ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis BSc^2 Econometrics/Economics

Solving Electric Vehicle Routing Problems using an Adaptive Large Neighbourhood Search

Author
M. Wagenvoort
Student number
423267

Supervisor
Dr. D. Galindo Pecin
Second assessor
Dr. R. Spliet

Abstract

The Adaptive Large Neighbourhood Search of Pisinger & Ropke (2007) is able to solve various types of vehicle routing problems. Due to recent developments regarding global warming awareness and the usage of electric vehicles, the extension of this heuristic to electric vehicle routing problems is of interest. Therefore, this Adaptive Large Neighbourhood Search is implemented to test the replicability of the model, where after alterations to the electrical vehicle routing problem are made. In the Adaptive Large Neighbourhood Search, a solution is destroyed and subsequently repaired using destroy and repair heuristics that are chosen at random based on their historical performance. Pisinger & Ropke (2007) have proven the good performance of the model relative to other heuristics using the same data sets. Replicating the model did however show the dependency of the model on the parameter setting resulting in difficulties in reducing the number of required vehicles. The extended model for the electric vehicle routing model has shown great potential as well as the addition of extra destroy heuristics. The recommendation for future research is therefore to improve the model as to improve the vehicle minimisation stage and the additional destroy heuristics suggested in this paper.

July 7, 2019

Contents

1 Introduction

The number of online purchases have increase over the years, resulting in an increase in the number of packages to be delivered. These often have to be delivered in a short time span due to promises of websites to customers regarding next day delivery. Some websites also offer the possibility of choosing a time window in which your package will be delivered, adding limitations to the delivery schedule. At the same time, the delivery companies have their own limitations in terms of the number of available vans and employees as well as working conditions in terms of, e.g. the length of working shifts. Therefore, a fast method to construct feasible delivery routes while also minimising the costs for the delivery company is of interest.

In a vehicle routing problem (VRP), goods have to be retrieved at pick-up locations and brought to their corresponding delivery location. The vehicle have a certain start and end location and the routes can be subject to various constraints, e.g. maximum route duration, vehicle capacities and time windows at the pick-up and delivery locations. Here, the aim is to construct routes such that, e.g. the total distance travelled or the required number of vehicles is minimised.

Due to the large spectrum of constraints that can be imposed to the VRP, various types of the VRP exist. Heuristics have been proposed for different types of vehicle routing problems. However, only a few heuristics have been proposed that can be used on a variety of vehicle routing problems. Such a heuristic has been created by Pisinger & Ropke (2007). Pisinger & Ropke (2007) have constructed an Adaptive Large Neighbourhood Search which can be applied to five different types of the VRP. In the Adaptive Large Neighbourhood Search, Pisinger & Ropke aim to find the best solution by removing and consequently adding part of the routes from the solutions. Here, various destroy and repair heuristics are used to determine the part of the routes to be removed and the order of adding the customer requests back into the solution. The five types of the VRP considered by Pisinger & Ropke are the capacitated vehicle routing problem (CVRP), vehicle routing problem with time windows (VRPTW), the open vehicle routing problem (OVRP), the multi depot vehicle routing pro-blem (MDVRP), and the site-dependent vehicle routing problem (SDVRP).

The CVRP is the most basic form of a VRP, where a single depot exists and the only limitations on the problem are the capacities of the vehicles and possibly a route duration. The VRPTW extends the CVRP with time windows, i.e., the service at a customer should start within a certain time interval. The CVRP and the VRPTW are most well known and hence most research exists on those problems. This increases the difficulty for a general heuristic to compete with type-specific heuristics for the CVRP and the VRPTW.

The OVRP is similar to the CVRP, where the vehicles have a certain capacity and a route duration might be imposed. However, here it is not required for the vehicle to terminate at the depot, but the route stops when the last customer is served. In the MDVRP, multiple depots exists and customers can possibly be served by more than one of these depots. These depots can thus not be handled as separate problems at it is unknown which customer should be served by which depot in the optimal solution.

In all the previous VRP types, a homogeneous fleet is assumed. The SDVRP handles cases where the vehicle fleet is in fact heterogeneous, i.e. the capacities are different or some vehicles

contain cooling compartments and others do not. As some customers could be served by multiple vehicles, as is the case with the depots in the MDVRP, the SDVRP cannot be considered as separate problems.

The OVRP, MDVRP and SDVRP have received less attention in research compared to the CVRP and the VRPTW. However, these problems are not uncommon for transportation companies and are therefore of great interest.

Due to global warming and health concerns related to emissions, vehicles based on alternative fuels have increased in popularity. Multiple cities have already introduced so-called environmental zones that exclude certain high-emission cars and trucks from parts of the city (Visbeen, March 7 2018). Amsterdam is taking this a step further by stating that from 2030 onward, no car using petrol can enter the city (Bouma, May 2 2019). An alternative to petrol vehicles are electrical vehicles, which are rising in popularity amongst users and car suppliers (Kottasová, May 9 2019). Due to battery limitations and limited charging station locations, scheduling visits to the charging stations are desired to avoid empty batteries or large detours to visit charging stations. This imposes more restrictions for delivery companies using electrical vehicles.

Recent research has focused more and more on the electrical vehicle routing problem (EVRP) such that these additional restrictions regarding the limited battery capacity of the vehicles and the limited number of charging stations available at this moment can be incorporated.

Combining both the increase in the number of deliveries and the use of electrical vehicles calls for methods to create cost-limiting routes for (electrical) vehicles within limited time. As the Adaptive Large Neighbourhood Search of Pisinger & Ropke (2007) is able to solve five different types of the VRP and it is able to compete with type specific heuristics, including the possibility of scheduling using electrical vehicles can increase the applicability of their heuristic for the future.

Therefore, the aim of this paper is to extend the Adaptive Large Neighbourhood Search of Pisinger & Ropke (2007) such that it can also solve the EVRP. As it is important that the results of a paper can be replicated, some results of Pisinger & Ropke (2007) are verified before the heuristic is extended to the electrical vehicle routing problem. Here, the focus is on the replication of the vehicle routing problem with time windows, the multi depot vehicle routing problem, and the site-dependent vehicle routing problem.

The implementation of the Adaptive Large Neighbourhood Search of Pisinger & Ropke appears to be prone to slight changes in the parameter setting and possible implementation and interpretation differences of some of the aspects of the heuristic. Hence, although good solutions are found, the heuristic of Pisinger & Ropke results in better solutions in many of the problems evaluated.

The extension of the Adaptive Large Neighbourhood Search results in good solutions that have the potential of resulting in new best solutions compared to other heuristic methods used to solve the EVRP. Also, the addition of destroy heuristics that aim to improve the routes of an EVRP show potential. However, in the construction of the extended Adaptive Large Neighbourhood Search, some assumptions are made regarding the electric vehicle routing problem. These could affect the applicability of the heuristic. Evaluating the effect of these assumptions is therefore of interest.

The structure of the paper is as follows. First, some literature on (electrical) vehicle routing problems is discussed. Second, the Adaptive Large Neighbourhood Search of Pisinger & Ropke (2007) is explained followed by the alterations made to extend this heuristic to electrical vehicles. After the data used to verify the results of Pisinger & Ropke and to test the performance of the extended Adaptive Large Neighbourhood Search is introduced, the results are reported. To conclude, the replicability of the results of Pisinger & Ropke and the performance of the extended heuristic is discussed and limitations of the paper and recommendations for future research are provided.

2 Literature Review

Due to the applicability of the VRP, it has been analysed extensively. Various papers discuss exact algorithms to solve the VRP. Most research focuses on the CVRP as this is the most basic form of the VRP. Lysgaard et al. (2004) solve the CVRP by using a branch-and-cut algorithm with the aim of reducing the feasible region in order to find or prove that a given solution is optimal. This is achieved by adding valid inequalities to the problem. Fukasawa et al. (2006) extend this idea by combining a branch-and-cut algorithm with column generation resulting in a branch-and-cut-and-price algorithm. They argue that the addition of column generation to the branch-and-cut algorithm has more potential than trying to construct new valid inequalities that can be added to already existing branch-and-cut algorithms. To improve the running time of the branch-and-cut-and-price algorithm, Pecin et al. (2017) use faster pricing heuristics for the column generation stage of the algorithm. Additionally, strong branching is applied in which branching a variable is only executed after the effect on the objective value is evaluated.

Although most research focuses on solving the CVRP, additional restrictions often exist for delivery companies. As this may require different cuts to reduce the feasible region and running time of the algorithm, some research also focuses on different types of the VRP. As time windows are very common restrictions for delivery companies, Baldacci et al. (2011) have targeted their algorithm not only on the CVRP, but also on the VRPTW. In their algorithm, they combine the LP-relaxation of the set covering problem with the removal and addition of some inequalities to reduce the size of the feasible region. Chen & Xu (2006) extend the basic VRPTW to a dynamic VRPTW. This implies that constant evaluation of the problem is required upon the addition of a new request to the problem. The authors use a column generation approach to solve this problem every time a new order arrives.

To solve the MDVRP, Contardo & Martinelli (2014) use both the set partitioning and the vehicle flow formulation to the MDVRP. To solve the set covering problem by column generation, valid inequalities are added. Furthermore, the lower bound from the vehicle flow formulation is used to eliminate edges in the set partitioning problem.

Letchford et al. (2007) found that the branch-and-cut algorithm can be used for the capacitated OVRP. However, different inequalities should be used compared to the regular CVRP. Choi & Tcha (2007) use column generation to solve the LP-relaxation of the set covering formulation of the VRP with a heterogeneous fleet. To convert the solution to an integer solution, branch-and-bound is used.

Dayarian et al. (2015) consider a very specific type of a VRP, where multiple depots and

a heterogeneous fleet exists and time windows are imposed, making this a combination of the MDVRP and the VRPTW with a heterogeneous fleet. Also for this problem, a branch-and-price algorithm is used, i.e. a branch-and-bound algorithm using column generation for the lower bounds.

From the aforementioned papers, it can be concluded that most recent research combines the addition of valid inequalities and column generation when aiming to find the optimal solution to a VRP as fast as possible. However, as the VRP is NP-hard, since the travelling salesman problem is a special case of a VRP, the need for heuristic methods arises when large instances are considered (Mester & Bräysy, 2005). As the VRP is widely studied, there exist many different types of heuristics that can be classified in larger groups.

A set of heuristics make use of neighbourhood structures to iteratively improve a solution (Vidal et al., 2013). In such heuristics, certain parts of a solution are moved or switched with another part. To avoid cycling by constantly selecting the same part of the solution to alter, a tabu search aspect can be implemented which prohibits the execution of the same move within a certain amount of iterations (Gendreau et al., 1994). Tabu searches have been applied to many VRP types, e.g. the CVRP (Gendreau et al., 1994), the MDVRP (Cordeau et al., 1997), the SDVRP (Cordeau & Laporte, 2001), the VRPTW (Cordeau et al., 2001), and the VRP with backhauls (Ropke & Pisinger, 2006b).

Another example of a neighbourhood search heuristic is simulated annealing (Vidal et al., 2013). In simulated annealing, a new solution is accepted with a certain probability based on its objective value, where this probability decreases over time (Vidal et al., 2013). Thus, diversification is possible at the start of the search, where intensification is the goal in a later stage. Czech & Czarnas (2002) use this method to solve the VRPTW.

Variable neighbourhood searches alter solutions by mutating or switching part of a solution to reach the local optimum corresponding to the neighbourhood structure (Vidal et al., 2013). Fleszar et al. (2009) show that such a heuristic can be used to solve the OVRP and variable neighbourhood searches have also proven to perform well on very large-scale problems (Kytöjoki et al., 2007).

Besides heuristics based on neighbourhood structures of solutions, there are also heuristics using a population approach, such as evolutionary algorithms (Vidal et al., 2013). evolutionary algorithms make use of the laws of nature where a population of solutions is maintained and updated by removing weak solutions (elimination), combining two solutions to form a new solution (crossover), and altering a solution (mutation) (Ho et al., 2008; Mester & Bräysy, 2005). As a set of solutions is stored and used to find a new solution, the algorithm contains diversifying aspects. This aspect can be extended by also allowing for infeasible solutions in the population (Vidal et al., 2014).

Ant colony optimisation is another type of heuristic that uses a population structure to solve the VRP (Vidal et al., 2013). Here, the quality of solutions is used to learn for the construction of new routes (Bell & McMullen, 2004; Yu et al., 2009).

There are thus various types of heuristics available to solve the VRP. However, various restrictions can be imposed on the VRP resulting in a set of different problems and most heuristics focus on one type of VRP. To avoid the need for various heuristics for the different types of the

VRP, Pisinger & Ropke (2007) have constructed a general heuristic that can handle multiple types of the VRP - the CVRP, the VRPTW, the OVRP, the MDVRP, and the SDVRP. Their heuristic is an Adaptive Large Neighbourhood Search that contains various heuristics to destroy and repair the neighbourhood structure as well as simulated annealing. As it can handle multiple VRP types and is able to compete with type-specific heuristics, the heuristic of Pisinger & Ropke is of relevance to the research done so far.

However, as global warming awareness has increases and the transportation sector is responsible for a large part of the emissions (Salimifard et al., 2012), recent research has aimed at reducing the emissions caused by delivery companies besides simply reducing the costs for the delivery company. The aim of a VRP is minimising the number of vehicles and the travel distances. As this positively affects emissions from car production and fuel usage, algorithms aimed at solving the VRP have unconsciously considered emissions since their occurrence (Salimifard et al., 2012). Recent research has extended this to consciously including the emissions of vehicles as part of the VRP.

Sbihi & Eglese (2007) consider the Vehicle Routing and Scheduling Problem. In this problem, travel times of arcs are dependent on the time that arc is traversed as road congestion during peak hours affects the travel times. Here, the aim is to minimise the travel time and not solely the travel distance. By reducing the travel time, the emissions of cars are also reduced. This concept is extended by Bektaş & Laporte (2011) by constructing the Pollution-Routing Problem. This problem aims at capturing the externalities of driving a car related to pollution by considering not only travel distance and time in its objective value, but also emissions by the vehicle, fuel usage, and their costs.

These problems aim at limiting emissions of vehicles using petrol. Alternatively, emissions can be reduced by replacing vehicles using petrol with electrical vehicles. As a result of developments regarding the quality of electrical vehicles, recent research has focused more and more on the electrical vehicle routing problem (EVRP). The need to consider the EVRP as a separate problem comes from the limited battery capacity of the vehicles and the limited charging stations available at this moment. To avoid large detours during the route to charge the battery, it is therefore beneficial to include visits to the charging station in the route of an electrical vehicle. Desaulniers et al. (2016) propose exact methods to use various types of the EVRP with time windows (EVRPTW). Here, they also differentiate between always fully charging the vehicle and allowing for partial charges. Montoya et al. (2017) extend the charging habits to include non-linearity in the charging speed at stations.

To solve the EVRP faster for large instances, heuristic methods have been proposed that combine variable neighbourhood search with tabu search (Schneider et al., 2014). Previously mentioned papers have all considered a fixed fleet of electrical vehicles, whereas Hiermann et al. (2016) extended the EVRPTW to include the decision on the size of the electrical fleet.

All of the aforementioned papers consider the EVRP as a specific type of VRP and hence no general method has been found that aims to construct a single heuristic that is able to solve the EVRP and other types of the VRP. Therefore, extending the Adaptive Large Neighbourhood Search of Pisinger & Ropke to electrical vehicles is of benefit. The extension of the Adaptive Large Neighbourhood Search will focus on the EVRPTW as the inclusion of time windows to

the basic EVRP increases the applicability greatly.

3 The Heuristic of Pisinger & Ropke

Pisinger & Ropke (2007) make use of an Adaptive Large Neighbourhood Search (ALNS) to solve the various types of the VRP. To apply this heuristic to all VRP types, they are transformed to a general problem, the rick pick-up and delivery problem with time windows (RPDPTW). The ALNS is then applied to this problem. Therefore, the RPDPTW will be explained before the ALNS is introduced.

3.1 The Rich Pick-Up and Delivery Problem with Time Windows

The RPDPTW consists of requests with a pick-up and delivery location such that time windows and vehicle capacities are respected. We can thus express request i in terms of a pick-up at location $i \in P$ and delivery at location $i + n \in D$, where $P = \{1, ..., n\}$ and $D = \{n + 1, ..., 2n\}$ are the sets with pick-up and delivery locations, respectively, jointly denoted by N. Precedence numbers, Π_i for $i \in N$, can be assigned to ensure that some locations are visited before others. These can be useful in, e.g. avoiding cross-contamination when transporting live stock (Sigurd et al., 2004) and to allow for multiple depots as in the MDVRP. There is a fleet of m vehicles denoted by the set K, were vehicle $k \in K$ can serve $N_k = P_k \cup D_k$, with $P_k \subseteq P$ and $D_k \subseteq D$ the pick-up and delivery locations of the requests that can be served by vehicle k, respectively. As this allows for a heterogeneous fleet, various types of vehicles for different purposes can be used as is the case in the SDVRP. Each vehicle has a start,τ_k, and end location, τ'_k, so the set of nodes V can be denoted by $N \cup \{\tau_1, ..., \tau_m\} \cup \{\tau'_1, ..., \tau'_m\}$. By assigning high costs to requests that are not served, i.e. requests that are placed on the request bank, the aim of the problem is to serve as many requests as possible while minimising the total travel costs.

Let d_{ij} denote the distance between nodes $i, j \in V$ and t_{ij} the corresponding travel time. The time to (un)load at node $i \in V$ is denoted by s_i and the (un)loading should start between times a_i and b_i. The demand at node $i \in V$ is represented by l_i, where l_i is non-negative at pick-up nodes and non-positive at delivery nodes, such that $l_i = -l_{i-n}$ for $i \in D$. The capacity of vehicle $k \in K$ is given by C_k.

The decision variables used are x_{ijk}, z_i, S_{ik} and L_{ik}. Here, x_{ijk} is a binary variable that is equal to 1 in case arc $(i, j) \in A$ is in the route of vehicle $k \in K$ and z_i is a binary variable that is equal to 1 if request $i \in P$ is not served, i.e. when request $i \in P$ is placed on the request bank. The time that service starts at node $i \in V_k$ for vehicle $k \in K$ is denoted by S_{ik} and the load of the vehicle after service of the node is given by L_{ik}.

Let c_{ijk} be the cost of travelling over arc $(i, j) \in A_k$ by vehicle $k \in K$ and Γ_i the cost of placing request $i \in P$ on the request bank, the model can be defined as

$$\min \sum_{k \in K} \sum_{(i,j) \in A_k} c_{ijk} x_{ijk} + \sum_{i \in P} \Gamma_i z_i \tag{1}$$

$$\text{Subject to}: \sum_{k \in K} \sum_{j \in N_k} x_{ijk} + z_i = 1 \qquad i \in P \tag{2}$$

$$\sum_{j \in V_k} x_{ijk} - \sum_{j \in V_k} x_{j,n+i,k} = 0 \qquad k \in K, i \in P_k \tag{3}$$

$$\sum_{j \in P_k \cup \{\tau'_k\}} x_{\tau_k jk} = 1 \qquad k \in K \tag{4}$$

$$\sum_{i \in D_k \cup \{\tau_k\}} x_{i\tau_k k} = 1 \qquad k \in K \tag{5}$$

$$\sum_{i \in V_k} x_{ijk} - \sum_{i \in V_k} x_{jik} = 0 \qquad k \in K, j \in N_k \tag{6}$$

$$S_{ik} + s_i + t_{ij} - S_{jk} \leq M_1(1 - x_{ijk}) \qquad k \in K, (i,j) \in A_k \tag{7}$$

$$S_{ik} \leq S_{n+i,k} \qquad k \in K, i \in P_k \tag{8}$$

$$L_{ik} + l_j - L_{jk} \leq M_2(1 - x_{ijk}) \qquad k \in K, (i,j) \in A_k \tag{9}$$

$$L_{\tau_k k} = L_{\tau'_k k} = 0 \qquad k \in K \tag{10}$$

$$\Pi_i - \Pi_j \leq M_3(1 - x_{ijk}) \qquad k \in K, i \in V_k, j \in V_k \tag{11}$$

$$x_{ijk} \in \{0,1\} \qquad k \in K, (i,j) \in A_k \tag{12}$$

$$z_i \in \{0,1\} \qquad i \in P \tag{13}$$

$$a_i \leq S_{ik} \leq b_i \qquad k \in K, i \in V_k \tag{14}$$

$$0 \leq L_{ik} \leq C_k \qquad k \in K, i \in V_k. \tag{15}$$

Here, the objective function is to minimise the travel costs of the vehicles and the costs of not serving a request. Constraint (2) ensures that each request is either served or placed on the request bank. Constraints (3) and (8) ensure that pick-up and delivery locations of a request are visited by the same vehicle and that the pick-up location is visited first, respectively. Constraints (4) - (6) are used to ensure that vehicles start and end at the correct locations and that vehicles depart a node if they enter it. Service times and vehicle loads are updated by constraints (7) and (9), respectively. Here M_1 and M_2 are set to ensure that if vehicle $k \in K$ does not traverse $(i,j) \in A$, no restrictions are imposed on the decision variables. As it is desired to have a feasible region that is as small as possible, it is desired to have M_1 and M_2 as small as possible. Therefore, M_1 is set to be equal to the maximum time window and M_2 to the maximum vehicle capacity. Constraint (10) ensures that vehicles are empty at the start and end of their route. Constraint (11) is introduced to satisfy precedence ordering of locations, where M_3 has again a value large enough to impose no restriction in case $x_{ijk} = 0$ while also limiting the size of the feasible region. Hence, M_3 is set to the maximum precedence value. Constraints (14) and (15) ensure that time windows and vehicle capacities are adhered to.

3.2 Transformations to the Rich Pick-Up and Delivery Problem with Time Windows

To use the heuristic based on the RPDPTW for the different types of problems, these problem types have to be transformed to the RPDPTW. Hence, the transformations from the VRPTW, the MDVRP and the SDVRP to the RPDPTW are explained below.

To transform the VRPTW to the RPDPTW, request are created such that all pick-up locations are at the depot and delivery locations at the locations of the customers. Here, the service time at the depot equals 0. To avoid restocking by revisiting the depot during the route, the precedence number of all pick-ups are set smaller than the precedence number of all deliveries. The time windows of the pick-up locations are equal to the time windows of the depot.

To introduce multiple depots, as in the MDVRP, an imaginary depot is created with dummy requests to the other depots. The costs of placing these requests on the request bank are set very high such that these are always placed in the route. By using a precedence for the depots such that they will be visited before and after the customers are visited, the MDVRP can be transformed to a RPDPTW. Furthermore, each dummy request can only be served by one vehicle such that each dummy request is chosen exactly once per route.

The SDVRP can be transformed to the RPDPTW in a similar way as the VRPTW, where the time windows at every pick-up and delivery location are set from 0 to the maximum route duration allowed. Furthermore, for each vehicle, the set of requests that can be served by the vehicle are specified.

3.3 The Adaptive Large Neighbourhood Search

After the VRP has been transformed to a RPDPTW, the heuristic can be applied to the problem to find a feasible solution with the aim of minimising the costs, i.e. the distance travelled by all vehicles. Here, a set of requests, called a neighbourhood, is removed from a solution and then added again with the aim of improving the solution. To avoid being stuck in a local minimum, the ALNS contains heuristics that aim at diversifying the search and it exhibits the principle of simulated annealing, i.e. solutions are accepted based on their objective value with a certain probability that decreases over time. The starting temperature for the simulated annealing is chosen such that a solution with a $w\%$ improvement of the objective value compared to the initial solution (where the objective value of the initial solution is excluding possible cost of requests on the request bank) is equally likely to be accepted or rejected. Then, after every iteration, the temperature is reduced with a cooling rate factor c.

The heuristic is adaptive as it adjusts the weights of the heuristics used for determining with which probability a destroy and repair heuristic is chosen. The procedure for picking the heuristics to execute and update the weights of the heuristics will be explained after the heuristics are introduced.

3.3.1 The Destroy Neighbourhood

There are seven heuristics for choosing the destroy neighbourhood on the current solution. Some of these heuristics have the aim of intensifying and other the aim of diversifying the search. As

Pisinger & Ropke noted, removing too large neighbourhoods rarely results in solutions that will be accepted. Hence, the size of destroy neighbourhood q is chosen randomly between $\min\{0.1n, 30\}$ and $\min\{0.4n, 60\}$, where n is the number of instances in the problem. This implies that the average size of the destroy neighbourhood increases linearly in the size of the problem with 45 as the maximum.

In the *random removal* heuristic, q requests are randomly selected and removed from the solution. This heuristic does not use any information on the structure of the current solution and hence it is an example of a heuristics that aims to diversify the search. In the *worst removal* heuristic, the requests with high costs in the current solution structure are removed. Here, the costs are determined as the reduction in the objective value in case the request is removed from the problem completely, i.e. when also the costs of placing the request on the request bank are ignored. Requests are then selected by using a randomisation parameter, p, to select q requests from the sorted list, L, by selecting request $y^p|L|$, with y a uniform random variable. Here the value of the randomisation parameter determines the probability of choosing a request with high costs, i.e. it determines how likely a request from the top of the list is removed from the solution.

Although information on the structure of the solution is used in the *worst removal* heuristic, the decision on the requests to be removed from the solution does not take into account any information on the location and time of the requests. This can increase the difficulty of improving the solution as the chance is relatively high that requests will be inserted at approximately the same locations in the routes. Hence, some heuristics are introduced that focus more on intensifying the search compared to the previously mentioned heuristics that focus more on diversification of the solution.

The first heuristic to use the location of the requests is the *related removal* heuristic, where requests are chosen based on their relatedness to a randomly selected request that is already in the destroy neighbourhood (the initial request placed in the destroy neighbourhood is chosen randomly). The relatedness between requests i and j is computed as denoted in equation (16). Here, D is the number of non-zero terms $d'(k, l)$ with $d'(k, l)$ equal to the distance between node k and l in case neither k or l are located at the terminal, and it equals zero otherwise. Hence, the average is taken of all arcs that do not originate or terminate at a terminal. Distances from or to the start or end terminal are excluded as a route will always have to depart the start terminal and terminate at the end terminal. As with the *worst removal* heuristic, a randomisation parameter is used for selecting the request to be removed from the sorted list.

$$R_{ij} = \frac{1}{D}(d'(i, j) + d'(i, j + n) + d'(i + n, j) + d'(i + n, j + n)) \tag{16}$$

Another method to remove requests based on their location is the *cluster removal* heuristic, where a randomly selected route is partitioned into disjoint sets according to Kruskal's algorithm (Kruskal, 1956). All requests in one of the disjoint sets are then removed from the solution and placed on the request bank. In Kruskal's algorithm, the shortest arc is added to the solution that does not result in a loop such that a minimum spanning tree is found. For the purpose of the request removal, the algorithm will terminal when there are exactly two disjoint sets, where after one of the clusters is randomly chosen as (part of) the destroy neighbourhood. Initially, a route is randomly selected. When the size of the removed cluster from this route is smaller

than the required destroy neighbourhood size q, another route is selected on which Kruskal's algorithm will be applied. A request from the destroy neighbourhood is randomly chosen and based on the relatedness as defined in the *related removal* heuristic, a request from an untouched route is selected. This route is the next route on which Kruskal's algorithm will be applied. This process is repeated until at least q requests are removed or all routes have been partitioned. This will give a destroy neighbourhood that is more likely to come from a relatively small selection of routes. Hence, the chance of different re-insertions that result in a better solution are larger compared to previous heuristics where the destroy neighbourhood is more likely to consist of request from a large selection of routes.

The RPDPTW can contain time windows, imposing a restriction on reinserting the requests. Hence, constructing a destroy neighbourhood based on the time at which requests are served can be beneficial. Combining this with the coordinates of locations increases the probabilities of inserting the requests in different locations in the routes even more. Hence, in the *time-oriented removal* heuristic, a set of B requests is selected based on the relatedness R_{ij} to a randomly selected request. Of this set, q requests are removed based on the time difference at which they are served compared to another request. This time difference is defined as $\Delta t_{ij} = |t_{pi} - t_{pj}| + |t_{di} - t_{dj}|$ with t_{pi} and t_{di} the pick-up and delivery time of request i in the current solution.

Finally, two heuristics are created based on information on previous solution structures. In the *historical node-pair removal* heuristic, the objective value of the best known solution in which an arc from node i to j occurs is stored. Then, the request is removed that is most "costly", i.e. for which the "costs" of the edges incident to the pick-up and delivery location are highest. Here, the cost of an edge is defined as the objective value of the best known solution that contains the edge. The randomisation parameter as described in the *worst removal* heuristic is used to select the request that should be removed. This heuristic thus breaks structures that have proven to result in solutions with high objective values.

In the last heuristic, the *historical request-pair removal* heuristic, the occurrence of request i and j in the same route amongst the set of B best solutions is determined. Initially, a request is randomly selected to be placed in the destroy neighbourhood. Based on the historical occurrence of the requests in the same routes, a request is added to the destroy neighbourhood using, again, the randomisation parameter as described in the *worst removal* heuristic. This process is repeated, where every iteration a request is randomly selected from the destroy neighbourhood, until the size of the destroy neighbourhood equals q.

3.3.2 The Repair Neighbourhood

After a set of requests is removed from their route and added to the request bank, the aim is to insert all requests from the request bank into the routes. For this, two types of heuristics are used.

The *basic greedy* heuristic inserts the requests into routes based on the minimum change in the objective value. Here, the change in the objective value is computed for inserting each request in each route of which the distance of the best location is stored. This is a greedy heuristic as it executes the placement of the least costs option. However, this might result in a bad solution as the possibilities for insertions become more limited when more requests are inserted. Hence, a

regret heuristic is introduced. In this heuristic, the request is inserted for which the regret of not inserting the request in this iteration is maximised. The *regret-2* heuristic compares the costs of placing a request in the best route possible with the cost of inserting it in the second best route, i.e. request u is removed for $u = \arg\max_{i \in U}(\Delta f_i^2 - \Delta f_i^1)$. Here, U is the set of requests on the request bank and Δf_i^j the change in the objective value by inserting request i in the j^{th} best route.

To incorporate randomisation in the repair heuristics, both heuristics can contain a noise term in their objective value. Hence, when choosing a repair heuristic, the choice should also be made whether the heuristic is executed with or without noise. When the heuristic with noise is executed, a random number on $[-N_{max}, N_{max}]$ is added to the objective value used to determine the best placement, where $N_{max} = \eta * d_{max}$ with d_{max} the maximum distance in the solution and η the size of the noise desired in the solution.

3.3.3 Choosing the Destroy and Repair Heuristic

The destroy and repair heuristic are chosen independently and randomly based on their weights, π_i, with i the heuristic number. Initially, all heuristics receive the same weight. Based on the performance of the heuristics, the weights are updated every 100 iterations. Here, credit is given to heuristics that result in a new global solution (σ_1), heuristics that find a new unique solution that improves the current objective value and is accepted (σ_2), and heuristics that find a new unique solution that does not improve the current objective value but is accepted nonetheless (σ_3). Finding new unique solutions is awarded as this indicates a diversification that could aid in escaping local optima.

After 100 iterations, the new weights are computed as $\pi_i = \rho * \frac{\tilde{\pi}_i}{n_i} + (1 - \rho)\pi_i$. The parameter ρ is used to determine the dependence of the weight on the performance of the last 100 iterations relative to the weight used for the decision on the heuristics in these past 100 iterations. The sum of rewards given to the heuristic over the past 100 iterations is denoted by $\tilde{\pi}_i$ and n_i denotes how often the heuristic was chosen in the past 100 iterations, such that $\frac{\tilde{\pi}_i}{n_i}$ gives the average reward of the heuristic over the past 100 iterations.

3.3.4 Vehicle Minimisation

For every route in the solution, a vehicle and employee should be available. Hence, the aim is to limit the number of vehicles in the solution besides merely minimising the total distance of all routes. As the heuristic is not able to both minimise the number of vehicles and the objective value in terms of the distance travelled, the number of vehicles will be minimised using the ALNS before the distance is minimised.

To minimise the number of vehicles, the starting point is a feasible solution in which all requests are placed in a route using the *regret-2* heuristic. One of the routes is randomly selected to be removed from the solution. The ALNS is applied to this solution until either a solution is found in which all requests are placed in the routes or τ iterations are reached. If the route could not be removed, it is placed on the tabu list, which is emptied when a route removal is successful. This process is repeated until the maximum number of iterations is reached for the

vehicle minimisation process or all routes are on the tabu list. The number of vehicles in the feasible solution is the starting point for the ALNS distance minimisation stage.

It is not always possible to execute a vehicle minimisation stage, e.g. for the SDVRP. Hence, this step is only included in the heuristic when possible.

3.3.5 The Initial Solution for the Distance Minimisation Stage

To find an initial solution, the *regret-2* heuristic is used. In case the vehicle minimisation stage is executed, the number of routes is limited to the number of vehicles resulting from that stage. When this stage is not included, the number of available routes is set to the number of routes available according to the data set. In case not all requests could be placed in a route, the requests are placed on the request bank.

3.3.6 The Parameters

Pisinger & Ropke use the same parameters as they did in a similar heuristic in Ropke & Pisinger (2006a). This implies that the parameters are chosen as follows: $(B, c, w, \rho, p, \sigma_1, \sigma_2, \sigma_3, \eta, \tau) = (100, 0.99975, 0.05, 0.1, 3, \frac{33}{55}, \frac{9}{55}, \frac{13}{55}, 0.0025, 2000)$.

Pisinger & Ropke do vary the simulated annealing parameters w, used to find the starting temperature, and c, the cooling rate, between the vehicle minimisation stage and the distance minimisation stage. Hence, for the vehicle minimisation stage, the performance of five values for each w and c will be evaluated and the best combination will be used.

To increase the replicability of the results presented in this paper, the seeds will be initialised. Here, the seeds used are randomly generated prime numbers, namely, (150659, 332803, 418219, 415993, 68371, 186917, 41, 56081, 609599, 218527).

For both the vehicle minimisation stage and the distance minimisation stage, the number of iterations that will be used are 25000.

4 Extending the Heuristic of Pisinger & Ropke

As the EVRP has gained popularity due to recent developments regarding global warming awareness and the quality of electric vehicles, more recent research focuses on constructing routes that include the battery level of these vehicles (Desaulniers et al., 2016; Hiermann et al., 2016; Montoya et al., 2017; Schneider et al., 2014). Therefore, it would be of interest to extend the heuristic of Pisinger & Ropke such that it can also be used to solve the EVRP. First, some additional notation to extend the RPDPTW to also include the EVRP as special case is introduced. Thereafter, the alterations to the ALNS will be explained and two additional destroy heuristics will be provided.

4.1 The Electric Rich Pick-Up and Delivery Problem with Time Windows

To extend the heuristic to the EVRP, the RPDPTW is extended such that positive battery levels are ensured throughout the constructed routes. Furthermore, the possibility to visit charging stations should be included in the model and the service time at this charging station should

be considered. Hence, an electric rich pick-up and delivery vehicle routing problem with time windows (ERPDPTW) is constructed in accordance with the model of Schneider et al. (2014). The model of Schneider et al. (2014) does however only consider single vehicle problems, whereas the aim of the extended ALNS is that larger instances of the EVRP can be solved.

To model the battery charge level and service time, several assumptions have to be made. First, it is assumed that the battery charge level decreases linearly in the distance travelled. This is denoted by a charge consumption rate for vehicle $k \in K$ of h_k. Second, it is assumed that the electric vehicles exhibit a linear recharging rate denoted by g_k and that vehicles will remain at the charging station until they are fully charged. Finally, at the start of the time window at the depot, it is assumed that the vehicles are fully charged.

Following the notation of Pisinger & Ropke and the model described in Section 3.1, the fleet is denoted by the set K and the vehicles have a battery capacity equal to Q_k. An additional decision variable, y_{ik}, is introduced to denote the charging level of vehicle $k \in K$ when arriving at node $i \in V'_k$, where $V'_k = V_k \cup CS$ is the set containing all nodes that can be visited by vehicle $k \in K$. Here CS denotes the set of charging stations. This results in the arc set $A'_k = V'_k \times V'_k$.

Using the notation introduced above, constraints can be added to the model given in Section 3.1 to transform this to the ERPDPTW. Here, constraint (17) is used to update the charging levels when commuting from node $i \in V'_k$ to node $j \in V'_k$. This implies that commuting from node $i \in V'_k$ to node $j \in V'_k$ reduces the battery level by $h_k d_{ij}$, i.e. the distance travelled multiplied by the charge consumption rate. When no direct arc between the two nodes exists, no bound should be placed on the relationship between the charge level at the two nodes. Hence, the constraint exhibits the principle of a big-M constraint. As a small feasible region is desired, the battery capacity is used. Constraint (18) ensures that after visiting a charging station or leaving the start terminal of the vehicle, the battery is fully charged.

$$0 \leq y_{jk} \leq y_{ik} - h_k d_{ij} x_{ijk} + Q_k(1 - x_{ijk}) \qquad k \in K, (i,j) \in A'_k \qquad (17)$$

$$0 \leq y_{jk} \leq Q_k - h_k d_{ij} x_{ijk} \qquad k \in K, j \in V'_k, i \in CS \cup \tau_k, i \neq j \qquad (18)$$

$$S_{ik} + t_{ij} x_{ijk} + g_k(Q_k - y_{ik}) - b_0(1 - x_{ijk}) \leq S_{jk} \qquad k \in K, i \in CS, j \in V'_k, i \neq j \qquad (19)$$

As the service time at a charging station is dependent on the charging level of the vehicle upon reaching the charging station, an additional constraint regarding the service times after visiting the charging station should be added. This is captured in constraint (19). Again, this constraint captures the case where the arc $(i,j) \in A'_k$ with $i \in CS$ is part of the solution and the service time at the charging station should be considered, as well as the case where $x_{ijk} = 0$ and hence bounds on the service time are implicitly imposed. To limit the size of the feasible region, the difference between the start of the service times at node i and j, S_{ik} and S_{jk}, is set to the the route duration, i.e. the end of the time window at the depot, b_0.

4.1.1 Transformations to the Electric Rich Pick-Up and Delivery Vehicle Routing Problem with Time Windows

To apply the ALNS to the different types of the VRP, they will be transformed to the ERPDPTW. The EVRPTW is transformed to the ERPDPTW in the same way as the VRPTW

is transformed to the RPDPTW with the addition of the creation of nodes for the charging stations. Furthermore, the vehicles should also contain information on their battery capacity, recharge rate and charge consumption rate.

The ERPDPTW can still be used to solve the other types of the VRP - the VRPTW, the CVRP, the OVRP, the MDVRP, and the SDVRP. For these problems, the charge consumption rate and the recharging rate can simply be set to zero, as well as the battery capacity. As there are no charging stations in these problems, the set CS is empty and hence constraints (18) and (19) are never imposed. This leaves constraint (17) which, as the charge consumption rate h is set to zero, imposes no additional restrictions compared to the model defined in Section 3.1.

4.2 The Extended Adaptive Large Neighbourhood Search

To apply the ALNS to an ERPDPTW, alterations to the heuristic should be made, resulting in the Extended Adaptive Large Neighbourhood Search (EALNS). Here, the feasibility of routes, not only in terms of vehicle load capacities and time windows, but also in terms of battery level, should be ensured. Furthermore, the addition of battery levels and charging stations to the problem increases the difficulty of optimising the problem. Hence, two additional destroy heuristics are introduced with the aim of improving the solution.

4.2.1 Maintaining Feasibility of Routes

When a repair heuristic is executed and the addition of a request to a route appears to be infeasible only in terms of the battery level, the possibility of adding the request with the help of one of the charging stations is evaluated. Here, the objective value of the addition of every charging station at any location in the route is evaluated to find the best feasible location for a charging station in the route, if any.

At the same time, when a request is removed from a route in the destroy heuristics, the possibility of removing (one of) the charging stations from this route is evaluated. When this is possible, the charging station is removed that leads to the best reduction in the objective value upon its removal. This process is repeated until either none of the charging stations can be removed from the route or the route does not contain any charging stations.

4.2.2 Additional Destroy Heuristics

It is possible that due to the structure of the repair heuristics, similar structures are formed around a certain charging station. With the aim of breaking this pattern, two additional heuristics are introduced that aim to remove requests that are in close proximity of the charging station. As these heuristics aim to intensify the search based on the current charging station structure, possible removals of charging stations will not be executed as in the other seven destroy heuristics.

In the *charging station cluster removal* (CSCR) heuristic, the order of the visits to charging stations relative to the deliveries as well as the proximity to the delivery locations are used to determine the destroy neighbourhood. This implies that, for each request served in a route that visits at least one charging station, a relatedness measure, $R_{i,cs}$ is determined, with $cs =$

$\arg\min_{cs \in CS} |r_{cs} - r_i|$. Here, r_i is the rank of delivery node i in the route. Hence, cs is the charging station that is scheduled to be visited closest to the delivery of the request. The relatedness measure is then determined as $R_{i,cs} = (r_{cs} - r_i)^2 * d(i, cs)$, with $d(i, cs)$ the distance between delivery node i and the charging station cs. The requests are sorted in ascending order of this relatedness measure and removed according to the randomisation parameter as described in Section 3.3.1.

The second heuristic that is added to the EALNS is the *charging station relatedness removal* (CSRR) heuristic. This heuristic has a similar aim as the CSCR heuristic, however the ranking of the requests is ignored and requests can be removed based on the relatedness with a charging station even if it is located in a different route. Hence, for each route and each request, the relatedness is determined in terms of the distance between the charging station and the delivery location. Then, the requests are removed that are located closest to charging stations using a randomisation parameter.

The aim of the CSCR heuristic is to remove requests that are located close to a charging station and are served closely before or after the charging station is visited. This will result in a type of cluster removal and could possibly result in a new solution where the delivery nodes located near the charging station with high relatedness can be rearranged. The aim of the CSRR is to improve the solution by interchanging requests from different routes that are located near the same charging station. A possible limitation of these heuristics could be that that the requests will be inserted in the same locations due to time window restrictions.

To evaluate the performance of the heuristics, they will be individually and jointly tested against the performance of the EALNS using only the original seven destroy heuristics.

5 Data

To evaluate the performance of a new heuristic, benchmark data sets can be used of which the results can be compared with results from other heuristics applied to the same benchmark data sets. Pisinger & Ropke use benchmark instances for all types of the VRP. Hence, the same benchmark data sets are used to evaluate the replicability of the ALNS as explained by Pisinger & Ropke.

For the VRPTW, the data sets from Solomon (1987) and Gehring & Homberger (1999) are used. The Solomon data set contains various classes with 100 instances. Here, the C1 and R1 class will be used. As Pisinger & Ropke report averages for these classes, the averages of these classes will be taken for the purpose of comparison. The Gehring & Homberger data set also contains various classes, however they have various instance sizes. From this data set, the C1 classes with 200 and 400 instances are tested. As with the Solomon classes, averages are reported for comparison purposes.

For the MDVRP, Pisinger & Ropke used the data set from Cordeau et al. (1997). From this data set, the PR class is tested. Contrary to the VRPTW, the values of these problems are reported separately as the problems have different sizes. The different problems vary in size from 48 till 288 instances.

The data set of Cordeau & Laporte (2001) is used to test the performance of the ALNS for the SDVRP. As with the MDVRP, the PR class will be tested for problems varying from 48 till

288 instances.

As comparison with heuristics is desired, the EALNS is tested on a benchmark data set from Schneider et al. (2014). This data set is a modification of the Solomon data set and hence contains 100 instances. To this data set, 21 charging locations are added, 1 at the depot and 20 located randomly throughout areas that can guarantee feasibility of the problem. This implies, that it should be possible to, by using maximally two different charging stations, reach each customer. Also, the vehicle battery capacity, battery consumption rate, and battery recharging rate are initialised for the vehicles.

6 Computational Experiments

In this section, the results of the ALNS and EALNS are presented. First, the results of the ALNS are given and compared to the results of Pisinger & Ropke as to evaluate the replicability of their results. Second, the performance of the EALNS is analysed. Here, the EALNS with the original seven destroy heuristics is compared to the EALNS that contains either one of the additional heuristics and the EALNS that contains both. Additionally, the comparison is made with the results of Schneider et al. (2014) such that the performance of the EALNS can be compared to another heuristic on the same data set.

The heuristic is implemented in Java and experiments are executed using an AMD PRO A6-8750 R5 3500 MHz processors with 2 cores and 16.0 GB RAM.

6.1 The Adaptive Large Neighbourhood Search

Before the heuristic can be applied to the data sets, the parameter setting for the simulated annealing aspect of the vehicle minimisation stage should be determined. Thereafter, the results for the VRPTW, MDVRP and SDVRP are given and compared to the results of Pisinger & Ropke. In determining the best solution found, a lower number of vehicles is prioritised over a shorter total distance of the routes, as an additional route comes with more costs than a larger distance.

Table 1: Total number of vehicles used in C1 (Solomon, 1987) for different parameter settings

		c				
		0.99975	0.999	0.9985	0.998	0.98
	2.50%	156	153	155	154	155
	5%	154	**152**	154	153	159
w	7.50%	155	154	155	**152**	156
	10%	154	154	154	153	159
	12.50%	155	154	154	153	157

6.1.1 Parameter Setting for the Vehicle Minimisation Stage

Five values for the cooling parameter c and the parameter w used for initialising the start temperature are evaluated, resulting in 25 combinations. The test parameters are based on the parameters used in Ropke & Pisinger (2006a). This minimisation stage is applied to the R1

class of the Solomon benchmark data set for the first seed as denoted in Section 3.3.6. The total number of vehicles as a results from this vehicle minimisation stage are given in Table 1. From this table, it can be seen that the performance of the vehicle minimisation stage does depend on the exact parameters that are used. The combinations (5%, 0.999) and (7.50%, 0.998) perform the best as they result in the lowest number of vehicles. The parameter setting (7.50%, 0.998) is randomly chosen to be the parameter setting for the vehicle minimisation stage.

Table 2: Replication results of the Vehicle Routing Problem with Time Windows using Solomon (1987) and Gehring & Homberger (1999) Benchmark Data Sets

	Replicated Results			Pisinger & Ropke (2007)		
	Best of 10	Avg. of 10	T(s)	Best of 10	Avg. of 10	T(s)
C1-100[1]	**862.72**	868.30	2543.3	828.38	828.38	86
	9.66	9.66		10	10	
R1-100[1]	1186.06	1186.60	2420.4	**1213.39**	1216.93	86
	12.75	12.83		**11.92**	12.03	
C1-200[2]	2713.12	2697.90	9525.9	**2732.10**	2732.46	258
	19.1	19.27		**18.9**	18.9	
	Replicated Results			Pisinger & Ropke (2007)		
	Best of 5	Avg. of 5	T(s)	Best of 10	Avg. of 10	T(s)
C1-400[2]	7112.20	7114.09	24115.2	**7369.88**	7450.84	582
	38.5	38.62		**37.6**	37.62	

For each class, the top number represents the average distance of the instances in the class and the bottom number is the average number of vehicles used.

[1] (Solomon, 1987)

[2] (Gehring & Homberger, 1999)

6.1.2 The Vehicle Routing Problem with Time Windows

The results of the ALNS on the Solomon and Gehring & Homberger benchmark data sets are provided in Table 2. Here, the average and best solutions of the 10 replications are provided. For the Gehring & Homberger data set on the C1 class with instances containing 400 instances, the results are replicated in five-fold as opposed to ten-fold due to longer running times. Here, the odd numbered seeds are used from the seeds initialised in Section 3.3.6.

The table shows that the implemented ALNS is able to provide a better results for the C1-100 class compared to Pisinger & Ropke, as the number of vehicles in the solution is lower. However, for the other classes, the results of Pisinger & Ropke are better as they require fewer vehicles.

6.1.3 The Multi Depot Vehicle Routing Problem

As with the VRPTW, each problem of the MDVRP is applied in ten-fold and the average and best solution is determined. These results can be found in Table 3. As Pisinger & Ropke did not report the number of vehicles used in their solution, no comparison regarding the number

of vehicles can be made. For some instances, the replicated results are equal, or even slightly better compared to the results of Pisinger & Ropke. This holds for instances PR01, PR02, PR03 and PR07. However, for the other instances, the replicated results are higher then the results presented by Pisinger & Ropke, both in terms of the best solution and the average of the ten solutions.

Table 3: Replication results of the Multi Depot Vehicle Routing Problem using Cordeau et al. (1997) Benchmark Data Sets

	Number of instances	Number of depots	Replicated Results			Pisinger & Ropke (2007)		
			Best of 10	Avg. of 10	T(s)	Best of 10	Avg. of 10	T(s)
PR01	48	4	**861.32**	861.32	51.8	**861.32**	861.32	16
PR02	96	4	**1307.34**	1308.24	194.5	**1307.34**	1311.54	52
PR03	144	4	**1804.76**	1809.28	422.2	1806.53	1810.90	106
PR04	192	4	2081.79	2095.15	625.0	**2066.64**	2080.55	146
PR06	240	4	2374.95	2407.56	729.1	**2341.65**	2352.59	188
PR06	288	4	2733.22	2759.15	998.8	**2685.35**	2695.15	232
PR07	72	6	**1089.56**	1089.56	96.0	**1089.56**	1089.56	29
PR08	144	6	1674.27	1685.90	386.9	**1665.80**	1677.31	105
PR09	216	6	2155.21	2171.06	756.2	**2136.42**	2148.85	173
PR10	288	6	2949.89	3004.68	900.8	**2889.49**	2913.34	228

Table 4: Replication results of the Site Dependent Vehicle Routing Problem using Cordeau & Laporte (2001) Benchmark Data Sets

	Number of instances	Number of vehicle types	Replicated Results			Pisinger & Ropke (2007)		
			Best of 10	Avg. of 10	T(s)	Best of 10	Avg. of 10	T(s)
PR01	48	4	**1369.74**	1370.11	33.1	1380.77	1387.37	10
PR02	96	4	**2297.14**	2306.10	109.0	2311.45	2311.54	32
PR03	144	4	2602.13	2623.67	299.7	**2590.01**	2608.31	71
PR04	192	4	3482.85	3523.60	438.4	**3481.44**	3510.26	98
PR06	240	4	4476.70	4518.53	588.8	**4382.65**	4430.28	123
PR06	288	4	4540.24	4574.75	884.0	**4452.93**	4475.52	159
PR07	72	6	**1887.86**	1922.46	53.2	1889.82	1926.52	19
PR08	144	6	2987.08	2999.07	284.6	**2976.76**	3001.88	66
PR09	216	6	3571.16	3604.35	521.1	**3548.22**	3581.58	113
PR10	288	6	4731.90	4766.14	788.9	**4646.96**	4675.65	162

6.1.4 The Site Dependent Vehicle Routing Problem

The results for the SDVRP are presented in Table 4. As with the MDVRP, Pisinger & Ropke did not report the number of vehicles used in their solution, so no comparison regarding the number of vehicle can be made. Again, differences in the results between the replicated model and the model from Pisinger & Ropke exists. The replicated model obtains better solutions for instances PR01, PR02 and PR07. For PR08, no clear conclusion on the relative performance can be given as the best solution of Pisinger & Ropke is better than the best solution for the

replicated model, but the replicated model gives a better result based on the average of the 10 runs. The other instances have a better solution for the implementation of Pisinger & Ropke.

6.1.5 Summarising Results

From the results in the past sections, it can be seen that the replicated ALNS model is not able to reduce the number of vehicles as much as the results from Pisinger & Ropke indicate is possible. This is supported by Table 5, which shows the gap between the replicated results and the results of Pisinger & Ropke. As Pisinger & Ropke do not report the number of vehicles used in their solutions for the MDVRP and the SDVRP, no comparison can be made for the number of vehicles used for these problems. Although the average gap for the VRPTW is negative for both the best and average solution, the results of Pisinger & Ropke are better due to the smaller number of vehicles used in their solution. For the MDVRP and SDVRP, the average gap is quite close to 0 and hence on average, the replicated results were able to attain the results of Pisinger & Ropke. However, the large difference between the minimum and maximum gap for the instances in these two VRP types, shows that the replicated model was not able to consistantly attain the results of Pisinger & Ropke.

Table 5: Comparison of the Replicated Results and the Results of Pisinger & Ropke (2007)

	Number of instances	$\Delta_{Best}(\%)$			$\Delta_{Avg.}(\%)$		
		avg.	min.	max.	avg.	min.	max.
VRPTW[1]	41	-0.77	-3.50	4.15	-1.08	-4.52	4.82
		2.15	-3.33	6.96	2.32	-3.44	6.68
MDVRP	10	0.73	-0.10	2.09	0.98	-0.25	3.14
SDVRP	10	0.59	-0.80	2.15	0.60	-1.24	2.22

For the VRPTW, the top number refers to the distance and the bottom number to the number of vehicles. For the MDVRP and SDVRP, the gap with the number of vehicles is unknown as (Pisinger & Ropke, 2007) so not report the number of vehicles used in their solutions.

[1] : Based on the weighted average of the gap of the C1-100, R1-100, C1-200 and C1-400 class.

Besides differences in the number of vehicles used in the solution and the total distance travelled, differences exist in terms of the running times. Table 2, 3 and 4 show that the implementation of Pisinger & Ropke terminates significantly faster compared to the replicated model. Besides a different type of computer used to execute the experiments, the large difference can most likely be assigned to differences in coding style and memory usage.

6.1.6 Dependency on the Parameter Setting

Section 6.1.1 showed that a small change in the parameter setting can result in differences in the performance of the vehicle minimisation stage. It is likely that the same holds for the second stage, in which the distance is minimised. Also, as can be seen from the differences between the average and best solutions by the ALNS, the seed of the random number generator used to determine, amongst others, the size of the neighbourhood and the heuristics to be used, affects

the result too. To fully replicate the results of Pisinger & Ropke, it is thus seems necessary to have knowledge on the exact parameter setting used in the heuristic.

A difference in the performance of the vehicle minimisation stage could also dependent on the exact implementation of this stage. In this implementation, a route is selected at random as a suggestion for a removal. When this removal does not result in a feasible solution within τ iterations, the route is placed in a tabu list. When a feasible route removal is found, the tabu list is emptied and the process is repeated until either all routes are on the tabu list or the maximum number of iterations is reached. As τ is set to 2000 and the maximum number of iterations for this stage equals 25000, it is likely that the stage is terminated when the maximum number of iterations is reached before all routes are tested for a possible removal based on the latest feasible solution. This implies that the order of the removals could be of influence and it is possible that an alteration in the route selection could result in a lower number of vehicles. Hence, the difference in the results with the results of Pisinger & Ropke could possibly be explained by differences in the implementation of this stage.

6.2 The Extended Adaptive Large Neighbourhood Search

The results of the EALNS are given in Table 6. This table reports the best solution reported by Schneider et al. (2014), the results of the EALNS without the additional destroy heuristics (EALNS-0), the results of the EALNS while including either the CSCR heuristic or the CSRR heuristics (EALNS-1 and EALNS-2, respectively), and the results of the EALNS while incorporating all nine destroy heuristics (EALNS-3). First, the affect of the additional destroy heuristics will be evaluated. Thereafter, the performance of the EALNS models will be compared to the best solutions as reported in Schneider et al. (2014).

6.2.1 The Performance of the Additional Destroy Heuristics

To test the performance of the CSCR heuristic and the CSRR heuristic, the performance of the EALNS without any of the additional heuristics is compared to the EALNS with either or both of the heuristics. For each of these models, the best and average out of the 10 replications are given in Table 6.

Comparing the different models for the different problems shows that the relative performance of the models differs. Hence, the relative performance of the different models will be explained for each problem separately, before a conclusion will be drawn.

For the c101 instance, all models result in the same best solution, except for the EALNS-1, whose solution is very close. Also in terms of the average solution, the models perform equally well. For the c102, all models result in the same best solution, but differences exist between the averages of the solutions. Here, the addition of the CSCR heuristic does not proof beneficial with an average distance that is 1.7% above the average distance found when no additional heuristics are implemented. The addition of the CSRR heuristic also affects the average distance, however this effect is less than 0.10%. Moreover, adding both heuristics to the model results in a worse average solution, which is most likely caused by the poor performance of the CSCR heuristic.

Table 6: Results of the Electric Vehicle Routing Problem with Time Windows using the Schneider et al. (2014) Benchmark Data Set

	Best	EALNS-0			EALNS-1			EALNS-2			EALNS-3		
	solution [1]	Best of 10	Avg. of 10	T(min)	Best of 10	Avg. of 10	T(min)	Best of 10	Avg. of 10	T(min)	Best of 10	Avg. of 10	T(min)
c101	**1053.83**	**1053.83**	1053.87	5.8	1053.85	1053.85	5.8	**1053.83**	1053.85	5.5	**1053.83**	1053.89	6.0
	12	**12**	12		12	12		**12**	12		**12**	12	
c102	1056.47	**1056.12**	1033.61	13.1	**1056.12**	1051.16	12.7	**1056.12**	1034.47	11.2	**1056.12**	1037.29	13.4
	11	**11**	11.70		**11**	11.30		**11**	11.67		**11**	11.60	
c103	**1041.55**	1001.81	1008.07	24.6	1003.26	1007.29	23.8	1001.81	1006.59	25.4	1002.89	1006.62	25.3
	10	11	11		11	11		11	11		11	11	
c104	979.51	970.18	974.53	48.4	970.18	971.77	52.5	970.90	974.53	45.3	**961.65**	973.43	49.4
	10	11	11		11	11		10	10.90		**10**	10.90	
c105	**1075.37**	1075.73	1071.95	8.9	1078.22	1066.63	8.7	1075.73	1081.53	8.4	1075.73	1085.66	8.9
	11	11	11.20		11	11.50		11	11.10		11	11.10	
c106	1057.87	**1031.56**	1052.77	13.0	**1031.56**	1046.48	12.9	**1031.56**	1051.15	13.2	**1031.56**	1040.70	12.9
	11	**11**	11		**11**	11		**11**	11		**11**	11	
c107	**1031.56**	**1031.56**	1048.72	13.3	**1031.56**	1053.22	12.5	**1031.56**	1047.49	14.3	**1031.56**	1053.93	13.1
	11	**11**	11		**11**	11		**11**	11		**11**	11	
c108	**1100.32**	1015.68	1017.77	17.2	1015.68	1020.54	15.9	1015.68	1016.43	16.9	1015.68	1020.36	16.9
	10	11	11		11	11		11	11		11	11	
c109	**1036.64**	993.29	999.08	21.9	994.25	997.97	20.9	995.03	998.63	22.1	994.25	999.02	21.4
	10	11	11		11	11		11	11		11	11	

For each class, the top number represents the average distance of the instances in the class and the bottom number is the average number of vehicles used.
EALNS-0: The Extended Adaptive Large Neighbourhood Search with the seven original destroy heuristics
EALNS-1: The Extended Adaptive Large Neighbourhood Search with the seven original destroy heuristics and the *charging station cluster removal* heuristic
EALNS-2: The Extended Adaptive Large Neighbourhood Search with the seven original destroy heuristics and the *charging station relatedness removal* heuristic
EALNS-3: The Extended Adaptive Large Neighbourhood Search with all nine heuristics
[1]: as reported in Schneider et al. (2014)

For the c103 instance, the EALNS-0 and EALNS-2 result in the best solutions with minor increases in the best solution for the EALNS-1 and EALNS-3. However, although the EALNS-0 results in the same best solution as the EALNS-2, its average solution is the worst amongst the four models. Here, the EALNS-2 has the best average solution closely followed by the EALNS-3. So again, it appears that the CSRR heuristic could possibly be an addition to the model, where as the CSCR heuristic has not proven to be beneficial yet.

For the c104 instance, the EALNS-2 and EALNS-3 result in solutions that require a vehicle less compared to the EALNS-0 and EALNS-1. Here, the EALNS-3 results in the best solution both in terms of the best solution out of the 10 replications and the average solution. Similarly, for the c105 problem, the average number of vehicle used in the 10 replications is lower for the EALNS-2 and EALNS-3. Here, the best solutions of the EALNS-0, EALNS-2 and EALNS-3 are equal and better than the solution of the EALNS-1. The best average solution is found for the EALNS-2. These two problems thus indicate that the CSRR heuristic is a good addition to the model when trying the minimise the number of vehicles used in the solution.

For the c106, c107 and c108 instances, all models result in the same best solution. When comparing the average solutions, the EALNS-3, EALNS-2 and EALNS-2 model appear the best for the c106, c107 and c108 instances, respectively. However, where the EALNS-3 model has the best average solution for the c106 instance, it gives the worst average solution for the c107 instance. For the c106 and c108 instances, the worst performance is for the EALNS-0 and EALNS-1, respectively.

Finally, for the c109 instance, the best solution for the EALNS-0 is the best, but it has the worst average solution with the EALNS-1 model obtaining the best average solution. Contrary to the other instances, the CSCR heuristic does perform well for this instance.

Overall, it appears that the addition of the CSRR heuristic to the seven original heuristics is beneficial as often the EALNS-2 and EALNS-3 result in the best or one of the best solutions. Especially the better performance in terms of vehicle reduction is desired. On the other hand, the CSCR heuristic has not proven to be a good addition to the EALNS. When used in combination with the CSRR heuristic, the results are amongst the best, but the EALNS-1 only outperforms all other models based on the average distance for the c109 instance. As the EALNS-3 does perform well and includes the CSCR heuristic, completely disregarding the potential of the heuristic is not desirable. However, more testing should be done to find potential improvements to this heuristic.

6.2.2 The Performance of the Extended Model

To analyse the performance of the EALNS models, their results is compared to the best known results as reported in Schneider et al. (2014). These can also be found in Table 6. As with the VRPTW, it appears that the heuristic did not succeed to minimise the number of required vehicles as much as the best solution of Schneider et al.. Hence, in three of the nine reported instances (c103, c108 and c109), the best known results are better than any of the EALNS models. In one case, the c104 instance, two of the EALNS models succeeded in attaining the same number of vehicles as the solution reported by Schneider et al., whereas the other two EALNS models did not succeed at this. In all other instances except for c105, the best solution

is attained or beaten by (one of) the EALNS models. For the c105 instance, the results are worse. However, the solutions from three out of four of the EALNS models are only 0.033% above the best solution from Schneider et al. (2014).

Similar to the VRPTW results, the main difficulty for the implementation of the model is reducing the number of vehicles. However, when the heuristic succeeds at reducing the number of vehicles, the heuristic performs well in finding routes as to minimise the total distance driven.

Schneider et al. (2014) do not report the running times of the best known solutions and only report the average running time of the various heuristics tested. Therefore, the running time of the different EALNS types cannot be compared with the running time to obtain the best known solutions. From Table 6 differences between the running times can be seen. It is however not possible to draw a clear conclusion on the relative performance of the EALNS types regarding their running time, as none of the heuristics consistently outperforms the other heuristics. As the code of the EALNS is based on the code from the replicated ALNS and these running times were significantly higher than the running times of Pisinger & Ropke, running times of the EALNS can be significantly reduced by improving coding style and memory usage.

7 Conclusion

Over the years, the number of deliveries have increased. Due to the set of limitations that have to be considered by the delivery companies, the need for fast heuristic methods arises. As global warming awareness has increased over the years as well as the availability of good electric vehicles, the usage of electric vehicles by delivery companies is of interest. However, the batteries of electric vehicles have a limited capacity and the availability of locations to charge electric vehicles are limited. Hence, to avoid large detours to charging stations, it is beneficial to schedule visits to the charging stations in the routes of the vehicles. Therefore, the aim of this paper was to extend a heuristic for solving different types of vehicle routing problems to electric vehicle routing problems.

The Adaptive Large Neighbourhood Search created by Pisinger & Ropke (2007) has proven to perform well and is able to handle various types of vehicle routing problems. Due to before mentioned reasons, an extension of this model to electric vehicle routing problems would be beneficial. As the replicability of a paper is an important aspect, the heuristic from Pisinger & Ropke was implemented and several results compared to the results as found by Pisinger & Ropke before alterations were made to fit the electric vehicle routing problem.

The Adaptive Large Neighbourhood Search is a heuristic in which a set of requests are removed from the solution, the destroy neighbourhood, and subsequently added to the solution. For destroying and repairing solutions, various heuristics are available that aim at diversifying and intensifying the solution. This large neighbourhood search is combined with aspects from simulated annealing with the aim of minimising the distance. As it is desired to have as few routes as possible, even if this results in a longer distance, a vehicle minimisation stage is executed before the heuristic is used to minimise the distance.

The comparison of the replicated results with the results of Pisinger & Ropke showed that the replicability of the paper could be improved. Testing different parameters for the vehicle minimisation stage has shown that slight changes in the parameters affect the solution. Further-

more, the random number generator has an effect on the quality of the solution too. Although good results are obtained from the replicated model, it did not always succeed in attaining the results from Pisinger & Ropke. Additionally, it is likely that small implementation and interpretation differences in the various aspects such as the vehicle minimisation stage and the different destroy and repair heuristics of the Adaptive Large Neighbourhood Search add to differences in the results.

To include the battery level of the vehicles in the problem, the heuristic had to be adjusted to an Extended Adaptive Large Neighbourhood Search. As with the replicated model, this heuristic could not always succeed at reducing the number of vehicles as much as possible. However, when the heuristic succeeded at reducing the number of vehicles as much as the results reported in Schneider et al. (2014), the model attained or improved the solutions.

In the Extended Adaptive Large Neighbourhood, two additional destroy heuristics, the *charging station cluster removal* heuristic and the *charging station relatedness removal* heuristic, were tested. These results showed that including a heuristic specified to the electric vehicle routing problem could result in improvements to using the seven original destroy heuristics, where especially the *charging station relatedness removal* heuristic appeared to contribute to the improved solutions.

From the analysis in this paper, it can thus be concluded that the Adaptive Large Neighbourhood Search of Pisinger & Ropke performs well at solving various types of vehicle routing problems. However, to improve replicability, more clarification on aspects such as the exact parameter setting and execution of the heuristics would add to the replicability of the heuristic. Furthermore, the extended heuristic performs well on electric vehicle routing problems. Also, adding destroy heuristics to the model that focus on this type of vehicle routing problems could have a positive effect on the solution.

The implemented Adaptive Large Neighbourhood Search did not succeed at reducing the number of required vehicles as much as the original model of Pisinger & Ropke and the results for the electric vehicle routing problem reported by Schneider et al. (2014). Hence, a suggestion for future research would be to improve the vehicle minimisation stage such that better solutions can be found for some of the benchmark problems.

Besides the possibility for improvement in the vehicle minimisation stage, there is potentially room for improvement in the additional heuristics for the electric vehicle routing problem. Therefore, a suggestion for future research would be to test various alterations to the implementation of the two additional destroy heuristics or test a different type of heuristic aimed at improving the structure of routes related to charging stations.

Although good results are obtained by the Extended Adaptive Large Neighbourhood, there are some limitations to the model. Some assumptions are made in this paper related to the electrical vehicle routing problem that affect the practical implementation of the model. First, a linear charge consumption rate is assumed although there are various elements that have been shown to affect the performance of the electric vehicle battery, e.g. the vehicle load, speed and acceleration (Lin et al., 2016). Second, the recharging rate is assumed to be linear while Montoya et al. (2017) indicate that this assumption is not valid. Third, it is assumed that electric vehicles will remain at the charging station until fully charged. It might however be of interest what the

improvement in the solution would be if this restriction is not imposed in the heuristic as done by Desaulniers et al. (2016). Finally, electric vehicles are assumed to depart the depot with a fully charged battery while their battery could possibly be only partially charged depending on the time that the vehicles have been at the depot since their last route. Hence, modelling for multiple time blocks in which employees work could provided steady-state results on the route distances when this assumption might not necessarily hold.

References

Baldacci, R., Mingozzi, A., & Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations research*, *59*(5), 1269–1283.

Bektaş, T., & Laporte, G. (2011). The Pollution-Routing Problem. *Transportation Research Part B: Methodological*, *45*(8), 1232–1250.

Bell, J. E., & McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, *18*(1), 41–48.

Bouma, K. (May 2 2019). *Amsterdam wil benzineauto's verbieden vanaf 2030.* `https://www.volkskrant.nl/nieuws-achtergrond/amsterdam-wil-benzineauto-s-verbieden-vanaf-2030 ba48c361/?referer=https%3A%2F%2Fwww.google.nl%2F`. (Last visited 2019-05-14)

Chen, Z.-L., & Xu, H. (2006). Dynamic Column Generation for Dynamic Vehicle Routing with Time Windows. *Transportation Science*, *40*(1), 74–88.

Choi, E., & Tcha, D.-W. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, *34*(7), 2080–2095.

Contardo, C., & Martinelli, R. (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, *12*, 129–146.

Cordeau, J.-F., Gendreau, M., & Laporte, G. (1997). A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems. *Networks: An International Journal*, *30*(2), 105–119.

Cordeau, J.-F., & Laporte, G. (2001). A tabu search algorithm for the site dependent vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, *39*(3), 292–298.

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.

Czech, Z. J., & Czarnas, P. (2002). Parallel simulated annealing for the vehicle routing problem with time windows. In *Proceedings 10th euromicro workshop on parallel, distributed and network-based processing* (pp. 376–383).

Dayarian, I., Crainic, T. G., Gendreau, M., & Rei, W. (2015). A Column Generation Approach for a Multi-Attribute Vehicle Routing Problem. *European Journal of Operational Research*, *241*(3), 888–906.

Desaulniers, G., Errico, F., Irnich, S., & Schneider, M. (2016). Exact Algorithms for Electric Vehicle-Routing P roblems with Time Windows. *Operations Research*, *64*(6), 1388–1405.

Fleszar, K., Osman, I. H., & Hindi, K. S. (2009). A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, *195*(3), 803–809.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., & Werneck, R. F. (2006). Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming*, *106*(3), 491–511.

Gehring, H., & Homberger, J. (1999). A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. In *Proceedings of eurogen99* (Vol. 2, pp. 57–64).

Gendreau, M., Hertz, A., & Laporte, G. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, *40*(10), 1276–1290.

Hiermann, G., Puchinger, J., Ropke, S., & Hartl, R. F. (2016). The Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and Recharging Stations. *European Journal of Operational Research*, *252*(3), 995–1018.

Ho, W., Ho, G. T., Ji, P., & Lau, H. C. (2008). A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering Applications of Artificial Intelligence*, *21*(4), 548–557.

Kottasová, I. (May 9 2019). *Volkwagen's systems can't cope with the 10,000 electric car orders.* `https://edition.cnn.com/2019/05/09/business/volkswagen-id-electric-car-reservation/index.html`. (Last visited 2019-05-14)

Kruskal, J. B. (1956). On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical society*, *7*(1), 48–50.

Kytöjoki, J., Nuortio, T., Bräysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, *34*(9), 2743–2757.

Letchford, A. N., Lysgaard, J., & Eglese, R. W. (2007). A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Society*, *58*(12), 1642–1651.

Lin, J., Zhou, W., & Wolfson, O. (2016). Electric vehicle routing problem. *Transportation Research Procedia*, *12*, 508–521.

Lysgaard, J., Letchford, A. N., & Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, *100*(2), 423–445.

Mester, D., & Bräysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, *32*(6), 1593–1614.

Montoya, A., Guéret, C., Mendoza, J. E., & Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, *103*, 87–110.

Pecin, D., Pessoa, A., Poggi, M., & Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, *9*(1), 61–100.

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

Ropke, S., & Pisinger, D. (2006a). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation science*, *40*(4), 455–472.

Ropke, S., & Pisinger, D. (2006b). A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. *European Journal of Operational Research*, *171*(3), 750–775.

Salimifard, K., Shahbandarzadeh, H., & Raeesi, R. (2012). Green Transportation and the Role of Operation Research. In *Int. conf. traffic transp. eng.(ictte 2012)* (Vol. 26, pp. 74–79).

Sbihi, A., & Eglese, R. W. (2007). The Relationship between Vehicle Routing & Scheduling and Green Logistics - A Literature Survey.

Schneider, M., Stenger, A., & Goeke, D. (2014). The Eelectric Vehicle-Routing Problem with Time Windows and Recharging Stations. *Transportation Science*, *48*(4), 500–520.

Sigurd, M., Pisinger, D., & Sig, M. (2004). Scheduling Transportation of Live Animals to Avoid the Spread of Diseases. *Transportation Science*, *38*(2), 197–209.

Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, *35*(2), 254–265.

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

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). A Unified Solution Framework for Multi-Attribute Vehicle Routing Problems. *European Journal of Operational Research*, *234*(3), 658–673.

Visbeen, K. (March 7 2018). *Arnhem krijgt strengste milieuzone voor personenauto's: oude dieselauto's vanaf 2019 niet meer welkom in binnenstad.* `https://www.volkskrant.nl/nieuws-achtergrond/arnhem-krijgt-strengste-milieuzone-voor-personenauto-s-oude-dieselauto-s-vanaf-2019-niet-meer-welkom-in-binnenstad~bf29f482/`. (Last visited 2019-05-14)

Yu, B., Yang, Z.-Z., & Yao, B. (2009). An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, *196*(1), 171–176.

Appendix

The ALNS and EALNS are implemented in Java. Below, a short description of the packages and classes used for the ALNS is presented.
Heuristics: This package contains the classes that will execute the different destroy and repair heuristics.

- **DestroyHeuristic:** This class contains the methods for the different destroy heuristics used in the ALNS/EALNS.
- **RepairHeuristic:** This class contains the methods for the different repair heuristics used in the ALNS/EALNS.

MasterProgram: This package contains the body of the ALNS.

- **ALNS:** This class is the master program of the adaptive large neighbourhood which minimises the number of vehicles, initialises the starting solution, determines the destroy and repair heuristics that should be used, and executes the simulated annealing aspect of the adaptive large neighbourhood search.
- **Main:** This main method is used to execute the adaptive large neighbourhood search. Here, the txt file containing the information of the instance is read and the data is converted to the (E)RPDPTW, where after the ALNS is initialised and executed.

Problem: This package contains the classes that store the problem and solutions.

- **Location:** This class is used to store the information on a location that should be visited.
- **OldSolution:** This class stores some information of previous solutions needed for the historical heuristics.
- **Request:** This class stores a request, i.e., the location of the pickUp and delivery as well as the request pairs belonging to the request.
- **RequestPair:** This class stores a request pair used for the relatedness measure and the measure of the historical request-pair removal.
- **Route:** This method stores a route in the solution, updates the route if a request is added or removed from the route, and can check for feasibility and the effect of a request removal or placement.
- **RPDPTW:** This class stores all information of the rich pick-up and delivery problem with time windows.
- **Solution:** This class stores a solution to the rich pick-up and delivery vehicle routing problem with time windows in terms of total distance travelled, the routes currently used in the solution, and the requests that are served and on the request bank.
- **Vehicle:** This class stores the vehicle properties.

Tools: This package contains a set of classes needed to execute the heuristics.

- **Arc:** This class stores an arc needed for the historical arc-removal heuristic.
- **DoublyLinkedList:** This class is for a doubly linked list which is used to store a route.
- **Edge:** This class creates an edge used for clustering a route.
- **EdgeComparator:** This comparator is used to sort edges in ascending order based on their distance.
- **Placement:** This class contains information on a placement of a request into a certain route in terms of the distance of the route in case the request is added and the location of the pick-up and delivery node.
- **PlacementComparator:** This comparator sorts placements in ascending order of their change in the distance incurred when the placement is executed.
- **Removal:** This class stores the effects of a removal of a request from a route in terms of the objective value if this request, is removed from the problem completely, the timing difference with another request, and the arc costs associated to the current solution.
- **RemovalComparatorArcCosts:** Sorts the removals in descending order based on the costs of the arcs adjacent to the request in the current solution.
- **RemovalComparatorObjRemoval:** Sorts the removals in descending order of objective value when the removal is executed.
- **RemovalComparatorTimeDiff:** Sorts the removals based on the time difference with another request.
- **RequestPairComparatorHistoricalRelatedness:** Sorts the request pairs based on historical relatedness.
- **RequestPairComparatorRelatedness:** This class can be used for comparing request pairs based on the relatedness measure of the related removal heuristic.

To solve the EALNS, alterations had to be made to some of the classes. Alterations within classes include e.g. adding an addition read method for EVRPTW txt files in the **Main** and adding the battery capacity, battery consumption rate and battery recharging rate in the **Vehicle** class. Besides in class alterations, the **RPDPTW** class is replaced by the **ERPDPTW** class, the **ALNS** class is replaced by the **EALNS** class and the **RemovalComparatorCSRelatedness** class is added. This class is used to sort the relatedness to a charging station in ascending order.