Electric Vehicle Routing Problem

David Woller (DW)
Tuesday 14:30 - 16:00
Computer Science
wolledav@cvut.cz

Václav Vávra (VV)
Thursday 14:30 - 16:00
Artificial Intelligence
vavravac@fel.cvut.cz

Viktor Kozák (VK)
Thursday 14:30 - 16:00
Artificial Intelligence and Biocybernetics
viktor.kozak@cvut.cz

I. Assignment

A. Problem Statement

This semestral project is motivated by the IEEE WCCI-2020 Competition on Evolutionary Computation for the Electric Vehicle Routing Problem (EVRP) [Mav20]. The content of Section I is, therefore, mostly taken over from the provided technical report [MMT+20].

The EVRP can be described as follows: given a fleet of EVs, we need to find the best possible route for each EV within their battery charge level limits. The EVs must start and end at the central depot and serve a set of customers. The objective is to minimize the total distance traveled, while each customer is visited exactly once, for every EV route the total demand of customers does not exceed the EV's maximal carrying capacity and the total energy consumption does not exceed the EV's maximal battery charge level. All EVs begin and end at the depot, EVs always leave the charging station fully charged (or fully charged an loaded, in case of the depot), and the charging stations (including the depot) can be visited multiple times by any EV.

The EVRP can be mathematically formulated as follows:

$$\min \sum_{i \in V, j \in V, i \neq j} d_{ij} x_{ij}, \tag{1}$$

s.t.

$$\sum_{j \in V, i \neq j} x_{ij} = 1, \forall i \in I, \tag{2}$$

$$\sum_{j \in V, i \neq j} x_{ij} \leq 1, \forall i \in F', \tag{3}$$

$$\sum_{j \in V, i \neq j} x_{ij} - \sum_{j \in V, i \neq j} x_{ji} = 0, \forall \in V, \tag{4}$$

$$u_j \leq u_i - b_i x_{ij} + C(1 - x_{ij}), \forall i \in V, \forall j \in V, i \neq j, \tag{5}$$

$$0 \leq u_i \leq C, \forall i \in V, \tag{6}$$

$$y_j \leq y_i - h d_{ij} x_{ij} + Q(1 - x_{ij}), \forall i \in I, \forall j \in V, i \neq j, \tag{7}$$

$$y_j \leq Q - h d_{ij} x_{ij}, \forall i \in F' \cup \{0\}, \forall j \in V, i \neq j, \tag{8}$$

$$0 \leq y_i \leq Q, \forall i \in V, \tag{9}$$

$$x_{ij} \in \{0, 1\}, \forall j \in V, i \neq j, \tag{10}$$

where $V = \{0 \cup I \cup F'\}$ is a set of nodes and $A = \{(i,j)|i,j \in V, i \neq j\}$ is a set of arcs in the fully connected weighted graph $G = (V, A)$. Set I denotes the set of customers, set F' denotes set of β_i node copies of each charging station $i \in F$ (i.e., $|F'| = \sum_{i \in F} \beta_i$) and 0 denotes the central depot. Then, x_{ij} is a binary decision variable corresponding to usage of the arc from node $i \in V$ to node $j \in V$ and d_{ij} is the weight of this arc. Variables u_i and y_i denote, respectively, the remaining carrying capacity and remaining battery charge level of an EV on its arrival at node $i \in V$. Finally, the constant h denotes the consumption rate of the EVs, C denotes their maximal carrying capacity, Q the maximal battery charge level, and b_i the demand of each customer $i \in I$.

B. Problem Categorization

The EVRP is a challenging $\mathcal{NP}$-hard combinatorial optimization problem as it is an extension of the ordinary shortest path problem incorporating additional constraints. It can be viewed as a combination of two variants of the classical Vehicle Routing Problem (VRP) - the Capacitated Vehicle Routing Problem (CVRP) and the Green Vehicle Routing Problem (GVRP). In the VRP, the goal is to minimize the total distance traveled by a fleet of vehicles/agents, while visiting each customer exactly once. In the CVRP, the customers are assigned an integer-valued positive demand, and the vehicles have limited carrying capacity. Thus, an additional constraint of satisfying all customers' demands while respecting the limited vehicle capacity is added to the VRP. Concerning the GVRP, the terminology is not completely settled, but the common idea among different formulations aims at minimizing the environmental impact, typically by taking into consideration the limited driving range of alternative fuel-powered vehicles and the possibility of refueling at rarely available Alternative Fuel Stations (AFSs).

The EVRP as formulated in [Mav20] has the same objective as the VRP, while incorporating the additional constraints from CVRP and GVRP. It is sometimes alternatively named CGVRP, while the name is EVRP often used for other variants of the GVRP (e.g., with considering the non-linear time characteristic of the recharging process or the influence of carried load on the energy consumption).

II. Related works

According to the [MMT$^+$20], the EVRP variant solved was first formulated in [GSS11]. So far, only three papers are dealing with this problem, and for each one of them, the exact problem formulation slightly varies. The first one is [ZGA18], which limits the maximum number of routes in addition to the previously introduced constraints. It presents the following solution method based on the ant colony system (ACS) algorithm. Initially, a Travelling Salesman Problem-like (TSP) route visiting all customers is constructed. This route is then turned into a valid EVRP route by a newly proposed fixing method. Then, the ACS modifies the underlying TSP route according to the solution fitness score, and the whole process repeats. The second one is [NYBS19], which also considers the maximum total delivery time. The initial valid EVRP tour is constructed from the nearest neighbor based TSP tour visiting all customers. The solution is then improved in a local search phase while using four different neighborhood operators (swap, insert, insert AFS, and delete AFS). The proposed algorithm uses a simulated annealing mechanism to allow for accepting non-improving moves with a certain probability, thus preventing premature convergence to a local optimum. Finally, [Pen19] proposes a novel construction method and a memetic algorithm consisting of a local search and an evolutionary algorithm. Similarly to [NYBS19], the local search combines the Variable Neighborhood Search (VNS) with Simulated Annealing (SA). The operators used in the local search are 2-opt, swap, insert, and inverse. Unlike the previous approaches, the proposed memetic algorithm maintains a whole set of solutions. In each iteration, all solutions are individually improved by the local search, and the set is then split into elite and non-elite solutions. The elite solutions are kept to the next iteration, while the non-elite are partially crossbred with the elite ones and partially discarded.

Various other approaches were successfully applied to the numerous variants of the VRP, and many of these can be adapted to the EVRP formulation solved. For example, a recent survey [ECLR19] focused only on the variants of EVRP, and GVRP presents a total number of 79 papers.

III. Problem solution

A set of 17 testing instances ranging from 22 to 1000 customers was provided in the competition. As the problem is NP-hard, it is impossible to solve the larger instances to optimality in a reasonable time. Therefore, a metaheuristic approach with polynomial time complexity is to be deployed. The EVRP is a constrained variant of the VRP, for which many effective methods were proposed over the years. In such a case, an efficient strategy is to select such a metaheuristic that yields state of the art performance on the original VRP problem and, at the same time, can be easily adapted to respect the additional constraints of the EVRP. These two criteria are met by the metaheuristics performing neighborhood-oriented search such as (Randomized) Variable Neighborhood Descent - (R)VND, Variable Neighborhood Search (VNS), Greedy Randomized Adaptive Search Procedure (GRASP), Iterated Local Search (ILS) and others. From these, VND, RVND, and VNS were chosen for addressing the EVRP and are described in Section III-A.

An integral part of these metaheuristics is local search, which enables systematic search in a neighborhood of a current solution using so-called local search operators. Most of the operators commonly used can be efficiently used in the EVRP, given that a solution validity checking function and an operator cost update function is designed. These functions typically have linear time complexity in a naive implementation, but can often be evaluated in constant time and without an actual application of the

operator. As the competition does limit only the total number of fitness evaluations and not the total running time, it is sufficient to design a constant time cost update function for all of the operators. Local search operators are described in Section III-B.

Another subtask to be addressed is designing a method for the construction of a valid initial solution. Multiple construction methods were developed and compared. Some of these are modified variants of constructions proposed for different variants of the VRP, while others are newly designed specifically for the EVRP. Construction methods are described in Section III-C.

For the purposes of formal components description, let's define an EVRP tour T as a sequence of nodes $T = \{v_0, v_1, ..., v_{n-1}\}$, where v_i is a customer, a depot or an AFS and n is the length of the tour T. Then, let e_{ij} be an edge from node v_i to node v_j and $w_{i,j}$ be its weight.

A. Metaheuristics

1) (Randomized) Variable Neighborhood Descent - (R)VND: VND is a simple metaheuristic used here as a local search method in a more complex metaheuristic VNS (Section III-A2). It has a deterministic variant (VND) and a stochastic one (RVND). Both variants are described in Algorithm 1. The input is a valid EVRP tour $\mathcal{T}$, a sequence of local search operators $\mathcal{N}$, corresponding to different neighborhoods in the search space, and a maximal number of fitness evaluations $evals_{max}$. The operators passed in $\mathcal{N}$ are described in Section III-B.

Both of the variants perform the local search (according to the best-improvement scenario) sequentially in the neighborhoods in $\mathcal{N}$. In the case of the RVND, the order of the neighborhoods is randomly shuffled first (line 3), whereas in the VND, the order remains fixed. Each time and improvement is made, the local search is restarted (line 1). The VND then starts again from the first neighborhood in $\mathcal{N}$, while the RVND randomly reshuffles the neighborhoods first. The algorithm terminates either when no improvement is achieved in any of the neighborhoods or when a stop condition is met. In the EVRP competition, the stop condition is defined by a maximal number of allowed fitness evaluations, which is checked in between changing neighborhoods (line 6).

Algorithm 1: (Randomized) Variable Neighborhood Descent - (R)VND

```
   input: T, N, evals_max
1  i ← 1
2  stop ← false
3  Randomly shuffle N // RVND only
4  while i ≤ |N| do
5  |  T' ← arg min_{T̃ ∈ N_i(T)} Cost(T̃)
6  |  if Cost(T') < Cost(T) then
7  |  |  T ← T̃
8  |  |  i ← 1
9  |  |  Randomly shuffle N // RVND only
10 |  else
11 |  |  i ← i + 1
12 |  Get the total no. of fitness evaluations evals
13 |  if evals ≥ evals_max then
14 |  |  stop ← true
15 |  |  break
16 return T, stop
```

2) Variable Neighborhood Search (VNS): VNS is a metaheuristic method proposed by [MH97], and it is commonly used for approximating solutions of optimization problems such as the VRP. It systematically changes the searched neighborhood in two phases: an exhaustive local search reaching a local optimum and a perturbation phase, which serves to get out of the corresponding valley. The process is described in Algorithm 2.

Algorithm 2: Variable Neighborhood Search (VNS)

```
1 T* ← Construction()
2 while stop = false do
3 |  T ← Perturbation(T*)
4 |  T, stop ← Local_search(T)
5 |  if Cost(T) < Cost(T*) then
6 |  |  T* ← T
7 return T*
```

First, an initial solution is constructed using one of the construction methods described in Section III-C (line 1). Then, the following process repeats. The best known solution $\mathcal{T}^*$ is modified by a randomized perturbation, resulting in a possibly

non-improving current solution $\mathcal{T}$ (line 3). The current solution $\mathcal{T}$ is then subject to a systematic local search, which uses the operators described in Section III-B passed in the $\mathcal{N}$ structure. These operators in the local search phase are applied according to the VND or RVND (Section III-A1) metaheuristics. If the cost of the current solution $\mathcal{T}$ is better than the cost of $\mathcal{T}^*$, it replaces it as the new best solution (line 5). The search process terminates when a stop condition is met. The EVRP competition defines it as a maximal number of fitness function evaluations. When this number is reached within the local search, it terminates and sets the $stop$ flag to $true$, thus terminating also the VNS loop.

The perturbation operator applied at line 3 is intended to move the search process out of the reach of the local search operators while keeping most of the properties of the current best solution $\mathcal{T}^*$. This follows the assumption of the VNS, that local minima with respect to different neighborhoods are relatively similar. For this purpose, the Double-Bridge perturbation, first introduced in [MOF97] for the Travelling Salesman Problem, is used. The Double-Bridge splits the best known solution $\mathcal{T}^*$ into $p+1$ subroutes, according to p randomly selected indices. These subroutes are randomly shuffled, inverted, and reconnected, producing a possibly invalid tour. This tour is then repaired if necessary by the SSF construction described in III-C and passed to the local search as a valid tour $\mathcal{T}$.

B. Local search

A description of individual local search operators corresponding to different neighborhoods follows. The cost update functions δ provided are expressed as a difference between the sum of removed edges weights and the sum of newly added edges weights. Thus, a positive value of the cost update corresponds to an improvement in fitness and vice versa. Two standard modes were tested in the local search - best improvement and first improvement. In the best improvement mode, all possible combinations of input parameters are tested, and the tour with the highest cost update is returned from the current operator-defined neighborhood. In the first improvement, the first improving tour found is accepted, and the search in the current neighborhood is terminated. In both cases, only valid tours are accepted.

1) 2-opt: This is an operator commonly used in many variants of classical planning problems such as TSP or VRP. It takes a pair of indices i, j, and a tour T as an input and returns a modified tour T', where the sequence of nodes from i-th to j-th index is reversed. It must hold, that $i < j$, $i \geq 0$ and $j < n$.

The cost update function δ_{2-opt} can be evaluated as

$$\delta_{2-opt} = w_{i-1,i} + w_{j,j+1} - w_{i-1,j} - w_{i,j+1}, \tag{11}$$

where the indices are expressed w.r.t. to the tour T.

2) 2-string and its variants: This operator is a generalized version of several commonly used operators, which can be obtained by fixing some of the 2-string parameters. The 2-string operator takes five parameters: a tour T, a pair of indices i, j valid w.r.t. to T, and a pair of non-negative integers X, Y. It returns a modified tour T', where the sequence of X nodes following after the i-th node in T is swapped with the sequence of Y nodes following after the j-th node. It must hold, that $i \geq 0$, $j \geq i + X$ and $j + Y \leq n - 1$. The following operators can be derived by fixing the values of X and Y:

- 1-point: $X = 0, Y = 1$
- 2-point: $X = 1, Y = 1$
- 3-point: $X = 1, Y = 2$
- or-opt2: $X = 0, Y = 2$
- or-opt3: $X = 0, Y = 3$
- or-opt4: $X = 0, Y = 4$
- or-opt5: $X = 0, Y = 5$

When performing the local search, the complementary variants of these operators (e.g. 1-point with $X = 1, Y = 0$) are considered as well.

The cost update function $\delta_{2-string}$ can be evaluated as

$$\delta_{2-string} = cut_1 + cut_2 + cut_3 + cut_4 - add_1 - add_2 - add_3 - add_4, \tag{12}$$

where cut_1 corresponds to the edge weight after i-th node in T, cut_2 to the edge after $i + X$, cut_3 to the edge after j and cut_4 to the edge after $j + Y$. Then, add_1 is the weight of the edge added after the index i-th node in T, add_2 of the edge added after the reinserted block of Y nodes, add_3 of the edge added after j and add_4 of the edge added after the reinserted block of the X nodes. For some combinations of the parameters, some of these values evaluates to zero, which must be carefully treated in the implementation. For example, if $X \neq 0$, then $cut_2 = w_{i+X,i+X+1}$, otherwise $cut_2 = 0$.

3) AFS reallocation: This is an operator specifically designed for the GVRP problem and works on individual capacitated subroutes separated by visits to the depot. Since the constructions base the AFS insertion only on the last nodes, where we run out of the battery capacity, their placement is often suboptimal. Moreover, most of the other local search operators only work over the set of nodes already included in the constructed EVRP path and do not allow for the insertion of different AFSs.

An example depicting the reallocation process can be seen in Figure 1. We start by inserting AFSs into the original capacitated subroute similarly as in the *Relaxed ZGA* construction. By running the insertion algorithm for both directions on the subroute we determine a set of valid insertion edges limited by the last nodes before the AFS insertion. We define the insertion cost for AFS k inserted between nodes i and j as

$$I\_Cost_{k,i,j} = w_{i,k} + w_{j,k} - w_{i,j} \tag{13}$$

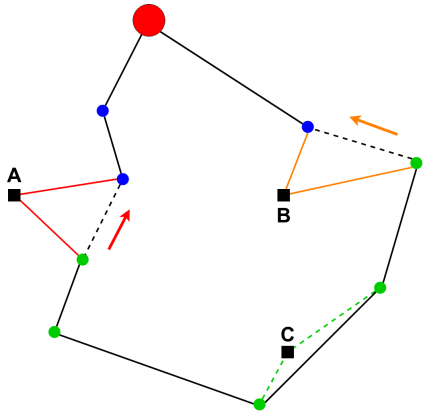


Fig. 1: AFS reallocation

We iterate over the set of insertion edges and determine the AFS with minimal insertion cost. Since the length of the traversed path is directly influenced by the insertion cost, any AFS that is inserted with a lower cost than the original is valid. If there is at least one AFS with a lower insertion cost, we can define the cost update function $\delta_{realloc}$ as

$$\delta_{realloc} = I\_Cost_{original} - I\_Cost_{new} \tag{14}$$

C. Initial constructions

1) Two phase constructions starting with a TSP feasible instance: These constructions are split into two phases, first of which creates a feasible solution to a TSP problem disregarding the AFS nodes and the load and energy constraints. This phase is usually implemented with the nearest neighbor algorithm. However, in general this splitting into two phases allows for convenient modularization in the sense that different methods in each phase can be used independently. Specifically, besides the nearest neighbor algorithm we also tried constructions of the initial TSP feasible solution based on the minimal spanning tree (MST) of the graph and based on Clarke-Wright Savings(CWS) method. That being said the output of the first phase is typically an invalid EVRP tour; therefore, the second phase can be seen as a repair procedure. Several examples of generated TSP and EVRP tours are presented in Figure 3.

a) Separate Sequential Fixing (SSF): In the examples shown, the initial TSP feasible solution was created according to the nearest neighbor rule. The second phase of SSF works is also split in two subphases. In the first subphase, the load constraint is sequentially checked, and whenever the next customer cannot be satisfied, a depot is inserted. Thus, the constraints on the load are fixed in this phase. In the second subphase, the constraints on the battery charge level are fixed, which is slightly more complicated. The current tour is again sequentially checked, and if the next customer is reachable from the current node and the vehicle will not get stuck in it (meaning that it can still reach the closest AFS), the customer is added to the final valid tour. Otherwise, SSF adds the AFS closest to the current customer, the AFS closest to the next customer, and any intermediate AFSs in between, if necessary. After that, the next customer can be safely added. An example of a valid EVRP tour constructed by this method is shown in Figure 3b.

b) Relaxed ZGA: We tried a construction proposed in [ZGA18]. The second phase of the construction adds AFS nodes to the route so that it's feasible in terms of EVRP. It iterates through the TSP route and inserts AFS nodes or depot when needed. It first checks whether the demand of the next customer will be satisfied. If not, the depot is inserted to the route at the current place. If the demand of the next customer is satisfied, then it's checked whether the depot can be reached via the next customer. If not, the nearest AFS is inserted to the route at the current position. If even the nearest AFS cannot be reached, the algorithm backtracks - it goes back in the route being constructed. The suggested algorithm for the second phase is visualized by Figure 4. However, we found out that it doesn't work in general as it can get stuck on some instances. It happens when a) the demand is satisfied, but b) the depot can't be reached via the next customer, c) the closest AFS can be found and after d) adding the

closest AFS this sequence repeats for the same customer. This can happen if there exists a node in the instance such that a route from closest AFS via this node and straight to the depot is too long (demands more energy than the battery charge level). This is actually not a very strong property. It's proved to be a sufficient condition for the infinite cycle to happen if the triangle inequality for the distances among nodes holds. We modified this algorithm so that for energy constraints we checked whether the closest AFS from the next customer can be reached via the next customer (unlike originally when we checked if the depot can be reached) and also for the returns to depot we also counted with routes via AFSs if the depot was not directly reachable. See Figure 5.

c) Method proposed in [Pen19]: We also tried method for construction of an initial feasible solution proposed in [Pen19]. However, here again we found that in general it doesn't work as it can get stuck on some instances, because originally it has the same shortcomings as the ZGA method, namely that sometimes it tries to return straight to depot (in this case when the demand of the next customer cannot be satisfied) disregarding the AFS nodes. Generally it tends to happen when there is a path in the initial TSP tour $n_1, n_2, ...n_k, n_{k+1}$ such that $distance(depot, n_1, n_2, .., n_k, depot)$ is larger than the battery capacity allows, but $distance(depot, n_1, n_2, .., n_k)$ is not larger than that and $demand(n_1, ...n_k)$ can be satisfied without returning to the depot, but $demand(n_1, ...n_k, n_{k+1})$ cannot. For the algorithm to really get stuck it also needs to insert depot before n_1 during the execution. To simplify the situation, the algorithm will positively get stuck on an instance with the only nodes $n_1, ...n_k, n_{k+1}$ with the properties as above. Practically the same fix as to the original ZGA method can be applied to this method to make it work (i.e. make the route return to the depot via AFS when needed).

2) Other constructions:

a) One Route Each (ORE): This construction is meant to be a simple baseline method intended primarily for comparison. It takes a list of all customers to be served as an input. The order of the customers in the input list does not matter, as a separate route from the depot and back is planned for each customer. If the customer cannot be satisfied without recharging, an AFS closest to the customer is added to the route before and after visiting the customer. It is assumed here that all AFSs are directly reachable from the depot, which holds for all of the provided instances. An example of a valid EVRP tour constructed by this method is shown in Figure 3a.

b) Modified Clarke-Wright Savings algorithm (CWS): This method was originally developed for the classical VRP. The initial construction can be likened to ORE where an individual route is created for each customer. The algorithm starts with a node that is furthest from the depot and individual routes are then merged while utilizing the saving distance $S_{i,j}$ defined in equation 15 as a difference between the original and resulting routes.

$$S_{i,j} = w_{i,j} - w_{depot,j} - w_{depot,i} \tag{15}$$

The original CWS algorithm intended for VRP was modified for CVRP by stopping the route merging procedure when there are no more customers that could be connected to the current route without exceeding the maximal capacity, a new route is then created from the remaining customers. The output of this algorithm is a valid CVRP tour, however, the satisfaction of the battery level constraints is not guaranteed, therefore AFSs generally have to be inserted afterward. Since this is not implemented directly in the modified CWS algorithm, AFSs can be inserted using one of the previously developed methods. An example of a valid EVRP tour constructed using CWS initialization can be seen in Figure 3d.

c) Density-based clustering (DBC): This heuristic exploits the spatial properties of VRP and considers the distribution of nodes over space. This allows us to divide the original set of nodes V into several sub-sets and downsize the original planning problem. The implemented DBC algorithm builds on concepts from the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) proposed in [EKSX96]. The algorithm separates nodes based on a given neighborhood radius ϵ and a density threshold $minPts$. We define the ϵ-Neighborhood $N_\epsilon(v_i)$ as a set of nodes within a radius of ϵ from v_i and introduce the condition $|N_\epsilon(v_i)| \geq minPts$ for potential cluster core candidates, wherein all nodes in the ϵ-Neighborhood of a core candidate node v_i are considered as directly density-reachable from v_i. Initial clusters are formed by identifying sets of density-reachable nodes based on the core candidates. Nodes outside of these initial clusters are typically considered as outliers and can be treated as separate entities or assigned to the nearest cluster.

The algorithm generates a number of cluster sets corresponding to each pair of $(\epsilon, minPts)$ parameter combinations. Values of the ϵ parameter are largely dependent on the problem definition, therefore the potential values are set as fractions of the maximal distance that can be traveled by a vehicle on a fully charged battery. Potential values of the $minPts$ parameter are set as integers in an interval $< minPts_{min}, minPts_{max} >$. The generated cluster sets V_i are subsequently augmented with a depot and AFSs as $V_i' = \{V_i \cup 0 \cup F\}$ and used as an input for the developed methods for EVRP tour construction.

IV. Experiments

A. Benchmark Settings

A benchmark set of 17 EVRP instances was provided in the competition. These instances are split into two sets - small (E) and large (X). The number of customers ranges from 21 to 100 in the E-instances and from 142 to 1000 in the X-instances. All of the instances contain exactly one depot, while the number of AFSs varies. A more detailed description of the instances can

be found in [MMT+20]. Section IV-B provides an overall evaluation of developed methods as well as evaluations for both set E and X separately since the results vary significantly for smaller and larger instances.

Concerning the competition evaluation criteria - in case that the implemented (meta)heuristic is stochastic, which is our case, 20 independent runs with random seeds from 1 to 20 are to be performed for each instance. The final ranking is then based on the average of final tour fitness values across the 20 independent runs. The stop condition for each run is defined by the maximum number of fitness evaluations w.r.t. the problem size as $evals_{max} = 25000n$ evaluations, where n is the problem size. For this reason, the algorithm was extensively tuned to obtain the best possible solution after this fixed no. of fitness evaluations and properties such as performance/execution time or time-to-target were not observed. The focus was on finding the best possible selection and configuration of components, which by itself is not an easy task.

The tuning process is documented in detail in Section IV-B. Ideally, all construction methods would be tested with all possible subsets of the implemented local search operators, together with tuning the additional parameters of the VNS metaheuristic. However, such an exhaustive systematic approach is not feasible due to the computational requirements. After combining all separate components and considering both randomized and deterministic variants, there are 17 different constructions in total. Then, the full set of the 10 implemented local search operators have $2^{10} = 1024$ subsets, all of which should be tested with every possible construction, resulting in $17 \times 1024 = 17408$ different methods. As the largest instance with 1000 customers requires circa 5 minutes of computer time and it needs to be solved 20 times for each instance, we would need $5 \times 20 \times 17408 = 1740800$ minutes of CPU time, or 1208 days, for solving only this instance (while neglecting the remaining instances, VNS parameters, and the fact, that the order of LS operators in the VND can influence the performance).

Therefore, a decoupled approach to algorithm tuning was deployed. First, all constructions were compared without the local search phase (Section IV-B1). Then, all of the constructions were tested in the VNS metaheuristic, while using the full set of the local search operators (Section IV-B2). Finally, the best performing subset of the operators was selected and the additional parameters of the VNS were tuned (Section IV-B3). Additionally, the complexity of the individual components w.r.t. the problem size n, no. of AFSs, and no. of fitness evaluations are discussed in Section IV-B4.

B. Results and discussion

Benchmark values of tour fitness were provided for the E-instances, however, these values are not guaranteed to be optimal. In fact, our developed methods proved to generally outperform the provided values. A new set of values was generated to provide benchmark values for both the E and X instances. New benchmark values were chosen as best results from the first 20 runs of the ORE method combined with the full set of the implemented local search operators. The results provided in this section are given as a ratio of the computed fitness in relation to its benchmark value.

1) Constructions comparison: Table I shows a comparison of generated EVRP route scores for individual construction methods averaged over all testing instances. All 17 construction methods generate valid solutions for all instances. Although we could simply rank all construction methods based on scores, we will try to provide more in-depth evaluation, since the selection of the best method can be influenced by several other factors, such as the complexity of said methods (presented in Table V).

ID	Construction method	Constructions only avg. score	Local search avg. score
c0	ORE	7.8082	1.0131
c1	Random-seed NN-based SSF	1.3348	1.0020
c2	Fixed-seed NN-based SSF	1.3091	1.0018
c3	Random-seed Random SSF	4.3478	1.0146
c4	Fixed-seed Random SSF	4.3093	1.0152
c5	Random-seed NN-based ZGA	1.2858	1.0023
c6	Fixed-seed NN-based ZGA	1.2526	1.0023
c7	Random-seed Random ZGA	4.0181	1.0127
c8	Fixed-seed Random ZGA	4.0076	1.0114
c9	CWS-based SSF	1.2430	1.0031
c10	CWS-based ZGA	1.1900	1.0015
c11	NN-based SSF from DBC	1.2857	1.0004
c12	NN-based ZGA from DBC	1.2421	1.0007
c13	CWS-based SSF from DBC	1.1843	1.0003
c14	CWS-based ZGA from DBC	1.1453	0.9997
c15	MST-based SSF	1.3839	1.0027
c16	MST-based ZGA	1.3317	1.0018

TABLE I: Initial constructions with and without local search. Average score over all instances.

Table I shows that solutions generated by several methods such as ORE and Random SSF and Random ZGA are far from optimal due to the approach used in their initial construction. These methods are clearly inferior in the initial construction phase but will become significant later when combined with local search operators. We can separate the remaining construction methods by two main aspects. First is the method used to generate the TSP tour (NN, CWS, and MST), second is the tour repair procedure that creates a valid EVRP tour from the original TSP (SSF or ZGA).

Methods using the TSP tour generated by CWS generally tend to produce better results. A possible reason for this is that CWS is the only method that directly incorporates the depot position and the carrying capacity during the TSP tour creation. Regarding the difference between NN and MST based methods, it is our belief that NN-based TSP is generally more suitable for VRP.

The ZGA tour repairing method proved to generate better results. One of its advantages over SSF is its approach in regard to the carrying capacity, while SSF divides the tour according to the carrying capacity in its first step, ZGA updates the current capacity dynamically during the whole tour repairing process. This means that ZGA sometimes picks depot as the nearest AFS to charge battery, but at the same time it loads the vehicle. This case is not counted for in the SSF method. There is also a slight difference in the AFS selection during the process.

Lastly, we can see that initial constructions originating from clusters generated by the DBC method have a significant chance for improvement over the initial construction methods. Although the improvement is clear this comes at the cost of higher computation complexity. Values for the ϵ and $minPts$ parameters used for the DBC method are shown in Table II. Where the ϵ parameter is problem specific and set using fractions of the maximal distance that can be traveled by a vehicle on a fully charged battery, which we define here as $reach$.

Parameter	Values
ϵ	$[\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{10}, \frac{1}{15}, \frac{1}{20}] \times reach$
$minPts$	2, 3, 4, 5

TABLE II: Parameter values for the DBC method

2) Pairing construction with local search: Results of initial constructions paired with local search operators are shown in Table I. An overall improvement can be seen and differences between generated EVRP routes are within 2% in the average score values. This is in direct contrast with EVRP route scores prior to the local search, where several methods could be considered vastly inferior to others. The importance of combining all initial construction methods with local search indiscriminately can be further demonstrated using tables III and IV, wherein scores for E and X instances are shown separately. The c14 method proved to be reliable on both small and large instances and was judged to have the best overall performance.

ID	avg. score	best score
c0	1.0106	1.0
c11	1.0114	1.0029
c13	1.0114	1.0027
c12	1.0121	1.0037
c14	1.0124	1.0022

TABLE III: Top 5 results on all E-instances

We can see that the ORE method combined with local search generates the best results for smaller instances. By generating individual subroutes for each customer, the initial EVRP route generated by ORE poses fewer restrictions on local search operators and leaves more space for random improvement. Nevertheless, it is not suitable for larger problems since it takes a lot of iterations to get the route close to the optimal solution and the final result shows no advantages when compared to more complex methods.

ID	avg. score	best score
c14	0.9907	0.9806
c2	0.9918	0.9807
c13	0.9925	0.9851
c12	0.9927	0.9824
c15	0.9927	0.9842

TABLE IV: Top 5 results on all X-instances

Creating the initial EVRP route with more complex methods seems advantageous for larger problems, where these outperform less complex methods by a large margin. Meanwhile, these methods might not show the best performance on smaller problems, since initial constructions might lead to local optima and do not leave much operating space for local search operators.

3) Final method tuning: The results shown in Section IV-B2 compare all possible constructions paired with an identical local search procedure. For obtaining these, the VNS metaheuristic with several fixed parameters was used. The results indicates, that the construction c14 (CWS-based ZGA from DBC) migth be the best choice, but the remaining parameters of the VNS were only estimated. As the competition deadline is within one week after this semestral report deadline, the computationally demanding testing process is still in progress. For this reason, only two parameters were separately tuned so far, while the rest is described in Section VI. The two tuned parameters are $merge \in \{true, false\}$ and $firstImprove \in \{true, false\}$. The $merge$ parameter determines, if the algorithm should merge two identical AFSs or two occurences of the depot, if they appear in the EVRP tour next to each other during the search. This can happen, when an AFS becomes redundant or when all customers from one subroute are distributed among other subroutes. As the distance from a node to itself is equal to zero, these redundant occurrences do not increase the solution cost and can be used somewhere else later in the search. On the other hand, having numerous duplicated neighboring nodes increases the tour length and makes the search space larger, which is not desirable. The $firstImprove$ parameter determines, whether a search in an operator-defined neighborhood terminates after reaching the first improving solution, or whether the whole neighborhood is searched exhaustively and the most improving solution is returned. This is a commonly used operator in many metaheuristics addressing classical problems such as the TSP or the VRP.

Each of the parameters has only two admissible values, so they could be easily tested together in all 4 possible combinations. The average relative performance of each setup is shown in Figure 2. Clearly, the best setup is c:14_ls:255_m:1_f:0 (yellow line), which stands for c14 construction, full local search (=with all operators), $merge = true$ and $firstImprove = false$.

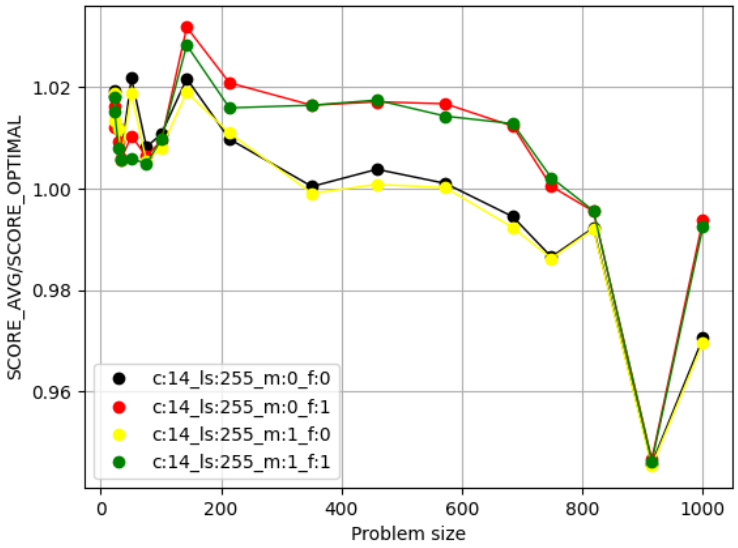


Fig. 2: Pameters tuning - $merge, firstImprove$

4) Complexity: This section provides a complexity analysis of individual components. Defining the problem as $V = \{0 \cup I \cup F\}$ where I denotes the set of customers and F denotes the set of charging stations, we can define $n = \|V\|$, $i = \|I\|$ and $f = \|F\|$ as the size of the problem, number of customers and number of charging stations respectively. Additionally, we define x as an unknown non-negative integer that depends on the specific problem. In our complexity estimates, we work with the assumption that i is significantly higher than f and x. The algorithm stop condition is defined as a maximum number of fitness evaluations $evals_{max} = 25000n$ of the whole EVRP tour and not e.g., time or lower bound on solution quality. Therefore, we do not provide formulas for time complexity, but for the complexity w.r.t. the number of requests for a distance between nodes a and b. One fitness evaluation then corresponds to determining the length of a whole EVRP tour and is estimated as requesting a single distance between two nodes n times.

The complexity of individual initial construction methods can be seen in Table V. The complexity for constructions initiating from DBC depends on the spatial distribution of nodes in the problem and the underlying construction method, generally $k \in (10, 17)$.

Type	Methods	$\frac{evals}{evals_{max}} \times 10^7$ (min-max)	Complexity
ORE	c0	4	$i \times f$
Random	c3, c4, c7, c8	47-54	$x \times i \times f$
NN-based	c1, c2, c5, c6	199-203	i^2
MST-based	c15, c16	202-206	$i^2.log(i)$
CWS-based	c9, c10	443-446	i^2
generated from DBC	c11, c12, c13, c14	5184-6204	$k \times complexity$

TABLE V: Initial constructions evaluation demand

The complexity of individual local search operators is given in table VI. Given an EVRP tour T and an operator $\mathcal{O}$, let us define a neighborhood $\mathcal{N}_{\mathcal{O}}(T)$ as a space of EVRP tours, that are reachable by a single application of the operator $\mathcal{O}$ on tour T. The provided values then correspond to the complexity of searching through the whole neighborhood $\mathcal{N}_{\mathcal{O}}(T)$ w.r.t. to the no. of requests for a distance between two nodes.

Operator	Complexity
2-opt	n^2
2-string (all variants)	n^2
AFS reallocation	$i \times f$

TABLE VI: Local search operators - complexity

As for the metaheuristics, they generally consist of single or multiple calls of a construction method and repeated use of one or more local search operators. Therefore, their complexity is determined by the most expensive component, which, in our case, is one of the $\mathcal{O}(n^2)$ local search operators. The perturbation operator used in the VNS does not contribute to the resulting complexity, as it does not perform any fitness evaluations.

V. Conclusion

Currently, the best method available consists of the c14 (CWS-based ZGA from DBC) construction and a RVND local search with all of the implemented operators. The construction and the local search are paired in the VNS metaheuristic, which repeatedly performs the Generalized Double Bridge perturbation and the RVND local search until the stop condition is reached. However, the parameter tuning for the competition is still in progress. Various other constructions were also implemented and tested during the design. Some lower bound estimates were provided for the small competition instances. Our method surpassed all of them, which gives us hope for a good result. All team members significantly contributed to the final method. Moreover, some theoretical results were achieved as well, as it was shown that two construction methods proposed in literature do not generally work without unmentioned assumptions about the problem instances, and one of them was made more robust.

VI. Future works

As explained in Section IV-B3, the process of parameter tuning is still in progress, to make the most of the time remaining to the competition deadline. Specifically, we are trying to determine the best performing operators subset by testing all possible combinations. As we employ numerous variants of the 2-string operator, it is possible that omitting some of them will be beneficial. We intend to test the VND metaheuristic in the local search phase, as we were using only RVND so far. If the set of useful operators reduces significantly, it may be feasible to determine also their optimal order in the VND. Then, there is a parameter p, which determines the strength of the perturbation used and is fixed to $p = 4$ in all of the previous experiments. This value was not chosen arbitrarily, as it corresponds to the commonly used Double-Bridge move. However, some other value might prove more suitable. Finally, we run the VNS until the termination condition is reached. Some authors report that restarting the entire VNS within one run improves the worst-case behavior; therefore, this will be tested as well.

The competition deadline is June 5, 2020, and the results will be publicly available at [Mav20] July 19, 2020. As a paper describing the implemented method was not submitted to the conference, it will most likely be published later on, depending on the competition results. This paper will include a comparison with the other competing algorithms, as well as with previously existing state of the art methods addressing the same EVRP formulation. It will also focus on evaluating the algorithm performance from other perspectives. In the competition, the only objective is the best possible average score after a fixed number of fitness evaluations. However, the stop condition may differ depending on the application, and there are more suitable metrics to capture the algorithm behavior, such as the time-to-target plots.

INDIVIDUAL CONTRIBUTIONS

Constructions and related methods	
Relaxed ZGA	VV
Method from [Pen19] disproved	VV, DW
SSF	DW
ORE	DW
Nearest Neighbor TSP Init.	DW
Minimum Spanning Tree TSP Init.	VV
Clarke-Wright	VK
Density-based Clustering	VK
Local Search	
2-string and its variants	DW
2-opt	DW
AFS reallocation	VK
Metaheuristics	
(R)VND	DW
VNS	DW

TABLE VII: Individual contribution of team members

REFERENCES

[ECLR19] Tomislav Erdelic, Tonči Carić, and Eduardo Lalla-Ruiz, *A Survey on the Electric Vehicle Routing Problem: Variants and Solution Approaches*, 2019.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, AAAI Press, 1996, p. 226–231.

[GSS11] F Goncalves, Cardoso S., and Relvas S., *Optimization of distribution network using electric vehicles: A VRP problem*, Tech. report, University of Lisbon, 2011.

[Mav20] Michalis Mavrovouniotis, *CEC-12 Competition on Electric Vehicle Routing Problem*, https://mavrovouniotis.github.io/EVRPcompetition2020/, 2020, accessed 2020-04-07.

[MH97] N Mladenović and P Hansen, *Variable neighborhood search*, Computers & Operations Research **24** (1997), no. 11, 1097–1100.

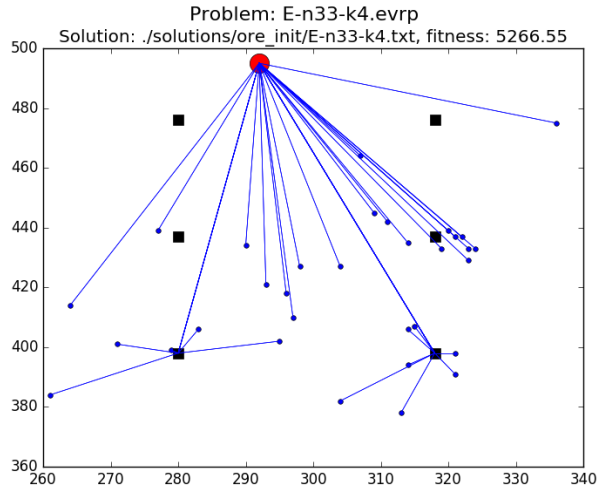
[MMT+20] Michalis Mavrovouniotis, Charalambos Menelaou, Stelios Timotheou, Christos Panayiotou, Georgios Ellinas, and Marios Polycarpou, *Benchmark Set for the IEEE WCCI-2020 Competition on Evolutionary Computation for the Electric Vehicle Routing Problem*, Tech. report, KIOS Research and Innovation Center of Excellence, Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus, 2020.

[MOF97] Olivier Martin, Steve Otto, and Edward Felten, *Large-step markov chains for the traveling salesman problem*, Complex Systems **5** (1997).

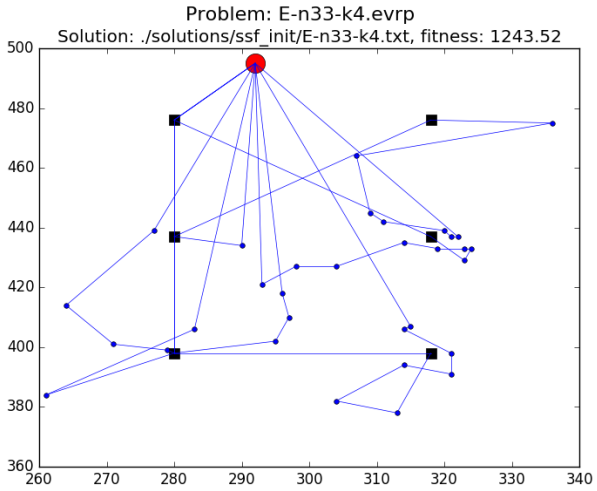
[NYBS19] Nur Mayke Eka Normasari, Vincent F. Yu, Candra Bachtiyar, and Sukoyo, *A simulated annealing heuristic for the capacitated green vehicle routing problem*, Mathematical Problems in Engineering **2019** (2019).

[Pen19] *A memetic algorithm for the green vehicle routing problem*, Sustainability (Switzerland) **11** (2019), no. 21, 516–526.

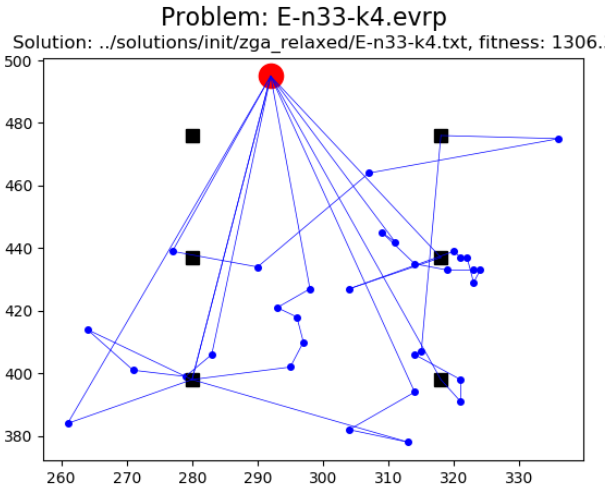
[ZGA18] Shuai Zhang, Yuvraj Gajpal, and S. S. Appadoo, *A meta-heuristic for capacitated green vehicle routing problem*, Annals of Operations Research **269** (2018), no. 1-2, 753–771.



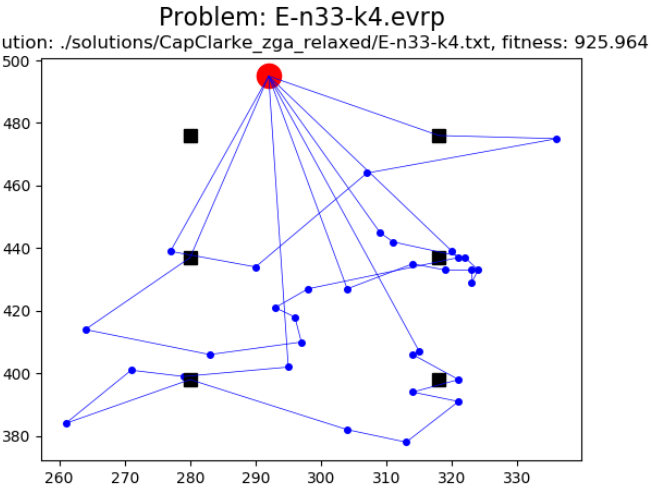
(a) One Route Each - EVRP tour



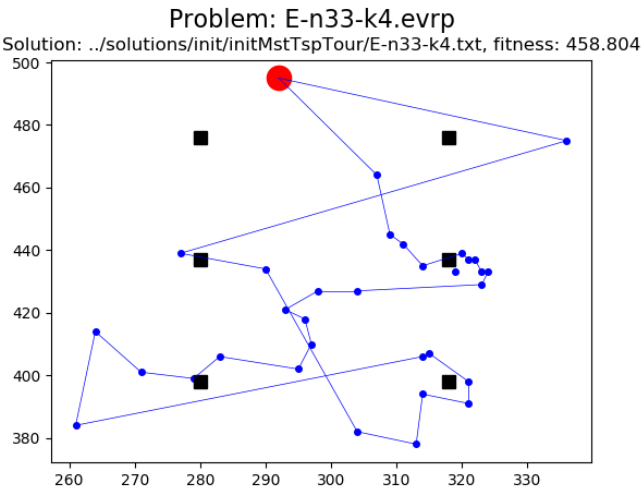
(b) Separate Sequential Fixing (NN-based) - EVRP tour



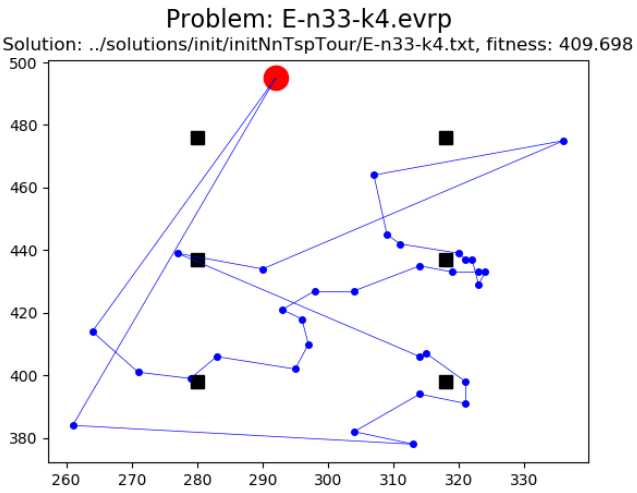
(c) ZGA relaxed (NN-based) - EVRP tour



(d) ZGA relaxed (CWS-based) - EVRP tour



(e) MST-based TSP tour



(f) NN-based TSP tour

Fig. 3: Construction methods

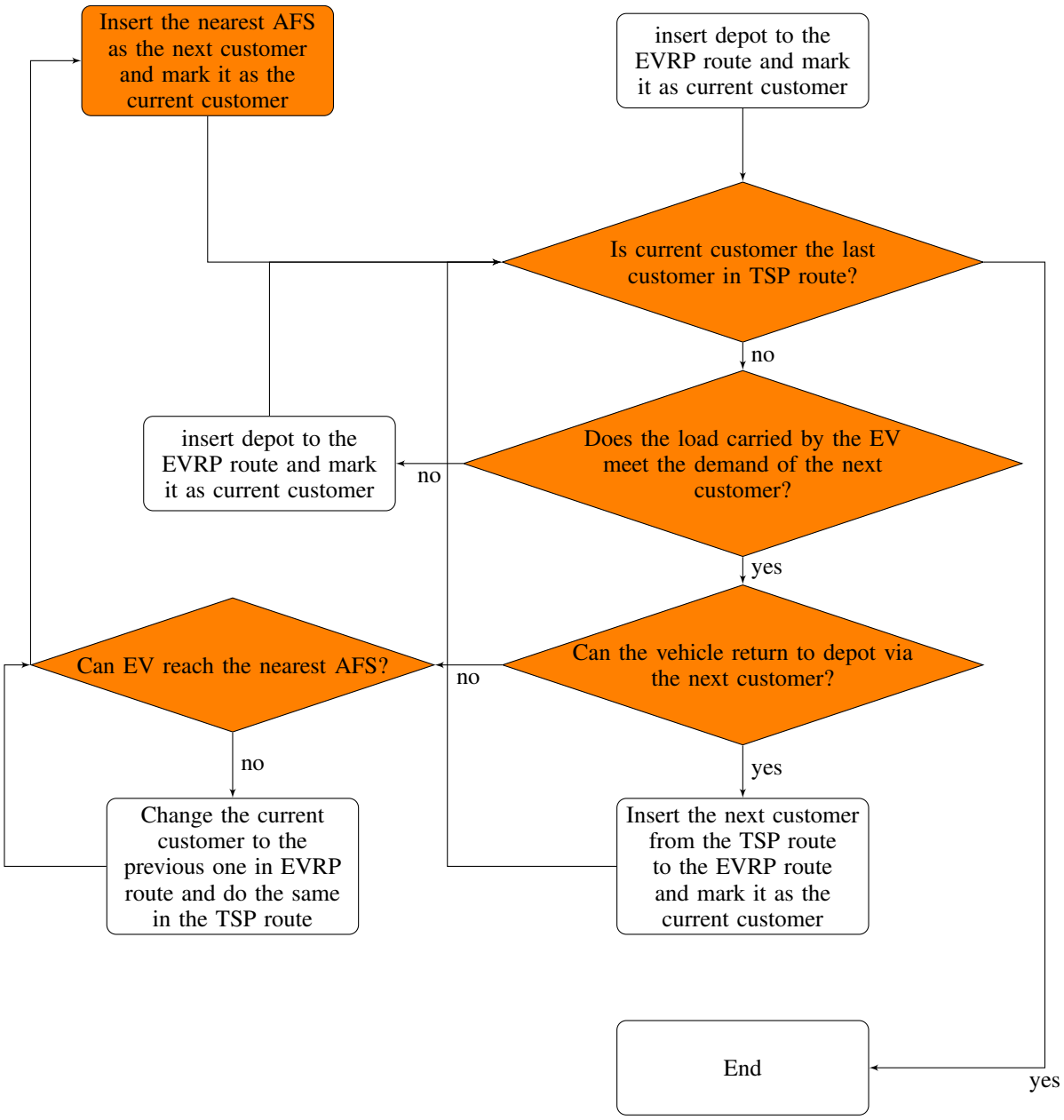


Fig. 4: Construction method proposed in [ZGA18] which can end up in an infinite cycle composed of the orange nodes

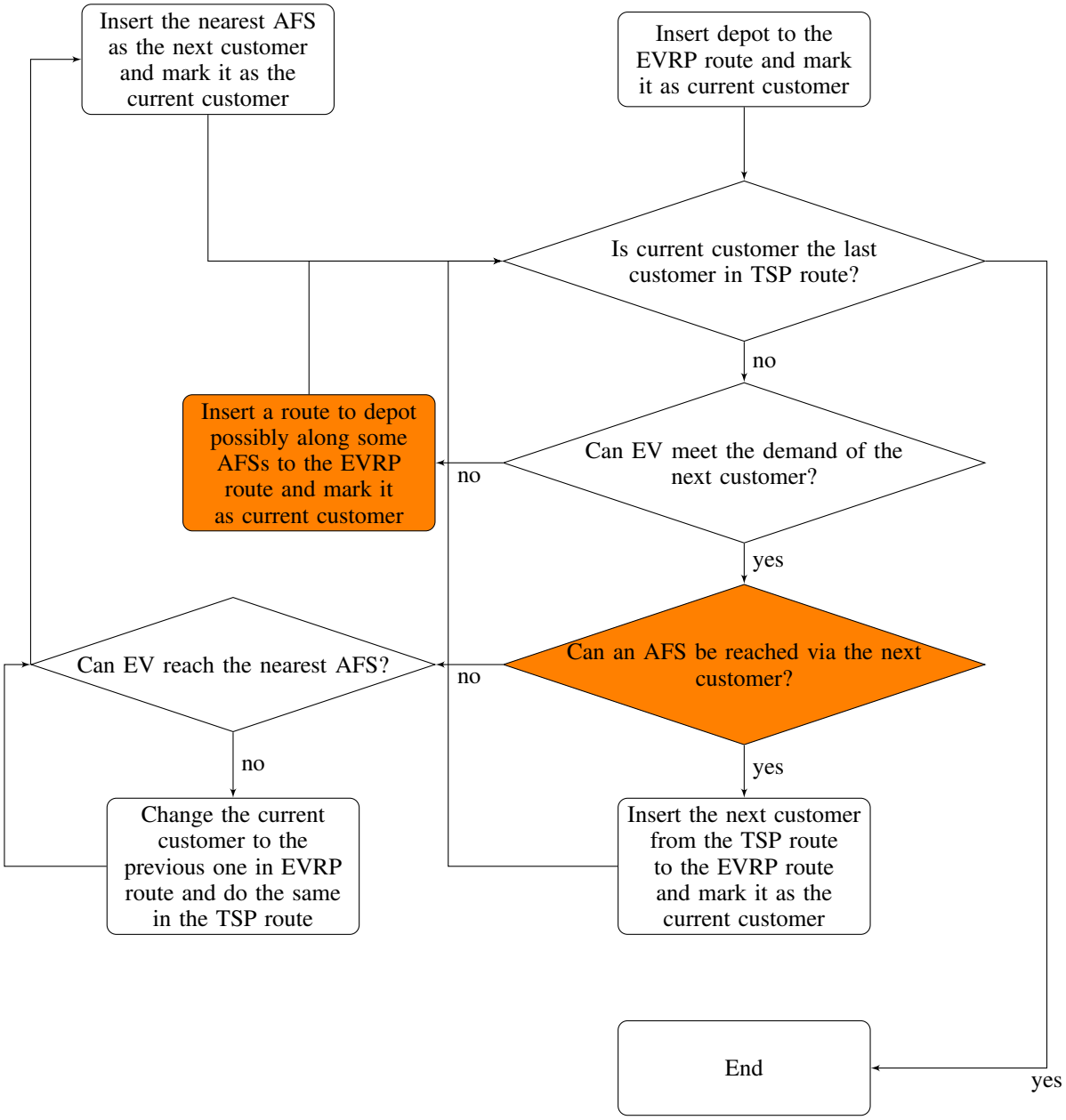


Fig. 5: Modification of the construction method proposed in [ZGA18] which doesn't end up in an infinite cycle (changes made to orange nodes)