ALNS_{2000}$		$ALNS^{800}_{1500}$		$ALNS_{800}$		$\overline{ALNS}^{800}_{1500}$	
name	obj	F	mix	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]	$\overline{obj}$	t[m]
c101	1809,93	690	A^7B^6	1816,06	14,08	1815,44	9,31	1818,94	5,77	1819,13	10,95
c102	1759,73	690	A^7B^6	1766,14	14,36	1767,44	9,71	1776,79	5,96	1773,52	10,76
c103	1750,94	690	A^7B^6	1759,20	15,09	1759,23	10,60	1762,65	6,13	1762,77	12,15
c104	1716,54	730	A^3B^8	1735,86	15,60	1732,42	11,39	1743,52	6,36	1741,73	12,13
c105	1783,25	690	A^7B^6	1785,43	14,51	1785,91	10,35	1787,33	5,79	1786,36	9,95
c106	1773,56	690	A^7B^6	1777,67	14,64	1776,64	10,75	1785,89	6,00	1780,43	11,15
c107	1764,02	710	A^5B^7	1768,33	14,57	1769,50	9,94	1773,49	5,90	1766,78	11,74
c108	1760,63	710	A^5B^7	1769,76	14,80	1766,92	11,19	1775,81	6,12	1770,46	10,94
c109	1739,98	710	A^5B^7	1749,07	15,30	1747,37	10,81	1749,01	6,18	1746,83	12,18
c201	1210,41	520	A^1B^3	1213,63	31,30	1215,74	18,87	1219,12	12,29	1216,46	22,58
c202	1209,73	520	A^1B^3	1220,97	34,17	1217,56	21,57	1223,69	13,78	1228,94	22,86
c203	1210,80	520	A^1B^3	1227,69	33,07	1225,04	19,24	1222,74	14,00	1228,49	23,68
c204	1179,25	520	A^1B^3	1199,37	32,15	1191,08	22,58	1211,63	13,76	1210,35	22,26
c205	1188,92	540	$A^2B^1C^1$	1195,24	31,64	1199,01	16,58	1190,06	12,78	1193,33	19,69
c206	1183,42	540	$A^2B^1C^1$	1192,30	31,01	1195,01	16,58	1194,39	12,65	1193,84	18,99
c207	1183,42	540	$A^2B^1C^1$	1190,37	31,44	1205,69	15,63	1198,19	12,89	1207,32	18,13
c208	1181,47	500	A^5	1192,96	30,82	1190,05	16,93	1189,34	12,48	1185,76	20,72
r101	1961,02	307	$A^3B^2C^9D^6$	1977,89	14,36	1973,60	10,70	1991,48	5,98	1986,34	11,59
r102	1762,84	294	$B^4C^8D^6$	1791,03	14,67	1792,64	10,89	1812,50	5,95	1796,75	11,79
r103	1598,44	328	$B^1C^5D^8E^1$	1618,81	15,52	1620,35	11,55	1634,27	6,34	1621,50	12,21
r104	1424,30	339	$C^1D^9E^2$	1448,31	16,20	1451,81	12,02	1456,97	6,34	1449,33	12,52
r105	1704,33	306	$B^1C^7D^8$	1728,12	14,68	1727,29	11,23	1741,48	6,08	1724,19	11,70
r106	1611,62	322	$A^1C^3D^9E^1$	1635,42	14,67	1641,98	11,02	1652,17	5,97	1641,81	12,08
r107	1485,45	342	$C^3D^6E^3$	1514,01	15,99	1516,72	11,47	1525,06	6,27	1517,24	12,87
r108	1392,23	339	$C^1D^7E^3$	1417,39	16,42	1424,49	11,94	1428,32	6,62	1425,43	13,21
r109	1560,34	320	$C^5D^8E^1$	1580,14	15,65	1588,30	11,85	1596,21	6,22	1592,68	12,22
r110	1435,99	339	$C^1D^9E^2$	1471,66	15,64	1467,84	11,37	1481,78	6,61	1471,06	12,73
r111	1441,38	347	$B^1C^1D^7E^3$	1479,75	16,16	1478,03	11,77	1500,45	6,38	1482,06	12,77
r112	1389,87	353	$C^2D^5E^4$	1403,82	16,10	1415,74	11,46	1429,16	6,69	1416,20	13,11
r201	1365,14	225	A^5	1378,77	29,69	1386,06	20,91	1391,40	11,67	1392,17	22,37
r202	1236,97	225	A^5	1249,65	29,13	1261,20	18,42	1267,71	11,58	1260,57	22,93
r203	1104,85	225	A^5	1124,07	30,23	1129,01	21,53	1131,00	11,99	1131,41	23,41
r204	977,72	225	A^5	983,97	26,81	989,01	18,66	989,05	10,38	985,29	22,14
r205	1207,69	225	A^5	1232,63	29,15	1236,51	20,61	1234,02	11,72	1235,02	22,65
r206	1133,05	225	A^5	1155,47	30,95	1157,46	22,48	1164,89	12,36	1162,55	23,45
r207	1031,22	225	A^5	1057,22	26,31	1056,94	19,20	1055,72	10,69	1055,75	20,66
r208	970,96	225	A^5	984,87	28,21	983,20	20,96	990,36	10,73	987,94	23,42
r209	1096,79	225	A^5	1117,68	29,62	1119,80	20,64	1120,97	11,80	1119,42	21,93
r210	1072,77	225	A^5	1100,27	29,88	1096,77	21,18	1105,13	11,83	1090,81	22,76
r211	1005,86	225	A^5	1026,07	25,93	1027,94	17,84	1036,95	10,31	1025,14	20,20
rc101	2141,19	405	$B^7C^7D^2$	2153,24	13,80	2162,37	10,68	2174,74	5,87	2161,04	11,09
rc102	1951,11	417	$A^2B^3C^9D^2$	1972,85	14,68	1984,48	10,81	1993,25	5,97	1985,59	11,61
rc103	1730,15	390	$B^4C^5D^4$	1764,22	14,54	1760,98	10,84	1778,77	5,92	1762,34	11,30
rc104	1587,86	390	$B^3C^4D^5$	1614,09	15,80	1607,72	11,76	1643,14	6,38	1614,01	12,52
rc105	1884,79	381	$A^1B^5C^7D^2$	1900,42	14,14	1905,81	10,01	1923,55	5,71	1901,24	11,08
rc106	1812,75	381	$A^1B^3C^8D^2$	1844,99	15,18	1849,55	10,92	1857,81	6,26	1843,19	11,84
rc107	1639,84	390	$B^1C^8D^3$	1675,58	15,31	1693,61	11,49	1690,65	6,52	1681,58	12,08
rc108	1578,51	390	$B^2C^6D^4$	1601,47	14,98	1613,71	10,67	1612,45	6,03	1614,22	11,28
rc201	1589,99	310	$A^3B^6C^1$	1617,52	15,64	1609,45	9,93	1623,12	6,42	1620,26	12,39
rc202	1481,05	325	$A^3B^1C^3D^1$	1497,99	16,72	1493,53	11,05	1503,61	6,84	1501,96	14,21
rc203	1309,09	330	$A^2C^4D^1$	1333,25	19,35	1334,04	14,39	1346,14	7,64	1332,54	15,45
rc204	1182,32	350	C^2D^3	1193,93	22,69	1199,08	17,51	1199,99	9,17	1205,66	17,14
rc205	1424,42	325	$A^1B^1C^5$	1440,00	20,95	1455,78	14,13	1447,39	8,24	1433,94	15,49
rc206	1431,21	320	$A^1B^4C^3$	1439,17	18,11	1448,51	13,34	1460,01	7,67	1443,80	14,36
rc207	1273,50	330	C^6	1299,14	21,30	1298,88	13,19	1308,83	7,88	1311,79	15,51
rc208	1161,23	350	C^2D^3	1171,52	22,91	1179,75	16,01	1181,71	9,06	1175,06	17,55
avg.	1456,35			1474,22	20,83	1476,79	14,26	1484,01	8,41	1478,06	15,72
dev.				1,23		1,40		1,90		1,49	

Table 3.8: Results for the E-FSMVRPTW instances, vehicle type C

	Schneider			ALNS			Δ
name	m	obj	t[m]	m	obj	t[m]	$\Delta(obj)$
c101	**12**	**1053,83**		**12**	**1053,83**	9,16	0,00
c102	**11**	**1056,47**		**11**	1057,16	9,76	0,07
c103	**10**	**1041,55**		**10**	1044,15	9,15	0,25
c104	**10**	**979,51**		**10**	984,61	11,09	0,52
c105	**11**	**1075,37**		**11**	**1075,37**	9,31	0,00
c106	**11**	1057,87		**11**	**1057,65**	9,44	-0,02
c107	**11**	**1031,56**		**11**	**1031,56**	9,66	0,00
c108	**10**	**1100,32**		**10**	1109,45	9,21	0,83
c109	**10**	**1036,64**		**10**	1051,50	10,17	1,43
c201	**4**	**645,16**		**4**	**645,16**	18,11	0,00
c202	**4**	**645,16**		**4**	646,52	21,13	0,21
c203	**4**	**644,98**		**4**	**644,98**	22,91	0,00
c204	**4**	**636,43**		**4**	638,32	19,70	0,30
c205	**4**	**641,13**		**4**	**641,13**	20,96	0,00
c206	**4**	**638,17**		**4**	**638,17**	23,26	0,00
c207	**4**	**638,17**		**4**	**638,17**	22,68	0,00
c208	**4**	**638,17**		**4**	**638,17**	22,30	0,00
r101	**18**	1670,80		**18**	**1663,04**	8,80	-0,46
r102	**16**	1495,31		**16**	**1488,97**	9,85	-0,42
r103	**13**	1299,17		**13**	**1285,96**	9,91	-1,02
r104	**11**	**1088,43**		**11**	1097,82	8,73	0,86
r105	*14*	*1461,25*		*15*	*1433,92*	*9,31*	-
r106	**13**	**1344,66**		**13**	1363,25	8,91	1,38
r107	**12**	**1154,52**		**12**	1165,33	9,16	0,94
r108	**11**	**1050,04**		**11**	1067,43	8,78	1,66
r109	*12*	*1294,05*		*13*	*1245,89*	*9,48*	-
r110	**11**	**1126,74**		**11**	1155,64	9,15	2,57
r111	**12**	**1106,19**		**12**	1120,48	9,34	1,29
r112	**11**	**1026,52**		**11**	1043,79	8,70	1,68
r201	**3**	**1264,82**		**3**	1269,52	27,05	0,37
r202	**3**	**1052,32**		**3**	1053,93	27,60	0,15
r203	**3**	**895,91**		**3**	897,12	27,81	0,14
r204	**2**	790,57		**2**	**788,67**	17,86	-0,24
r205	**3**	**988,67**		**3**	1002,00	24,85	1,35
r206	**3**	925,20		**3**	**922,70**	24,98	-0,27
r207	**2**	**848,53**		**2**	859,78	20,83	1,33
r208	**2**	**736,60**		**2**	740,24	18,25	0,49
r209	**3**	**872,36**		**3**	890,69	25,56	2,10
r210	**3**	**847,06**		**3**	863,53	25,56	1,94
r211	**2**	**847,48**		**2**	873,67	20,62	3,09
rc101	**16**	1731,07		**16**	**1726,91**	8,05	-0,24
rc102	*15*	*1554,61*		*14*	*1659,53*	*8,57*	-
rc103	**13**	**1351,15**		**13**	1369,34	8,89	1,35
rc104	**11**	1238,56		**11**	**1229,82**	8,79	-0,71
rc105	**14**	**1475,31**		**14**	1478,67	7,88	0,23
rc106	**13**	1437,96		**13**	**1436,61**	7,63	-0,09
rc107	**12**	**1279,08**		**12**	1283,52	8,02	0,35
rc108	**11**	1209,61		**11**	**1204,87**	7,68	-0,39
rc201	**4**	**1444,94**		**4**	1464,33	23,02	1,34
rc202	**3**	**1418,79**		**3**	1437,02	25,12	1,28
rc203	**3**	**1073,98**		**3**	1084,71	26,12	1,00
rc204	**3**	**885,35**		**3**	902,69	26,45	1,96
rc205	**3**	1330,53		**3**	**1282,58**	23,42	-3,60
rc206	**3**	**1190,75**		**3**	1218,79	24,36	2,36
rc207	**3**	**1004,38**		**3**	1016,12	23,75	1,17
rc208	**3**	**837,82**		**3**	847,89	24,64	1,20
avg			15,34			15,92	0,52

Table 3.9: Results for the E-VRPTW instances

		BDHMG08	RT10	VCGP13			ALNS		
type	name	Best 3	Single	Avg 10	Best 10	t[m]	Avg 10	Best 10	t[m]
A	C1	0,12	0,10	0,09	0,09	2,90	0,66	0,28	10,37
	C2	0,95	0,11	0,30	0,29	5,16	1,17	0,35	31,91
	R1	0,94	0,51	0,39	0,24	5,62	2,31	1,39	8,69
	R2	2,79	0,71	0,15	0,08	9,41	1,25	0,63	33,56
	RC1	0,75	0,74	0,20	0,00	5,58	1,67	0,87	8,52
	RC2	0,61	0,35	0,26	0,19	5,77	1,15	0,55	12,20
	avg.	0,92	0,38	0,23	0,15	5,88	1,34	0,66	17,64
B	C1	0,24	0,02	0,06	0,06	3,00	0,28	0,09	10,00
	C2	0,81	0,26	0,39	0,34	3,86	1,16	0,40	25,17
	R1	0,88	0,72	0,49	0,37	4,88	2,37	1,16	9,23
	R2	2,30	1,57	0,25	0,12	6,97	2,82	1,01	23,81
	RC1	0,31	0,64	0,24	0,12	4,23	1,60	0,40	8,77
	RC2	1,52	0,80	0,40	0,29	5,26	2,53	0,83	15,20
	avg.	0,95	0,65	0,30	0,22	4,80	1,76	0,65	15,28
C	C1	0,19	0,01	0,03	0,03	2,45	0,25	0,06	11,07
	C2	0,33	0,33	0,09	0,09	3,61	0,93	0,18	27,82
	R1	1,07	0,68	0,56	0,44	4,72	2,19	1,12	9,44
	R2	1,67	1,88	0,11	0,00	6,63	2,51	0,64	23,57
	RC1	0,33	0,60	0,22	0,02	4,19	1,61	0,43	9,06
	RC2	1,67	1,88	0,11	0,00	6,63	2,51	0,64	23,57
	avg.	1,02	0,76	0,40	0,27	4,60	1,89	0,75	16,01

Table 3.10: Average results for the FSMVRPTW instances