Iterated Greedy for The Two-Echelon Electric Vehicle Routing Problem with Time Windows

Igor de Andrade Junqueira[1] , **Heder Soares Bernardino**[1] ,
Lorenza Leão Oliveira Moreno[1] , **Luciana Brugiolo Gonçalves**[1] ,
Stênio Sã Rosário F. Soares[1]

[1]Universidade Federal de Juiz de Fora (UFJF)
Rua José Lourenço Kelmer - Campus Universitário, Juiz de Fora - MG

{igor.andrade.junqueira, heder, lorenza, lbrugiolo, ssoares}@ice.ufjf.br

Abstract. *Cargo transportation can be very challenging in big cities due to congestion, pollution, and the prohibition of circulation at certain times of the day. Many transportation models can help to improve these issues. An example is the Two-Echelon Electric Vehicle Routing Problem with Time Windows (2E-EVRP-TW), which combines Electric Vehicles (EVs) with small capacity and Combustion Vehicles (CVs) with large capacity and range. These types of vehicles are combined in the solution using satellites, which are intermediary depots. The satellites attend to customers using EVs, and the satellites are attended by CVs that leave the central depot. As EVs have a limited range due to their batteries, recharging stations can be used. We propose here an Iterated Greedy (IG) with destroy and repair operators only. We compare the proposed IG with two Variable Neighborhood Search approaches from the literature, and the proposal obtained better results, both concerning distance (objective function) and CPU time.*

1. Introduction

The greenhouse effect is a natural process that comprises the heat of the planet by trapping some of the radiation from the sun and making the planet Earth habitable. The absorbed heat is made by the oceans and by atmospheric gases (greenhouse gases: methane, CO_2, and water). Most of the CO_2 in the atmosphere is generated by humans, and since the Industrial Revolution, the CO_2 levels have increased. According to [Romm 2022], since 1950, the emissions have been six times higher. One immediate result is the increase in temperature, as since 1900 the Earth has warmed 0.85ºC. Another result is the rise of the global average sea level. The main contributions to sea level rise are the thermal expansion of the ocean and ice loss in Glacier, Greenland, and Antarctica. There is amplifying feedback on the warming as the planet warms with more CO_2 in the atmosphere, which increases sea level due to ice loss. Another consequence of climate change is that it makes extreme weather events even more extreme and frequent. Those events can be heat waves, droughts, flooding, and hurricanes.

Also according to [Romm 2022], as most of the carbon in the atmosphere comes from burning fossil fuels and transportation handles 25% of the energy-related emissions, it is important to reduce the production of CO_2 in the transportation sector. One possibility for that is the adoption of Electric Vehicles (EVs) as CO_2 is not generated directly when this type of vehicle uses renewable energy (or nuclear power). Currently, some drawbacks include the limited range of the vehicle and the long charging time. Considering the potential importance of EVs in reducing CO_2, this work approaches EVs in the context of planning routes.

The application of the classic Vehicle Routing Problem (VRP) model has some limitations, such as the weight of vehicles and parking restrictions in large cities, as well as the high cost of heavy-duty trucks, according to [Sluijk et al. 2022]. In order to address these issues, the Two-Echelon approach involves splitting the distribution of cargo into two echelons.

Due to their large capacity and long operational range, trucks are commonly used to move large cargo outside of cities on long trips. Electric vehicles have a small capacity and are limited in range by battery capacity. These vehicles can be used to move small cargo inside cities on quick trips. In 2E-EVRP-TW [Akbay et al. 2022], trucks transport goods from the depot to the satellites (intermediary depots), and then electric vehicles attend customer demands, considering both delivery requirements and time window constraints. The satellites enables an improvement of logistics operations by allowing an efficient combination of both small and large-capacity vehicles.

In this paper we propose a heuristic IG (Iterated Greedy) with a RVND (Random Variable Neighborhood Descent) local search for solving the 2E-EVRP-TW. Computational experiments are performed using instances from [Akbay et al. 2022] and results indicate that the proposed IG obtained better results for large instances when compared to those from the literature.

This work is organized as follows: in Section 2 the problem is formally defined, in Section 3 a review of the literature is presented, in Section 4 the solution proposed is described, in Section 5 the results of the IG are compared to the literature algorithm, and Section 6 concludes the work.

2. Problem Description

The Two-Echelon Electric Vehicle Routing Problem with Time Windows (2E-EVRP-TW) is defined in [Akbay et al. 2022] with a graph G (V, A), where V is the vertex set and A is the set of arcs. The set V = $\{V_c \cup V_s \cup V_{rs} \cup V_d\}$ where V_c is the set of customers, V_s is the set of satellites, V_{rs} is the set of recharging stations (RS), and V_d is the set of depots (this work only considers one depot, the set of depots is defined as in [Akbay et al. 2022]). Each customer $i \in V_c$ has a positive demand d_i, a service time s_i and a time window $[twe_i, twl_i]$. The set of arcs A = $A_1 \cup A_2$, where $A_1 = \{(i,j) | i \neq j \; and \; i,j \in V_d \cup V_s\}$ are the arcs between depot and satellites and $A_2 = \{(k,r) | k \neq r \; and \; k,r \in V_s \cup V_c \cup V_{rs}\}$ are the arcs between satellites, customers, and recharging stations. A distance $dist_a$ is assigned to each arc $a \in A$.

There are two types of vehicles: Electric Vehicle (EV) and Combustion Vehicle (CV), one for each echelon. For the first echelon (satellites routing), heavy-duty trucks are used, and for the second echelon (customers routing), electric vehicles are used. Each vehicle has a payload capacity, Cap_{EV} and Cap_{CV} according to its type. The consumption of the batteries for an EV is calculated with an equation: $tx_{bat} \times dist_a$, where tx_{bat} is the consuming rate of the battery. The travel time in the arc a is proportional to the $dist_a \times C_{dist}$, where C_{dist} is a constant. To work around the range limit of the EV, recharging stations can be used to recharge the battery. The time consumed to make a full recharge is calculated as: $(Ch_{full} - ch_{rs}) \times Ch_{tx}$, where Ch_{full} is the total charging of the electric vehicle, ch_{rs} is the remaining charging in the vehicle when it arrives in the RS and Ch_{tx} is the charging rate of the electric vehicle.

Each customer must be attended by a single EV, every EV has to depart and arrive in the same satellite and each satellite can be attended by one or more CVs (this is known as Split Delivery). After connecting every customer with one satellite, it is possible to compute the sum of the demands, that is, the demand of the satellite. For the routing of the CVs to be correct, it is necessary to visit each satellite with positive demand, and the arrival time of the last CV must be smaller than or equal to twl_s (implicit late time window by attending to the customers). The EVs can only depart after all the demands arrive on their corresponding satellite. Satellites are allowed to not have customers associated with them. The CVs do not visit those satellites.

Figure 1 shows a solution for an instance (C104_21x) of the problem. The dots are the customers, triangles are the RS, diamonds are the satellites, and the square is the depot. In Figure 1, the CV leaves the depot with the total cargo and attends the satellites with positive demand. Each customer is attended to concerning its time window by EV. The EV only leaves the satellite after the CV arrives.

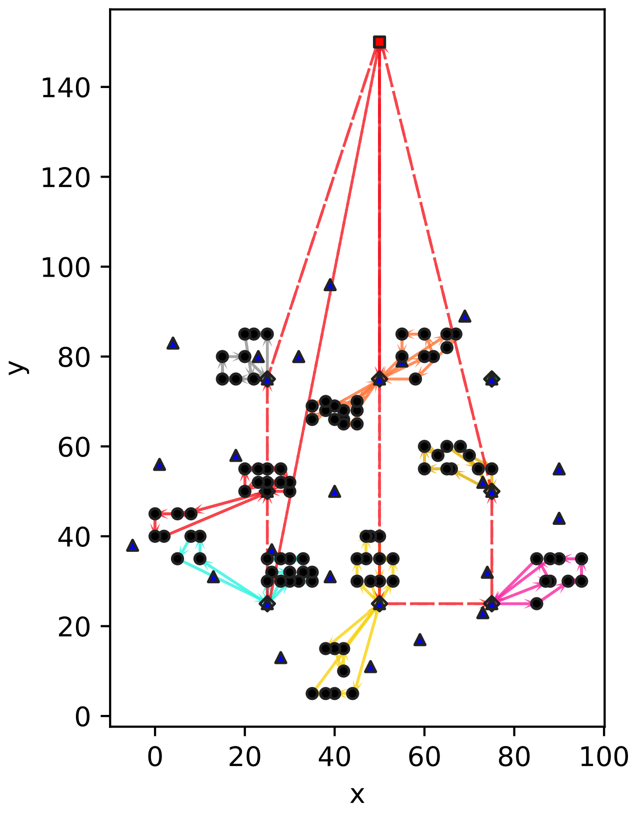

Figure 1. Example of a solution for the instance C104_21.

3. Related Work

The related work presented here is composed of those that use the Two-Echelon Electric Vehicle Routing Problem as the base problem. Next, the reference approach is described, and other works are described in a general form.

The solution proposed by [Akbay et al. 2022] is composed of two VNS (Variable Neighborhood Search) approaches, VNS_{red} and VNS_{full}. They use an extended objective function with distance and penalty to deal with infeasible solution. The infeasibility factors considered are capacity, battery, and TW. There is one weight for each penalty, and they are dynamic choices in each iteration of the VNS. The Clarke and Wright heuristic produces the initial solution. The VNS receives the start solution and uses shaking operators for escaping local optima and a VND (Variable Neighborhood Descent) for a local search. There is an acceptance criterion based on temperature, similar to Simulated Annealing. The shaking operators are divided into standard (Random cyclic exchange and Random sequence relocation), Removal/Destroy Operators (Random customer removal, Random route removal, and Close satellite), and Repair Operators (Greedy customer insertion, Greedy customer insertion with noise, Best customer insertion, and Greedy CS insertion). The VNS_{red} uses the standard shaking operators and the VNS_{full} uses all the shaking operators. The stop condition of the VNS is time. Despite the time limit, VNS most likely has a second stop condition (maximum number of iterations or maximum number of iterations without improvement) because the time of the results is not equal to the limit.

A problem for the repair operators is the lack of diversification in customer insertion. As the greedy customer insertion uses last-in-first-out (LIFO) for the customer order, they inserted each customer into the position with the lowest insertion cost (distance). Another problem occurs when there is an infeasible insertion. Since the customer is inserted in the best position, the infeasibility is ignored, and the extended objective function is not applied in this scenario.

In the literature, there are works similar to [Akbay et al. 2022], such as [Breunig et al. 2019, Jie et al. 2019, Wang and Zhou 2021]. There is a difference in the recharge process of the electric vehicle. In those works [Wang and Zhou 2021] and [Jie et al. 2019], they consider battery swapping stations to extend the vehicle range, as opposed to [Breunig et al. 2019, Akbay et al. 2022], which considers recharge stations. Another key difference is the use of an electric vehicle in the first and second echelon of this work [Jie et al. 2019], while the other works use the second echelon (customer routing). Only [Akbay et al. 2022] have time windows. For the object function, [Akbay et al. 2022] uses the travel distance, and [Breunig et al. 2019, Jie et al. 2019, Wang and Zhou 2021] uses the total cost, such as transportation, utilization of vehicles, battery swapping, and utilization of satellites.

The proposed approaches are single-solution based, the meta-heuristics used are VNS in [Akbay et al. 2022, Wang and Zhou 2021] and LNS/ALNS in [Breunig et al. 2019, Jie et al. 2019]. One characteristic used by all works is the penalty of violation solutions. In [Breunig et al. 2019], only the battery violation is considered, unlike [Akbay et al. 2022, Jie et al. 2019, Wang and Zhou 2021], which considered the penalty in a more general form. The works [Akbay et al. 2022, Wang and Zhou 2021] use an adaptive schema for the weights of penalties. Regarding the start solution, [Akbay et al. 2022, Jie et al. 2019] uses Clarke and Wright's saving heuristic, [Wang and Zhou 2021] uses the angle betting of the satellites and customers, and [Breunig et al. 2019] uses a random initial solution. The meta-heuristic Large Neighborhood Search (LNS) consists of a set of destructive procedures that destroy parts of the solution and a set of constructive procedures that try to rebuild the solution. Each procedure of destruction and constructive is chosen with equal probability in the LNS. For the ALNS (Adaptive LNS), the selection probability is determined by the previous iterations.

Although [Akbay et al. 2022] uses VNS, the shaking operators contain the same destruction and repair procedures as the LNS. For the shaking operators in VNS, [Wang and Zhou 2021] uses the operators: Random cyclic exchange and Random sequence relocation, and [Breunig et al. 2019] uses: 2-Opt*, inter-relocate, inter-exchange, inter-swap, and inter-shift, satellite-change. In the destruction phase, the most common operators used by [Akbay et al. 2022, Breunig et al. 2019, Jie et al. 2019] are: customer removal and route removal. This work also uses other operators in this phase. For the repair phase, all works [Akbay et al. 2022, Breunig et al. 2019, Jie et al. 2019] use some sort of greedy customer insertion. This generates a lack of diversification because only one customer is considered (generally in a FIFO order) and the customer is placed in the cheapest position.

For local search, [Akbay et al. 2022, Wang and Zhou 2021] uses a VND (Variable Neighborhood Descend), and [Breunig et al. 2019] uses the random version of the VND. [Jie et al. 2019] does not use a local search. Some common neighbors in [Akbay et al. 2022, Wang and Zhou 2021, Breunig et al. 2019] are: shift, swap, 2-opt, station reinsertion. The insertion of the RS [Wang and Zhou 2021] by a heuristic, and in [Jie et al. 2019] for battery swapping stations, a label optimization algorithm is used for the optimal insertion of the stations.

The works that are being presented share a common problem, but it is impossible to compare them because each one defines the problem differently (with small changes) and deals with unique instances. The time windows are a good example. The tightness of the time windows can have a huge effect on the viability of the solution. For solving the 2E-EVRP, the most commonly used technique in the literature is the destroy and repair approach. One characteristic of these methods is the use of various operators. The hypothesis behind the high number of operators is that there is a lack of variability in the repair phase, which uses a greedy algorithm to select only one customer per iteration and select the best insertion.

4. Proposed Approach

According to [Marte et al. 2018], the meta-heuristic Iterated Greedy (IG) is single-solution based, for escaping local optimal solutions, the strategies of destruction and repair are used. The algorithm begins with a stated solution in a local optimum, and then a destroy operation is made, removing parts of the solution. For repair, a greedy algorithm fixes the solution. If the new solution is feasible, the next step is to conduct a local search. Otherwise, the solution is discarded, and the best solution copes with the current one.

The local search used in the IG is the Random Variable Neighborhood Descent (RVND), from [Souza et al. 2010, Subramanian et al. 2010], which combines several neighborhoods to improve a solution. The sequence of neighborhoods is chosen randomly at the beginning of the algorithm. If the current neighborhood cannot improve the solution, it changes to the next. If an improved solution is found in the search, the algorithm goes back to the first one. The RVND finishes when all neighborhoods fail to find an improved solution.

The proposed algorithm begins by generating a feasible initial solution to the 2E-EVRP-TW and subsequently applies an RVND to find a local optimal one. To escape local optima, it destroys the solution by removing either customers, EVs, or satellites. The following repair phase fixes it with a greedy approach based on cheaper insertions. If the resulting solution is feasible, the RVND is used again as a local search; otherwise, the algorithm continues with the last local optimal solution.

Another characteristic of the IG is the acceptance criteria. A solution is acceptable if it is feasible and the delta percentage between it and the best solution is less than a threshold. The stopping condition of the IG is the number of iterations that are controlled by the parameter *numItMaxIG*.

Algorithm 1 summarizes the core aspects of the proposed IG for the 2E-EVRP-TW. The algorithm receives several input parameters, including: instance (*Inst*), vector of alphas (*vetApha*), vector of betas (*vetBeta*), the maximum number of iterations for starting the solution (*numItMaxGreedy*), number of iterations for update probability (*numItUpdateProb*), alpha of IG (*igAlpha*), beta of IG (*igBeta*), percentage difference of best (*difBest*), removal rate (*rmRate*) and selection strategy (*select*).

In line 1, *SolBest* is started through a function that uses a greedy heuristic that has several strategies to produce a feasible solution, described in Section 4.1. Lines 5 to 7 start variables with *numRm* (number of elements that are removed from the solution), *limitCall* (number max of iterations for the function *destroyEV* (Section 4.2)), *numCall* (current number of calls to *destroyEv*), and *SolC* (current solution). Lines 11 and 14 destroy *SolC* and produce Sol_P. Line 16 Sol_P, the function *repairSol* (Section 4.3), tries to generate a feasible solution and produce *Sol'*. If *Sol'* is feasible, in line 18, a local search is applied. In line 22, *acceptanceCriteria* decide if *Sol'* goes for the next iteration. In Line 23, the best solution returns.

The following sections present the components of the proposed IG in details.

4.1. Start Solution

The start solution (*startSolution*) receives a list of alphas (*vetApha*), a list of betas (*vetBeta*), the maximum number of iterations (*numItMaxGreedy*), and the number for updating the probability (*numItUpdateProb*). If a feasible solution is found, the algorithm immediately returns. The algorithm consisted of choosing an α and β and calling two inserting heuristics, one for each echelon. The α and β are randomly chosen from their respective lists. In the first *numItUpdateProb* iterations, each α and β have the same probability. Each not feasible solution has the objective function accumulated. This objective function has the partial distance plus a penalty (proportional to the number of customers that are not visited). After every *numItUpdateProb* iterations, the probability is recalculated using the accumulated objective function.

The insert heuristic, for the second echelon (customer side), receives a list of customers and generates a list of candidates (sorted by distance increment). For each

Algorithm 1: Iterated Greedy for 2E-EVRP-TW.

```
Input: Inst, numItIG, vetAlpha, vetBeta, numItMaxGreedy,
       numItUpdateProb, igAlpha, igBeta, difBest, rmRate, select,
       multLimitCall
Output: Solution
1  SolBest ← startSolution(vetAlpha, vetBeta, numItMaxGreedy,
     numItUpdateProb);
2  if !feasible(SolBest) then
3      return EmptySol;
4  numEvUsed ← getNumVehicEvUsed(SolBest);
5  numRm ← rmRate × numEvUsed;
6  limitCall ← multLimitCall × getBaseLimitCall(numRm, numEvUsed);
7  numCall ← 0;
8  SolC ← copy(SolBest);
9  for i ← 0, ..., numItMaxIG-1 do
10     if numCall < limitCall then
11         Sol_p ← destroyEVs(SolC, numRm);
12         numCall++;
13     else
14         Sol_p ← destroySat(SolC);
15         numCall ← 0;
16     Sol' ← repairSol(Sol_p, igApha, igBeta, select);
17     if feasible(Sol') then
18         Sol' ← rvnd(Sol');
19         if cost(Sol') < cost(SolBest) then
20             SolBest ← Sol';
21             numFuncDestroy ← 0;
22     SolC ← acceptanceCriteria(difBest, Sol', SolC, SolBest);
23 return SolBest;
```

customer, there is only one candidate, the best insertion, considering all EV routes on all satellites. In each iteration, one candidate is randomly picked from a restricted list of candidates. This list is created from a portion of the candidate list (using α). After picking one candidate and inserting it into the solution, the list updates, and this process continues until all customers are in the solution (feasible) or the list is empty (infeasible). After having a feasible solution for the customers, the first echelon can be solved. Using the information from the EV routes and the sum of the demands in each satellite, the first echelon is solved similarly to the second one, using the β parameter.

Since the heuristics can generate infeasible solutions, there are three methods for producing a feasible solution. The first one is the chosen of the α and β (explained above). The second approach begins if the previous one fails. The process involves establishing the probability that each customer will be included in the solution. With this likelihood, one or more customers are picked to begin a route specifically for them. The technique that corrects a single EV route (explained in Section 4.5) does not address the dedicated route's potential for having two consecutive RSs, so this is necessary. Since this cannot work because of the maximum number of EVs, the third strategy is to start one EV route with a random pick RS. This can work because it enables two consecutive RSs.

4.2. Destroy

To destroy the solution, there are three methods: remove EVs (*destroyEVs*), customers, and satellites (*destroySat*). In all cases, one element is randomly chosen and then removed from the solution. The removal of EVs and satellites used together in large instances and the removal of customers used in small instances. This approach is necessary, because removing one EV in a small instance can destroy all components of the solution.

The removal of EVs is used for *limitCall* iterations, and it is calculated by the function (*getBaseLimitCall*). This function calculated the minimum number of calls needed to remove all EVs from the solution. Since there are many possible combinations, it can be necessary to have more calls to remove EV. Because of that, the parameter (*multLimitCall* = [1,3]) multiplies the base value.

After using the remove EVs for *limitCall* iterations without improvement, one satellite is removed by the function *destroyEVs*, and in the next iteration, removing EVs continues. The number of EVs that are deleted (*numRm*) is a percentage (*rmRate*) of the EVs used in the second solution. The number of removed customers is similar, and only one satellite can be removed.

4.3. Repair

For fixing the current solution, the IG uses a greedy algorithm (*repairSol*) to rebuild the destroyed solution. Similar to the start solution, the repair phase uses the cheaper insert heuristic, the difference is that one customer has various candidates. One customer has one candidate for each EV route (the best insertion) because it is necessary to generate different solutions. We use this for large instances. For small ones, the customer has various candidates for each EV route, one candidate for each route and position. Another change from the starting solution is the selection strategy (random or ternary tournament). The selection strategy (*select*) is applied to the restricted candidate list. The second echelon is solved in the same way as the start solution. The randomness of the algorithm is controlled by two parameters (*igAlpha* and *igBeta*) received from the IG. After the heuristic, the solution should be feasible. Otherwise, the solution before the destruction phase is restored.

4.4. Local Search

We use the RVND only in the second echelon, and the first one is made again when there is an improved solution in the second one. The neighborhood used are divided into: inter satellites, intra satellite inter routes, and intra satellite intra route. Inter satellites neighborhoods select two different satellites and one route from each. The possible neighborhoods are: shift, swap, cross, and 2-shift. Intra satellite inter routes select two routes from the same satellite and use the same neighborhoods as inter satellites. Intra satellite intra route select only one route from one satellite and use the neighborhoods: shift, swap, and 2-opt. All neighborhoods use the first improvement approach. The neighborhoods described can be checked in [Toth and Vigo 2014].

4.5. Fixing EV Routes

As EVs have a short range due to their battery capacity, it is necessary to recharge the EV to extend the distance traveled. When an infeasible route concerning the battery constraint is found, an algorithm is called in order to fix this route. This algorithm tries to insert a recharging station into the route. The first step is to locate the arc $(i, j) \in A_2$ that causes the failure. The second is to determine the start of the insertions $b \in V_s \cup V_{rs}$, which can be i or another vertex before i in the current route. The fixing algorithm tries to put an RS in every arc between b and i. Every RS of the instance is considered, and this algorithm checks if the failure was solved. This EV route is returned when a feasible solution is found. The current route is discarded otherwise. One can notice that only a single RS can be inserted. Functions *startSolution*, *repairSol* and *rvnd* use this algorithm to fix their infeasible EV routes.

5. Computational Experiments

In this section, the results from [Akbay et al. 2022], VNS$_{\text{red}}$ and VNS$_{\text{full}}$ (explain in the Section 3), will be compared with the IG for the 2E-EVRP-TW. The subsection 5.1 explains how the instances were generated by [Akbay et al. 2022], the subsection 5.2 explains how the parameter tuning was made, and the subsection 5.3 shows the results.

The IG for the 2E-EVRP-TW was coded in C++, and the tests were executed on a computer with an AMD Ryzen 7 5800X 3.8 GHz (single thread) with 16 GB of RAM memory. [Akbay et al. 2022] uses an Intel Xeon 5670 with 2.9 GHz and 32 GB of RAM memory. In single thread for the benchmark (PassMark[1]), the AMD CPU has 3447 points, and the Intel has 1392 points. The rate (0.4) between the processors is used for converting the CPU time. The source code and supplementary material are public available[2].

The paired Student t-test (significance level is 0.05) was used to compare the columns (distance best (*dist best*), distance average (*dist avg*), and CPU time (*CT*)), those columns are only in the supplementary material. Each column for all algorithms was checked using the Kolmogorov-Smirnov Test of normality. The statistical tests are in the supplementary material. The instances were executed 10 independent times, the same as the [Akbay et al. 2022] experiments.

5.1. Instances

The instances were made by [Schneider et al. 2014] for the EVPR-TW with recharge stations, and adapted by [Akbay et al. 2022] for the 2E-EVRP-TW. The set of instances comprises 92 instances, including 36 small and 56 large instances. In the small instances, there are 3 groups: C5, C10, and C15, respectively, with 5, 10, and 15 customers. For the large instances, there are 3 groups: C100, R100, and RC100, all of which have 100 customers. Table 1 shows the average number of RS, satellites, EV, and CV for each group. One big problem with the methodology used by [Akbay et al. 2022] to generate the new instances is that for every instance in one group, the location of the satellites is the same for every instance in the same group. The same instances of [Akbay et al. 2022] can be accessed in the supplementary material.

[1]`cpubenchmark.net/compare/1307vs3869/Intel-Xeon-X5670-vs-AMD-Ryzen-7-5800X`
[2]`github.com/Routing-Problems-in-Green-Logistic/ENIAC23_2E_EVRP_TW`

Table 1. Instances AVG characteristics.

Group	RS	Sat	EV	CV
C5	3.5	1.0	1.6	1.0
C10	4.3	1.0	3.3	1.0
C15	6.8	2.0	4.3	1.0
C100	29.0	8.0	26.3	2.5
R100	29.0	8.0	19.0	1.8
RC100	29.0	8.0	21.5	2.1

Table 2. Parameters List.

Parameter	Values
igAlpha	(0.005, 0.01, 0.05, 0.1, 0.15, 0.25, 0.35, 0.45, 0.65, 0.9)
igBeta	(0.15, 0.25, 0.4, 0.5, 0.65, 0.75, 0.8, 0.85, 0.9)
difBest	(0.005, 0.01, 0.015, 0.02, 0.03, 0.04, 0.05, 0.07, 0.1)
select	(0, 1)
rmRate	(0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6)
multLimitCall	(1, 1.5, 2, 2.5, 3)

5.2. Parameter Tuning

The parameter tuning was made by the the Irace ([López-Ibáñez et al. 2016]), the parameters list are in the Table 2. The Irace was run into two groups, the first (*Small*) composed of instances of 5, 10, and 15 customers, and the second (*Large*) of instances of 100 customers. This division is necessary as IG has two ways to optimize small and large instances. [Akbay et al. 2022] also makes this division into the tuning process. For the small group, 6 instances (16.6%) were chosen, and the Irace was set to 1500 maximum runs. For the large one, 13 instances (23%) were chosen, and the Irace was set to 4000 maximum runs. The Table 3 shows the parameters chosen by the Irace for the two groups of instances. The parameters *igBeta* and *multLimitCall* are not used for small instances since there are few satellites (the *igBeta* is set to 1.0) and there is only one destroy method.

There are some parameters that are not chosen by the Irace. The parameters are: (*numItIG* = 3000), (*vetAlpha*, *vetBeta* = (0.05, 0.07, 0.1, 0.15, 0.2, 0.3, 0.35, 0.4, 0.5, 0.6)), (*numItMaxGreedy* = 3000) and (*numItUpdateProb* = 250). The instances used in the tuning process can be checked in the supplementary material.

Table 3. Parameters used.

Parameter	Small	Large
igAlpha	0.9	0.05
igBeta	-	0.8
difBest	0.05	0.015
select	0	0
rmRate	0.5	0.15
multLimitCall	-	2

5.3. Results

Tables 4, 5, 6 show the results obtained by VNS_{red} and VNS_{full}, from [Akbay et al. 2022], and by the proposed IG. There are 3 columns for each algorithm: Gap-Best, Gap-Avg, and CPU time in seconds. For IG, CT represents the CPU time, and for both VNS approaches, the CTC column refers to the CPU time converted. The distance values for each algorithm can be checked in the supplementary material, as the tables only have the Best* (min) of the best results from 10 runs from the VNS and from the IG. The Gap-Best is the percentage difference between the distance of the best run and the Best*. The Gap-Avg is the percentage difference between the avg distance of 10 runs and the Best*. For each instance and for the Avg case, the best values for each performance metric are highlighted in boldface. The results are presented using a single decimal place, but the best values are selected considered two decimal places.

Table 4 shows the results for small instances. The Gap-Best and Gap-Avg averages of the proposed IG are competitive to both VNS for the groups C5 and C10. The Gap-Best for the proposed IG is equal to 0.0%, as the same as the VNS. The proposed IG has the average of Gap-Avg for the C5 group equal to 1.2% when the VNS_{full} has 0.1%. The difference between the methods is 1.1% which is the large difference. For the C15 group, the proposed IG is competitive which the VNS_{full}, with a maximum difference, in terms of Gap-Avg, equal to 0.3%. The proposed IG is better than the VNS_{red} for this group, with a difference equal to 3.6% in Gap-Avg. Concerning the CPU time, VNS_{red} obtained the best values for the C5 group while IG reached the best values for the C10 and C15 groups.

Table 5 shows the results for large instances of the groups C100 and R100. For the C100 group, the Gap-Best of the proposed Ig, and the VNS_{full} has a close result, with a 0.2% difference. For Gap-Avg and CPU time, the proposed IG has better results, with a 1% difference and 8 times faster. For the VNS_{red}, the proposed IG has better results, with 2.3% and 2.35% differences in Gap-Best and Gap-Avg. The CPU time is 8 times faster. For the R100 instances, the IG has a big improvement compared with the VNS. The proposed IG has an 0% Gap-Best, while VNS_{full} has 6.8% and VNS_{red} 7.0%. For Gap-Avg IG has 1.1%, while VNS_{full} has 12% and VNS_{red} 11%. For CPU time, the proposed IG 8 times faster than VNS_{full} and 9 times faster than VNS_{red}.

Table 6 shows the results for large instances of the group RC100. The IG has an improvement compared with the VNS, IG has an 0.3% Gap-Best, while VNS_{full} has 1.4% and VNS_{red} 3.3%. For Gap-Avg, the IG has 1.8%, while VNS_{full} has 6.0% and VNS_{red} 6.8%. For the CPU time, the IG is faster than VNS_{red} and VNS_{full}, because it has a speedup of 10 times.

The values of the statistical tests concerning the results presented in Tables 4, 5, and 6 are in the supplementary material. The results for the C5 and C10 reveal a non statistically significant difference between the proposed IG and both VNS for the averages of dist-avg and dist-best. In the group C15, the proposed IG has a statistically significant difference for the averages dist-best and dist-avg when compared with VNS_{red}. For the VNS_{full}, the result is non significant. For the groups C100, R100, and RC100, the tests show there is a statistically significant difference between the proposed IG and both VNS for the averages of dist-avg, dist-best, and CPU time. The exception is for the C100 group in dist-Best for VNS_{full}, where there is a non statistical significance difference.

Table 4. Results for the small instances.

Instance	Best*	VNS$_{\text{red}}$ [a]			VNS$_{\text{full}}$ [a]			IG		
		Gap Best	Gap Avg	CTC	Gap Best	Gap Avg	CTC	Gap Best	Gap Avg	CT
C101_C5x	385.49	**0.00%**	**0.00%**	0.40	**0.00%**	**0.00%**	5.00	**0.00%**	**0.00%**	**0.22**
C103_C5x	341.33	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	0.20	**0.00%**	**0.00%**	0.21
C206_C5x	417.31	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	0.26
C208_C5x	381.91	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	0.29
R104_C5x	317.02	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	0.28
R105_C5x	453.74	**0.00%**	9.13%	**0.00**	**0.00%**	**0.00%**	11.88	**0.00%**	**0.00%**	0.35
R202_C5x	347.82	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	**0.00**	**0.00%**	0.04%	0.26
R203_C5x	371.31	4.09%	4.09%	**0.00**	**0.00%**	**0.00%**	2.88	**0.00%**	**0.00%**	0.25
RC105_C5x	432.64	**0.00%**	0.72%	**0.20**	**0.00%**	1.09%	8.60	**0.00%**	**0.00%**	0.21
RC108_C5x	460.89	**0.00%**	4.09%	**0.00**	**0.00%**	**0.00%**	1.31	**0.00%**	11.12%	0.22
RC204_C5x	332.86	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	0.01	0.61%	0.61%	0.22
RC208_C5x	327.30	1.37%	1.37%	**0.00**	**0.00%**	**0.00%**	6.08	**0.00%**	**0.00%**	0.34
Avg	380.80	0.43%	1.41%	**0.05**	**0.00%**	**0.10%**	3.00	0.04%	1.17%	0.26
C101_C10x	538.31	**0.00%**	0.08%	**0.20**	**0.00%**	**0.00%**	5.29	0.26%	0.26%	0.87
C104_C10x	484.32	**0.00%**	**0.00%**	**0.40**	**0.00%**	**0.00%**	2.85	**0.00%**	**0.00%**	0.88
C202_C10x	425.53	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	0.81	**0.00%**	1.44%	0.74
C205_C10x	415.48	**0.00%**	1.00%	**0.00**	**0.00%**	**0.00%**	0.40	**0.00%**	0.14%	0.80
R102_C10x	505.50	**0.00%**	**0.00%**	1.00	**0.00%**	3.78%	9.02	**0.00%**	**0.00%**	**0.73**
R103_C10x	436.08	**0.00%**	**0.00%**	1.00	**0.00%**	0.33%	1.44	**0.00%**	**0.00%**	**0.68**
R201_C10x	460.71	**0.00%**	**0.00%**	1.10	**0.00%**	**0.00%**	6.58	**0.00%**	1.47%	**1.05**
R203_C10x	436.51	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	**0.00**	0.06%	2.17%	1.28
RC102_C10x	618.75	**0.00%**	**0.00%**	16.60	**0.00%**	**0.00%**	4.41	**0.00%**	0.96%	**0.74**
RC108_C10x	559.88	**0.00%**	**0.00%**	**0.00**	**0.00%**	**0.00%**	0.01	**0.00%**	0.01%	0.59
RC201_C10x	495.54	**0.00%**	0.30%	**0.00**	**0.00%**	**0.00%**	1.01	0.07%	0.07%	1.35
RC205_C10x	576.17	**0.00%**	0.28%	**0.10**	**0.00%**	**0.00%**	6.15	**0.00%**	0.11%	1.19
Avg	496.06	**0.00%**	**0.13%**	1.70	**0.00%**	0.35%	3.16	0.03%	0.53%	**0.91**
C103_C15x	575.18	**0.00%**	1.19%	2.00	**0.00%**	**0.00%**	**0.25**	0.72%	1.39%	2.26
C106_C15x	516.60	**0.00%**	1.45%	0.80	**0.00%**	**0.00%**	**0.41**	**0.00%**	0.24%	1.91
C202_C15x	550.32	12.16%	12.42%	12.00	**0.00%**	**0.00%**	4.98	0.93%	2.14%	**1.23**
C208_C15x	550.02	12.67%	12.67%	2.80	**0.00%**	**0.00%**	8.80	**0.00%**	2.63%	**1.52**
R102_C15x	703.94	1.79%	1.79%	3.80	1.79%	1.79%	4.82	**0.00%**	**0.96%**	**0.84**
R105_C15x	607.96	**0.00%**	**0.00%**	12.30	**0.00%**	**0.00%**	10.24	**0.00%**	0.53%	**0.27**
R202_C15x	593.69	**0.00%**	0.69%	3.20	**0.00%**	**0.00%**	24.40	**0.00%**	**0.00%**	**0.58**
R209_C15x	475.10	**0.00%**	9.34%	**0.30**	**0.00%**	**1.52%**	30.95	**0.00%**	3.86%	0.77
RC103_C15x	616.32	**0.00%**	0.94%	**0.60**	**0.00%**	**0.00%**	0.72	**0.00%**	0.39%	1.07
RC108_C15x	603.87	**0.00%**	**0.00%**	**0.10**	**0.00%**	1.86%	0.13	0.90%	1.53%	1.58
RC202_C15x	552.70	8.89%	8.89%	**1.00**	**0.00%**	6.23%	4.64	0.15%	**0.15%**	1.06
RC204_C15x	485.34	13.64%	13.64%	**0.30**	**0.00%**	**0.00%**	6.23	1.25%	2.27%	0.80
Avg	569.25	3.87%	4.90%	3.26	**0.18%**	**0.96%**	8.05	0.32%	1.28%	**1.16**

[a][Akbay et al. 2022]

Table 5. Results for the large instances.

Instance	Best*	VNS$_{\text{red}}$ [a]			VNS$_{\text{full}}$ [a]			IG		
		Gap Best	Gap Avg	CTC	Gap Best	Gap Avg	CTC	Gap Best	Gap Avg	CT
C101_21x	1494.18	1.32%	4.59%	199.90	**0.00%**	2.98%	231.99	0.07%	**2.77%**	**17.40**
C102_21x	1447.86	3.72%	4.08%	215.00	**0.00%**	**2.71%**	228.81	1.62%	2.81%	**28.40**
C103_21x	1399.25	3.48%	4.58%	203.70	**0.00%**	**1.90%**	262.65	1.60%	3.23%	**35.86**
C104_21x	1400.52	2.46%	3.26%	162.10	**0.00%**	2.80%	216.39	0.53%	**2.10%**	**41.37**
C105_21x	1484.35	2.60%	3.86%	143.60	0.63%	2.48%	186.45	**0.00%**	**1.79%**	**44.00**
C106_21x	1429.75	3.15%	4.33%	144.50	**0.00%**	3.29%	214.77	1.63%	**2.77%**	**41.26**
C107_21x	1476.88	1.55%	2.47%	160.30	0.60%	2.46%	232.98	**0.00%**	**1.10%**	**42.62**
C108_21x	1417.00	3.12%	4.21%	193.20	2.40%	5.13%	209.55	**0.00%**	**1.80%**	**38.48**
C109_21x	1409.97	2.65%	3.33%	130.60	**0.00%**	3.26%	269.50	0.68%	**1.97%**	**39.00**
C201_21x	1204.16	3.94%	6.00%	169.10	0.38%	2.47%	218.01	**0.00%**	**1.29%**	**18.35**
C202_21x	1187.87	3.43%	6.08%	213.10	**0.00%**	3.78%	281.23	0.17%	**1.55%**	**21.77**
C203_21x	1168.41	2.49%	4.69%	142.50	2.82%	4.10%	307.03	**0.00%**	**1.90%**	**22.10**
C204_21x	1137.03	3.62%	4.83%	185.70	2.11%	3.90%	231.18	**0.00%**	**2.44%**	**22.97**
C205_21x	1197.95	2.38%	4.31%	184.10	0.61%	2.17%	265.76	**0.00%**	**0.75%**	**24.66**
C206_21x	1182.63	1.68%	3.41%	207.90	**0.00%**	1.35%	222.41	0.04%	**0.77%**	**25.02**
C207_21x	1168.39	2.29%	3.69%	105.10	0.45%	1.76%	225.15	**0.00%**	**1.15%**	**24.24**
C208_21x	1164.39	2.46%	4.90%	171.70	0.45%	2.10%	261.19	**0.00%**	**1.08%**	**21.76**
Avg	1315.92	2.71%	4.23%	172.48	0.59%	2.87%	239.12	**0.40%**	**1.88%**	**29.96**
R101_21x	1707.79	26.73%	28.08%	229.56	27.64%	35.08%	235.88	**0.00%**	**0.86%**	**23.18**
R102_21x	1572.51	17.04%	20.45%	233.54	17.23%	28.79%	120.03	**0.00%**	**1.11%**	**25.84**
R103_21x	1484.57	14.27%	18.17%	262.91	16.53%	23.22%	140.26	**0.00%**	**2.37%**	**17.01**
R104_21x	1412.26	4.34%	16.23%	308.16	4.10%	15.28%	214.12	**0.00%**	**1.45%**	**14.87**
R105_21x	1583.95	16.31%	19.88%	215.76	20.53%	24.74%	185.45	**0.00%**	**1.07%**	**18.69**
R106_21x	1515.02	14.71%	23.45%	138.38	13.79%	24.58%	61.26	**0.00%**	**0.98%**	**19.62**
R107_21x	1447.34	4.95%	15.51%	114.80	2.95%	15.38%	129.40	**0.00%**	**1.11%**	**15.09**
R108_21x	1404.87	3.57%	10.58%	128.98	3.15%	11.73%	73.73	**0.00%**	**2.13%**	**10.74**
R109_21x	1483.13	4.34%	14.25%	142.58	3.14%	13.53%	154.46	**0.00%**	**1.35%**	**13.22**
R110_21x	1449.32	0.12%	2.54%	239.02	1.47%	4.43%	264.32	**0.00%**	**1.42%**	**8.14**
R111_21x	1459.41	1.95%	7.75%	231.68	4.32%	9.18%	193.29	**0.00%**	**1.69%**	**7.33**
R112_21x	1384.48	5.24%	5.24%	**0.00**	2.12%	4.93%	25.79	**0.00%**	**1.16%**	14.19
R201_21x	1199.22	3.31%	5.57%	194.56	1.64%	4.42%	255.92	**0.00%**	**0.47%**	**19.93**
R202_21x	1122.67	3.20%	4.24%	250.56	1.17%	3.92%	225.62	**0.00%**	**0.62%**	**18.38**
R203_21x	1042.00	2.13%	4.93%	205.24	2.49%	5.25%	217.03	**0.00%**	**0.89%**	**36.35**
R204_21x	961.43	0.08%	3.43%	213.63	0.44%	1.64%	236.41	**0.00%**	**0.19%**	**27.07**
R205_21x	1103.12	3.02%	5.88%	284.66	2.83%	4.72%	180.81	**0.00%**	**0.58%**	**28.09**
R206_21x	1074.75	2.93%	5.80%	315.68	1.66%	4.01%	205.12	**0.00%**	**1.07%**	**32.37**
R207_21x	1014.16	2.00%	5.72%	235.21	1.08%	4.12%	205.20	**0.00%**	**0.70%**	**30.68**
R208_21x	960.97	3.15%	5.94%	193.95	0.97%	3.67%	207.60	**0.00%**	**0.28%**	**30.64**
R209_21x	1057.06	2.03%	4.71%	239.14	1.98%	3.05%	239.07	**0.00%**	**0.71%**	**38.11**
R210_21x	1036.17	2.23%	5.25%	206.14	0.91%	3.13%	192.16	**0.00%**	**0.98%**	**35.79**
R211_21x	996.72	3.42%	5.73%	209.84	0.25%	3.76%	168.78	**0.00%**	**0.36%**	**24.89**
Avg	1281.43	6.99%	11.47%	208.43	6.76%	12.46%	179.64	**0.00%**	**1.08%**	**22.18**

[a][Akbay et al. 2022]

Table 6. Results for the large instances.

Instance	Best*	VNS_{red} [a]			VNS_{full} [a]			IG		
		Gap Best	Gap Avg	CTC	Gap Best	Gap Avg	CTC	Gap Best	Gap Avg	CT
RC101_21x	1860.94	9.89%	22.21%	117.72	2.50%	13.19%	242.10	**0.00%**	**1.77%**	**13.09**
RC102_21x	1814.25	10.50%	12.20%	206.57	1.14%	12.87%	158.85	**0.00%**	**1.47%**	**11.21**
RC103_21x	1726.56	1.24%	11.99%	157.29	0.09%	6.93%	188.09	**0.00%**	**2.40%**	**18.96**
RC104_21x	1644.36	**0.00%**	2.59%	128.80	0.06%	2.69%	149.18	0.98%	**1.95%**	**21.15**
RC105_21x	1789.64	**0.00%**	**1.78%**	188.53	0.74%	8.23%	202.71	0.79%	2.58%	**6.64**
RC106_21x	1728.89	1.81%	3.98%	233.66	1.26%	4.54%	180.24	**0.00%**	**3.71%**	**8.04**
RC107_21x	1683.97	0.23%	1.77%	197.34	0.17%	2.13%	272.48	**0.00%**	**1.55%**	**17.57**
RC108_21x	1622.76	3.08%	3.31%	176.32	**0.00%**	**2.00%**	304.28	1.95%	3.06%	**18.10**
RC201_21x	1289.28	1.84%	4.08%	272.83	2.28%	5.39%	113.04	**0.00%**	**0.36%**	**21.82**
RC202_21x	1190.98	2.30%	4.64%	276.60	0.81%	3.36%	212.75	**0.00%**	**0.05%**	**19.20**
RC203_21x	1074.41	4.21%	6.17%	235.72	2.70%	5.99%	249.91	**0.00%**	**0.90%**	**21.92**
RC204_21x	1020.51	2.47%	5.63%	185.06	1.92%	3.38%	188.06	**0.00%**	**0.26%**	**36.61**
RC205_21x	1177.37	3.91%	6.45%	147.37	3.40%	5.76%	142.73	**0.00%**	**2.46%**	**27.24**
RC206_21x	1163.64	4.56%	6.16%	198.26	2.53%	4.51%	244.32	**0.00%**	**1.63%**	**25.04**
RC207_21x	1085.42	2.85%	4.47%	213.09	1.95%	5.59%	176.81	**0.00%**	**0.94%**	**23.89**
RC208_21x	1011.68	2.63%	6.89%	214.28	3.73%	5.55%	206.72	**0.00%**	**1.61%**	**33.07**
AVG	1430.29	3.32%	6.84%	196.84	1.41%	6.04%	202.02	**0.27%**	**1.80%**	**20.22**

[a][Akbay et al. 2022]

6. Conclusions

The 2E-EVRP-TW is very important as it approaches the real world by using two echelon with EVs in one of them, and despite EV adoption increasing, it can be challenging to make goods routing. The proposed IG has one main difference from the related work due to the repair phase having multiple candidates. One advantage is that a simple approach and has fewer operators and parameters.

The computational experiments compared the proposed IG with two VNS from [Akbay et al. 2022]. After comparing the experiments, we can conclude that the IG has better performance than VNS_{red} and VNS_{full} for most of the tested instances. The experiment was performed with the best distance, average distance, and CPU time. Also, a statistical test was made by each group to compare the averages. For the instance R101_21x, the proposed IG has a gap of the best distance of 0% while VNS_{full} has 27%. The major limitation in all related problems with 2E-EVRP (including this work) is using a linear function to calculate the recharge time of the battery due to the nonlinear property [Pelletier et al. 2017] of the recharging time of the battery.

In future work, we intend to create new instances for the 2E-EVRP-TW as well as add the destruction of the first echelon. We also intend to propose a new model that includes the nonlinear recharging function time, as [Montoya et al. 2017].

7. Acknowledgment

The authors thank the financial support provided by CAPES, CNPq, FAPEMIG, and UFJF.

References

Akbay, M. A., Kalayci, C. B., Blum, C., and Polat, O. (2022). Variable neighborhood search for the two-echelon electric vehicle routing problem with time windows. *Applied Sciences*, 12(3):1014.

Breunig, U., Baldacci, R., Hartl, R. F., and Vidal, T. (2019). The electric two-echelon vehicle routing problem. *Computers & Operations Research*, 103:198–210.

Jie, W., Yang, J., Zhang, M., and Huang, Y. (2019). The two-echelon capacitated electric vehicle routing problem with battery swapping stations: Formulation and efficient methodology. *European Journal of Operational Research*, 272(3):879–904.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Marte, R. et al. (2018). *Handbook of heuristics*. Springer,.

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

Pelletier, S., Jabali, O., Laporte, G., and Veneroni, M. (2017). Battery degradation and behaviour for electric vehicles: Review and numerical analyses of several models. *Transportation Research Part B: Methodological*, 103:158–187.

Romm, J. J. (2022). *Climate change: What everyone needs to know*. Oxford University Press.

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation science*, 48(4):500–520.

Sluijk, N., Florio, A. M., Kinable, J., Dellaert, N., and Van Woensel, T. (2022). Two-echelon vehicle routing problems: A literature review. *European Journal of Operational Research*.

Souza, M. J., Coelho, I. M., Ribas, S., Santos, H. G., and Merschmann, L. H. d. C. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, 207(2):1041–1051.

Subramanian, A., Drummond, L. M., Bentes, C., Ochi, L. S., and Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911.

Toth, P. and Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

Wang, D. and Zhou, H. (2021). A two-echelon electric vehicle routing problem with time windows and battery swapping stations. *Applied Sciences*, 11(22):10779.