applied sciences

MDPI

Type of the Paper (Article, Review, Communication, etc.)

A Hybrid Genetic Algorithm approach to the Multi Charging Electric Vehicle Capacitated Routing Problem with Time Windows and Recharging Stations

Juan Betancourt [1], Alejandra Tabares [1,*] and Daniel Giraldo Herrera [1]

[1] Universidad de Los Andes, Bogotá, Colombia; jm.betancourt@uniandes.edu.co
[1] Universidad de Los Andes, Bogotá, Colombia; ds.giraldoh@uniandes.edu.co
* Correspondence: a.tabaresp@uniandes.edu.co; Tel.:

Abstract: The Multiple Charging Stations Electric Vehicle Capacitated Routing Problem with Time Windows and Recharging Stations (EV-CRPTW) is a complex combinatorial optimization problem that arises in the context of last-mile delivery with electric vehicles. The problem consists of finding a set of routes that visit a given set of customers, while satisfying the following constraints: The total distance traveled by all vehicles must be minimized; each vehicle must have sufficient battery capacity to reach all customers on its route; each customer must be visited within its specified time window; and vehicles can recharge at multiple recharging stations, but recharging times must be taken into account. In this paper, we propose a hybrid genetic algorithm (GA) for solving the EV-CRPTW. The GA is based on a feasible solution approach, which means that it only generates feasible solutions. The GA starts by generating a population of feasible solutions using a random-cluster-based heuristic. The heuristic constructs routes by randomly selecting customers from a set of suitable candidates. The GA then improves the routes in the population using crossover and mutation operators. The hybridization of the GA relies on local-search operators that evaluate and repair the generated routing schemes. We evaluated the performance of the proposed GA on a set of benchmark instances. The results show that the GA outperforms state-of-the-art methods for solving the EV-CRPTW. The GA is able to find high-quality solutions in a reasonable amount of time. The following are some additional details about the proposed GA: 1) The GA uses a population of feasible solutions. This means that the GA does not generate infeasible solutions, which can save a significant amount of time; The GA uses crossover and mutation operators to improve the routes in the population. 2) These operators are well-known and have been shown to be effective for solving other routing problems. 3) The GA uses local-search operators to evaluate and repair the generated routing schemes. These operators can help to improve the quality of the solutions found by the GA. The proposed GA is a promising approach for solving the EV-CRPTW. The GA is able to find high-quality solutions in a reasonable amount of time. The GA is also able to handle a wide range of problem instances. Put any relevant numerical result.

Keywords: Multi Charging Electric Vehicle Capacitated Routing Problem, Time Windows, Recharging Stations, Genetic Algorithm, Hybridization, Local Search.

Citation: To be added by editorial staff during production.

Academic Editor: Firstname Lastname

Received: date
Revised: date
Accepted: date
Published: date

1. Introduction

In an increasingly interconnected world, the urge for faster, more efficient, and sustainable supply and distribution chains is an important motor for innovation. As defined by [1] environmental sustainability is one of the principal dimensions when discussing sustainability. One of the main mechanisms to reduce the environmental footprint, is to transition from fossil fuel to renewable energy sources as the fuel in transportation on supply chains (SC). Due to the popularity of electric vehicles and policies to decrease the carbon footprint, organizations have started to transition from a fleet of internal

combustion vehicles to electric vehicles. Despite the great catch they have, these vehicles contemplate several operational challenges that conventional fleets do not have. For instance, the distances these vehicles can travel are shorter than combustion vehicles and the electrical charging network is not as developed as for internal their internal combustion counterpart. Furthermore, the time required to charge these vehicles can range from 30 minutes to several hours depending on the vehicle and the charger type [2].

The usual logistic tasks performed through a distribution network can be represented as vehicle routing problems (VRP). In this problem, a total transportation cost is minimized when visiting a set of customers with routes that start and end at a depot. Over the years, multiple variants for these VRPs have been developed as more realistic and applicable models are required. The most usual ones are capacitated vehicle routing problems (CVRP), in which vehicles have limited capacity and each client a defined associated quantity [3]. In addition, when customers have a limited time window in which they can be serviced are called VRPs with time windows (VRPTW). This constraint changes the visiting dynamics, now all vehicles must assure visiting each client within its designated time interval; if a vehicle reaches a costumer before it is ready to be serviced, it must wait to initiate the service until the costumer's time window starts. The objective of the CVRPTW is to comply with all constraints with a minimum number of vehicles and travel distance. Other ranking criteria for solutions are the schedule and waiting times [4].

Nowadays, electric vehicles (EVs) have remarkable performance and can travel longer distances on a full charge, even so, only sometimes EV for longer distances are the best option. While EVs with longer ranges are available, they can be more expensive and may only be necessary for some applications. For example, EVs with longer ranges may be overkill for last-mile delivery, where vehicles typically only travel short distances. Furthermore, to make these vehicles sustainable, an energy supply network must be in place so that they can be efficiently harnessed. However, contemplating these charging times is critical to assure the delivery of orders to customers within their time windows.

Hence, this work presents a metaheuristic approach to solve the multi charging electric vehicle routing problem with time windows and charging stations (EV-CVRPTW) based on a Hybrid Genetic Algorithm (HGA). The metaheuristic considers a population of feasible solutions and explores the solution space through the iterative use of different operators and updates on the individuals of the population. In concept, the better individuals have a higher probability of carrying their genes (routes) to some extent, due to cross-over and mutation, to the next generation. Therefore, the algorithm should provide better and better solutions as more generations are simulated. The methodology incorporates the possibility of recharging the vehicle's battery at charging stations during any route stage. It also considers capacity, energy constraints, service time, and customer time windows.

This article presents on section 2 the problem definition, and a literature-coherent mathematical formulation is presented. A thorough literature review is presented on section 3, remaking the areas of possible contribution. On section 3, the genetic algorithm with the methodology and operators are defined in pseudo-codes and explained to the detail. On section 4, the instances used on the experiments are referenced and all the conditions of the experimentation are defined. On section 5, the results are shown, and a corresponding analysis is performed. Finally, on section 6 conclusions are made over the results and section 7 presents areas of future work.

2. Problem definition

The critical factor of electric vehicles is energy; the energy consumption of these vehicles and any other in general depends on the power required to move, depending on the weight of the load to accelerate the vehicle when there are sloping roads. The E-VRPTW model contains some considerations, such as that the deliveries are made on flat terrain and without inclines; it does not consider the increase in battery consumption if the vehicle is carrying a load or not [5]. The speed of the vehicle is constant between vertices. All these factors are essential in measuring battery consumption and determining

whether the vehicle should visit recharging stations, how much it should recharge, and how long it will take.

It is defined in a complete directed network $G = (V_{N+1}, A)$ with a set of arches $A = \{(i,j)|i,j \in V'_{0,N+1}, i \neq j\}$. For each arc there is an associated distance $d_{i,j}$ and a travel time $t_{i,j}$. Traveling through an arc i,j consumes a portion of the battery's energy $h * d_{i,j}$, being h an energy consumption rate constant. The vehicle fleet is homogeneous and has a charging capacity C, this vehicle flora always starts from the depot. Each vertex in the network except for the charging stations has a positive demand of q_i, also has a time window $[e_i, l_i]$ in which it can be serviced, all these vertex $i \in V_{0,N+1}$ have a service time s_i which cannot start before e_i, so if the truck arrives before the time window starts it must wait, and it is not possible to be serviced after the end of the time window l_i, however, the service time s_i can end after the time window. At charge stations, the vehicle will recharge until complete charge Q based on his actual charge, with a rate charge g that recharge the vehicle linearly.

$0, N+1$	Depot instances
F'	Set of visits to recharging stations, dummy vertices of set of recharging stations F
F'_0	Set of recharging visits including depot instance 0
V	Set of customers V = {1, …, N}
V_0	Set of customers including depot instance 0
V'	Set of customer vertices including visits to recharging stations, $V' = V \cup F'$
V'_0	Set of customers and recharging visits including depot instance 0: $V'_0 = V' \cup \{0\}$
V'_{N+1}	Set of customers and recharging visits including depot instance $N+1$: $V'_{N+1} = V' \cup \{N+1\}$
$V'_{0,N+1}$	Set of customers and recharging visits including depot instances 0 and N+1: $V'_{0,N+1} = V' \cup \{0\} \cup \{N+1\}$
$d_{i,j}$	Distance between vertices i and j
$t_{i,j}$	Travel time between vertices i and j
C	Vehicle capacity
g	Recharging rate
h	Charge consumption rate
Q	Battery capacity
q_i	Demand of vertex $i, 0\ if\ i \notin V$
e_i	Earliest start of service at vertex i
l_i	Latest start of service at vertex i
S_i	Service time at vertex i ($S_0, S_{N+1} = 0$)
τ_i	Decision variable specifying the time of arrival at vertex i
U_i	Decision variable specifying the remaining cargo on arrival at vertex i
y_i	Decision variable specifying the remaining battery capacity on arrival at vertex i
x_{ij}	Binary decision variable indicating if arc (i,j) is traveled

$$\sum_{i \in V'_0, j \in V'_{N+1}, i \neq j} d_{ij} x_{ij} \tag{1}$$

$$\sum_{j \in V'_{N+1}, i \neq j} x_{ij} = 1 \ \forall i \in V \tag{2}$$

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

$$\sum_{j \in V'_{N+1}, i \neq j} x_{ij} - \sum_{i \in V'_0, i \neq j} x_{ij} = 0 \ \forall j \in V' \tag{4}$$

$$\tau_i + \left(t_{ij} + s_{ij}\right) x_{ij} - l_0\left(1 - x_{ij}\right) \leq \tau_j \quad \forall\, i \in V_0, \forall j \in V'_{N+1}, i \neq j \tag{5}$$

$$\tau_i + t_{ij}x_{ij} - g(Q - y_i) - (l_0 + gQ)(1 - x_{ij}) \leq \tau_j \; \forall \; i \in F', \forall j \in V'_{N+1}, i \neq j \quad (6)$$

$$e_j \leq \tau_j \leq l_j \; \forall j \in V'_{0,N+1} \quad (7)$$

$$0 \leq u_j \leq u_i - q_i x_{ij} + C\left(1 - x_{ij}\right) \forall i \in V'_0, \forall j \in V'_{N+1}, i \neq j \quad (8)$$

$$0 \leq u_0 \leq C \quad (9)$$

$$0 \leq y_i \leq y_i - \left(h * d_{ij}\right)x_{ij} + Q\left(1 - x_{ij}\right), \forall j \in V'_{N+1}, i \in V, i \neq j \quad (10)$$

$$0 \leq y_i \leq Q - \left(h * d_{ij}\right)x_{ij}, \forall j \in V'_{N+1}, i \in F'_0, i \neq j \quad (11)$$

$$x_{ij} \in \{0,1\} \;\; \forall i \in V'_0, \forall j \in V'_{N+1}, i \neq j \quad (12)$$

Equation (1) is the objective function which seeks to minimize the distances traveled. Constraint (2) makes it must visit customers; constraint (3) manages the visit of vehicles to charging stations. Constraint (4) guarantees flow conservation in the network, where a vehicle cannot stay at a customer vertex or charging station, the number of incoming arcs must be equal to the number of outgoing arcs. Constraint (5) guarantees the feasibility of time windows for customers visited from the depot and constraint (6) guarantees the feasibility of serving customers from the refueling stations. Constraint (7) guarantees that we can serve the customer only within the time window it has. Constraints (8) and (9) guarantee the fulfillment of the demand of all customers in the network considering truck loads and non-negativity. Constraints (10) and (11) make sure that the electric vehicle charge never reaches zero; it is necessary to clarify that every time a vehicle arrives to recharge at a station, it will charge the battery to the maximum. Constraint (12) presents the natures of the variables.

3. Literature review

The multi charging electric vehicle capacitated routing problem with time windows (EV-CVRPTW) is a type of vehicle routing problem that involves scheduling a fleet of vehicles to serve a set of customers, each with specific time windows for service, while considering the limited battery capacity of the vehicles, charging times and costs, and other constraints such as customer demand and vehicle capacities. Solving the EV-CVRPTW is a complex optimization problem, and several methods have been proposed in the literature to find efficient and effective solutions.

The most cited paper for the EV-CRPTW is [6]. This work proposes a tabu search heuristic for the EV-CRPTW. The heuristic is based on a two-phase approach. A feasible solution is constructed using a greedy algorithm in the first phase. In the second phase, the tabu search heuristic improves the solution quality. The tabu search heuristic can find significantly better solutions than the greedy algorithm. The authors show that the tabu search can find significantly better solutions than the greedy algorithm.

One of the methods used to solve the EV-CRPTW is genetic algorithms (GAs). GAs is a heuristic optimization method inspired by natural selection and genetics. They use a population of candidate solutions that evolve by applying genetic operators such as crossover and mutation. The goal is to find the best solution to the problem by iteratively improving the candidate solutions over multiple generations. In the literature, several studies have proposed using GAs to solve the EV-CRPTW. These studies have shown that GAs is a promising approach for finding efficient and effective solutions to the EV-CRPTW. Work [7] proposes the first one for the EV-CRPTW. The GA starts with a population of randomly generated solutions. Each solution is a sequence of nodes, and each node is assigned to a vehicle. The fitness of a solution is calculated as the total distance traveled plus the sum of the tardiness penalties for all nodes that are not delivered on time. The

GA then iteratively improves the population of solutions. In each iteration, the following steps are performed: 1) Selection: A population subset is selected for reproduction. The selection process is based on the fitness of the solutions. 2) Crossover: Two solutions are selected from the subset, and their genes are crossed to create new solutions. 3) Mutation: Some random changes are made to the genes of some of the new solutions. 4)Evaluation: The fitness of the new solutions is calculated. 5) Replacement: New solutions are added to the population, and some old ones are removed. The GA is repeated until a termination criterion is met.

For instance, a study by Yang et al. (2015) [8], proposed a GA-based solution for the EVPTW with time windows and battery constraints. The study showed that the GA-based answer could find good quality solutions in a reasonable amount of time, compared to other optimization methods such as linear programming and ant colony optimization.

A recent study by Zhang et al. (2021) [9], proposed a hybrid GA-based solution for the EVPTW with time windows and battery constraints. The study combined the advantages of GAs and local search algorithms to find reasonable quality solutions in a shorter amount of time. The study showed that the hybrid GA-based solution could find high-quality solutions comparable to those found by other optimization methods, such as simulated annealing and tabu search.

The literature review shows that GAs is a promising approach for solving the E-CVRP-TW. Several studies have proposed using GAs for the E-CVRP-TW and have demonstrated that GAs can find good-quality solutions in a reasonable amount of time compared to other optimization methods. The literature also suggests that hybrid GAs that combine the advantages of GAs and other optimization methods may be particularly effective for solving the E-CVRP-TW using feasible and infeasible solutions, looking to repair a fraction of the infeasible solution to explore the search space.

Work	Author(s)	Metaheuristic	Results
[10]	H. Mao, J. Shi, Y. Zhou and G. Zhang (2020)	Ant colony optimization algorithm	improves quality functions by 3%-39%
[11]	Wang, L., Gao, S., Wang, K., Li, T., Li, L., & Chen, Z. (2020).	Variable neighborhood search algorithm	The range of well services eficiency for the EVs is 120km-260km
[12]	Zang, Y., Wang, M., & Qi, M. (2022)	Column generation optimization	model can result in lower operational cost, about 9%–10% less than models with non linear discharge
[13]	Yilmaz, Y., & Kalayci, C. B. (2022)	Variable neighborhood search algorithm	Reach the optima for small instances in short times 0.03 s - 330 s
[14]	Fateme Attar, S., Mohammadi, M., Reza Pasandideh, S. H., & Naderi, B. (2022)	Mathematical model optimization wirh benders descomposition	Better preformance in the time solutions in the schneider instances
[15]	Löffler, M., Desaulniers, G., Irnich, S., & Schneider, M. (2020)	Large neighborhood search with granular tabu search algorithm	The amount of improvement is substantial on many of the large instances,resulting in average gaps of−1.13% and−0.81%
[16]	Arias, A., Sanchez, J., & Granada, M. (2018)	Mathematical model optimization	Created their own instances to the E-VRP
[17]	Jiang, Z., Bao, F., & Wang, N (2023)	Ant colony optimization algorithm	Improves solution quality by 22% and runing times in 34.35%

4. Methodology

As a solution for the Electric Vehicle Capacitated Routing Problem with Time Windows and multiple charging stations, we propose a genetic algorithm scheme with several crossover and mutation operators for the routes. The algorithm combines elitism, random sampling with a fitness function, and tournament to exploit three key ideas:

1. The use of a population of solutions to guide search.
2. The use of crossover operators that recombine two or more solutions to generate new and potentially better solutions.
3. The active management of diversity to sustain exploration.

New ideas that are also introduced in this chapter include (1) the use of deterministic recombination operators that can tunnel between local optima and (2) the use of deterministic constant time move operators [3, 8, 9, 10]. The EV-CRPTW has constraints that make the problem challenging and defiant; even evaluating a single route can be a difficult task from the computational burden due to the need to verify the feasibility of energy, visit the charging stations, arrival times, and capacity of EV.

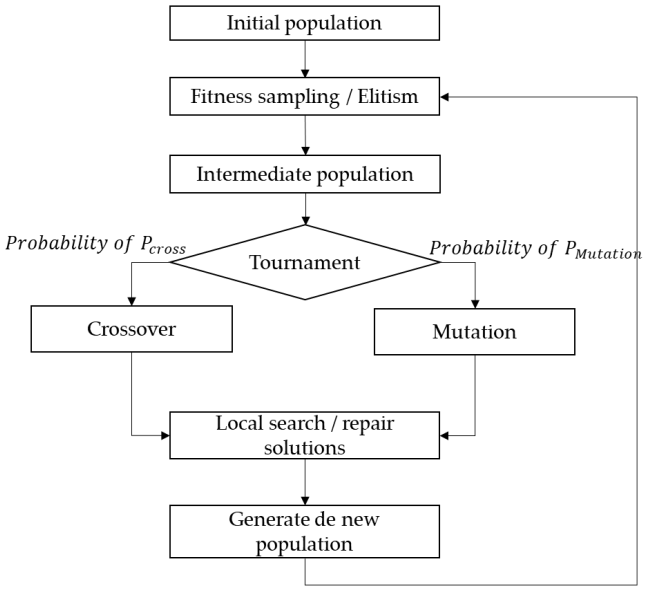

Figure 1. Overall view of the methodology

4.1 Initial population

Before the genetic algorithm can take place, an initial pool of feasible solutions (population) is required. This population must ensure diversity in solutions to promote solution space exploration. Furthermore, we require feasible routes based on three different criteria: (1) the EV can arrive at the customer if it can be attended to in his time window, (2) the EV has enough energy, then can reach at least a charging station after visits a customer, (3) the EV has enough capacity load to supply the demand at a customer. For this, we designed a Restricted-Candidate-List (RCL)-based heuristic founded in the approach proposed by [3]. The heuristic is randomized to achieve a high level of diversity. A pseudocode for this heuristic is presented in Algorithm 1.

Algorithm: *RCL based constructive* (α, *v*, *r*, *g*, *dist*, *closest*, *RT*, *DD*, *ST*, *mode*, *slack*)

1: $t \leftarrow 0$
2: $q \leftarrow Q$
3: $k \leftarrow 0$
4: *node* ← *Depot*
5: *route* ← {*node*}
6: *stable* ← **true**

```
7: while stable do
8:     target, energy feasible ← generate candidate from RCL (node, t, q, k, v, r, d, v, r, dist, α,
       mode, slack)
9:     if target = false then
10:            route, t, q, k ← route costumers (node, target, route, t, q, k, v, r, d, dist)
11:            node ← target
12:    else
13:            if energy feasible then
14:                    closest station ← closest(node)
15:                    t ← t + dist (node, closest station)/v
16:                    t ← t + (Q − q) ∗ g
17:                    q ← Q
18:                    node ← closest station
19:                    target, energy feasible ← generate candidate from RCL (node, t, q,
                       k, v, r, d, v, r, dist, α, mode, slack)
20:                    if target = false then
21:                            route, t, q, k ← route costumers (node, target, route, t, q, k,
                               v, r, d, dist)
22:                            node ← target
23:                    else
24:                            route, t, q, k ← route to depot (t, q, k, node)
25:                    end if
26:            else
27:                    route, t, q, k ← route to depot (t, q, k, node)
28:            end if
29:    end if
30: end while
31: return t, q, k, route
```

Algorithm 1. RCL-based constructive heuristic.

The constructive heuristic requires the following parameters: alpha, the speed, the energy consumption and recharge rates, the distances, the closest station to each customer, the ready, due, and service times of each customer, the criterion to restrict the list of candidates, and finally, a slack parameter for routing termination. The heuristic returns the total time, final charge, load, and route. Lines 1-5 initialize time, charge, load, and route (starts at the depot). While the route has inserting options (Line 6), The function returns a target node append to the route or *False* if there is no feasible candidate. If there is a feasible candidate (Lines 9-11), it is appended to the route, and the metrics are updated. If there is not (Line 12), the variable stores whether at least one candidate is feasible by time and load but not by charge (not enough to reach). If there is (Line 13), the vehicle is routed to the nearest station and charged fully (Lines 14-18). Then, a new candidate is selected. If there is one, it will be visited next (Lines 20-22). If there is no (Lines 23-24) or no energy-feasible candidate (Lines 26-27), the vehicle is routed to the depot.

Algorithm *Generate candidate from RCL* (*node, t, q, k, v, r, d, dist, α, mode, slack*)

```
1: feasible candidates ← {}
2: crit_max = −M
3: crit_min = M
4: energy feasible ← false
5: for target in pending costumers do
6:     feasible, energy feasible ← evaluate candidate (candidate)
7:     if feasible then
8:         feasible candidates ← feasible candidates ∪ {target}
9:         crit ← criterion defined by parameter mode(target)
```

10: $crit_{max} \leftarrow \max(crit, crit_{max})$
11: $crit_{min} \leftarrow \min(crit, crit_{min})$
12: **end if**
13: **end for**
14: **if** *feasible candidates* = {} **then**
15: **return false**, *energy feasible*
16: **else**
17: *upper bound* $\leftarrow crit_{min} + \alpha\,(crit_{max} - crit_{min})$
18: *RCL* ← candidates with criterion < *upper bound*
19: *target* ← selected random from *RCL*
20: **return** target, energy feasible
21: **end if**
22: **return** Updated time, charge and load, route

Algorithm 2. Constructing the RCL and selecting the candidate.

The pseudo-code on Algorithm 2. details how the Restricted List is constructed, and the candidate is selected. This algorithm uses the following inputs: the current node, the current time, charge and load, other instance parameters, the accumulated distance, the distances, alpha, and the criterion to restrict the list and the slack. It returns the updated time, charge, load, and route. Lines 1-4 initialize the list of candidates, the minimum and maximum values of the criterion, and the Boolean existence of an energy-feasible candidate. This work considers two different criteria: distance and time window finalization. The candidate's feasibility is evaluated for all unvisited customers (Line 5), and the candidate's feasibility is assessed (Line 6). The detailed criteria are:

1. Time window: The accumulated time plus the travel time to the candidate is less than the due date of the service for that customer.
2. Energy: The vehicle has enough energy to visit the candidate and attend to recharge to the nearest station.
3. Capacity: The vehicle has enough load capacity to service the candidate

If the candidate satisfies all three conditions, it will be appended to the restricted list (Line 8). Then, the designed criterion of the candidate is stored, and the maximum and minimum criteria values are updated (Lines 9-11). If there are no feasible candidates (satisfies all three conditions), the algorithm returns false and the variable (Lines 14-15). If at least one feasible candidate (Line 16), an upper bound is computed using the maximum and minimum values found, as shown in Line 17. Then, the candidate list is assembled with those candidates whose criterion value is less or equal to the upper bound computed (Line 18), and a candidate is selected randomly from the list (Lines 19-20).

As for the other two functions displayed on Algorithm 1, there will be no pseudo-code shared, but the main logic of the algorithms is presented below:

- *Route customers:* As it is used on entirely feasible transitions, this function updates the time with the travel and service times, updates the charge with the loss of energy on the transportation, updates the load with the customer's demand, and updates the route by adding the customer to the route.

- *Route to the depot:* If the vehicle can transport from the current node to the depot with the current charge, then it's routed directly to the depot, and the time, charge, and route are updated accordingly. If there isn't enough energy, the closest station (best station) to the current node and the depot are computed. If the vehicle has enough energy to reach this station, it's added to the route, and the time and charges are updated accordingly. If the vehicle cannot get the best station, the vehicle is routed through the closest station to the current node and the depot.

4.3 Intermediate population

After an initial feasible population is generated, the genetic process begins. The first step in every algorithm iteration is to generate an intermediate population from which parents will be selected. This ensures that the best individuals will endure the recombination and mutation processes. This intermediate population is set to be the same size as the initial population. However, the order and number of times every individual appears in the intermediate population varies. The placement of individuals into the intermediate population has been randomized, and adjacent individuals can be recombined [10] and routed through the closest station to the current node and the depot.

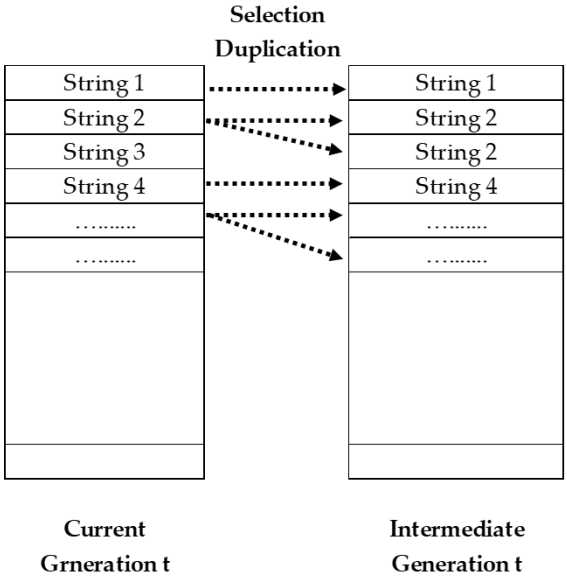

Figure 2. Constructing the RCL and selecting the candidate

4.3.1 Elitism

As the genetic algorithm progresses, it would be desirable to build a list with the best-known individuals and somehow convince them to seek local searches y promisor neighborhoods. This is precisely the concept attacked by an elite class. On each iteration, the best 5% of individuals prevailed over the intermediate population.

4.3.2 Fitness sampling

The remaining 95% of individuals of the intermediate population are randomly sampled out of the initial population (including the elite individuals). The sampling is done via a fitness function computed for every individual, as shown in Equation 1.

$$f_i = \frac{\sum_{j \in Population} d_j}{d_i} \qquad \forall \qquad i \in Population$$

Equation 1. Computation of the fitness function.

Were, for every individual i of the initial population, the fitness function will be larger for individuals with smaller distances and smaller for individuals with larger distances. With this fitness function, the probability of choosing an individual for the intermediate population is computed as shown in Equation 2.

$$p_i = \frac{f_i}{\sum_{j \in Population} f_j} \qquad \forall \qquad i \in Population$$

Equation 2. Computation of the probability based on fitness.

Hence, the probability for individuals with smaller distances (objective) will have a larger probability of being chosen for the intermediate generation.

4.3.3 Tournament

The pairs of individuals to recombine and mutate from the intermediate population are chosen through a tournament. To generate each pair of parents, two individuals are chosen randomly, and the best one, ergo tournament, is selected as one of the parents. For the second parent, the same process is applied. As mentioned in Section 3.1, one pair of parents will produce one offspring. Hence, there will be as many pairs of parents as individuals in the population.

4.3.4 crossover 2-OPT

This operator is somewhat more complex than its predecessor. Again, two positions are chosen at random from the vector. However, these positions must be at least 2 positions apart. The new vector will remain with the first section (from the start to the first position) and the final section (from the second position to the end). As for the middle section, the costumers will be reversed in order. Figure x shows the operator on the vector example. For the example, positions will be 1 and 6 (first and last). Therefore, these two positions will remain the same and the nodes in between ('C5', 'C6', 'C3' and 'C4') will be reversed in order.

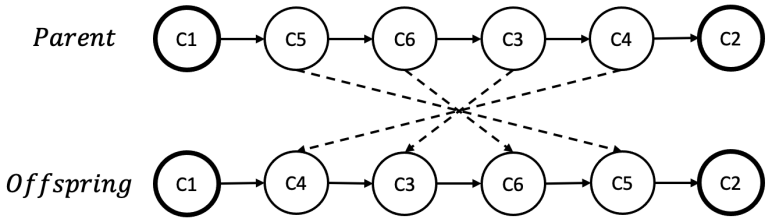

Figure 2. 2-Opt operator

4.3.5 Mutation

The algorithm 3 is called Darwinian phi rate is a specialized mutation algorithm that initializes by organizing the routes of the individual based on a performance rate given by the route's total distance divided by the number of nodes (customers and service stations) it visits. It then iterates over each route and the performance rate and checks if the length of the route is greater than or equal to a ρ times the maximum size of all routes. If true, it adds that route to the new individual, and that route's distances, nodes, and times are updated. The routes that do not exceed this ratio are eliminated. Through the constructive algorithm, the missing clients are added on the remaining routes, resulting in individuals with fewer routes representing better solutions as they are more compact and visit other clients that previous individuals did not, expanding the search space and exploiting new solutions.

Algorithm Darwinian phi rate (*RCLbasedconstructive, individuals, route, t, q, k, v, r, d, dist, penalization,* α, σ, ρ)

1: *Rankedindex* ← {}
2: *route efficiency*$_i$ ← *traveled distance*$_i$ / (*nodes visited*$_i^{penalization}$)
3: *Ranked index* ← *Sorted* (*route efficiency*$_i$)
4: *new individual* ← {}
5: *new distance* ← 0
6: *new distances* ← {}
7: *new time* ← 0
8: *new times* ← {}
9: *Pending_c* ← *all costumers*
10: $i = 0$
11: *cont* = 0
12: **while** (*i / individual routes*) < σ **do**

```
        ii ← Ranked route_i
        cont ← cont + 1
        if cont = nodes route then
        if route nodes_ii >= ρ * max (route_i (nodes visited) then
            new individual ← route_ii
            new distance ← distance + distance route_ii)
            new distances ← {distances between nodes_ii}
            new time ← time + time route_ii
            new times ← {times between nodes_ii}
            for node in route_ii do
                if node is a customer then
                    remove node from Pending_c
                end if
            end for
        end if
        end if
        i = i + 1
end while
while Pending_c ≠ empty do
        RCLcriterion ← choice [distance, time window]
        route, t, d, q, k ← route costumers (node, target, route, t, q, k, v, r, d, dist)
        new individual ← route_ii
        new distance ← distance + distance route_ii)
        new distances ← {distances between nodes_ii}
        new time ← time + time route_ii
        new times ← {times between nodes_ii}
end while
return new individual, routes, load, distances, time
```

There is a warming time to define which α, σ, and ϱ is the best for the instance, to generate the restricted candidate list, the proportion of routes that will stay in the individual, and the last hyperparameter; while the maximum number of nodes of that route is less than a proportion of the best routes, we will take the nodes from that route and insert them in other routes taking into account feasibility charge, load a time windows. The penalty for indexing the routes is supported by the number of vertices visited in the route, which can be: Lineal, quadratic or cubic. Furthermore, the warming period establish which hyperparameter presents better solutions in the RCL constructive and in the crossover and mutation operators.

4. Discussion

To validate the performance of the proposed heuristic, we used classic instances proposed by schneider et. 2014 [5]. Those instances contain a set of 56 large-sized instances consisting of 100 customer locations and 21 charging stations, and a set of 36 small-sized instances consisting of 5, 10, and 15 customer locations. The charging stations in each instance are determined as follows: One of the charging stations is located at the depot node. The remaining charging stations are located randomly with the assumption that each customer can be reached from the depot using at most two charging stations. The VRPTW dataset is divided into three classes based on geographical distribution: random customer distribution (R), clustered customer distribution (C), and a mixture of both R and C classes (RC).

Table 1. Small instances experimentation

Ranking								
Rank	Experiment	Darwinian		eval insertion	Genetic			Gap
		pen	length r	pen	pop. size	c. rate	m. rate	
1	19	quadratic	TRUE	cubic	1500	0.6	0.3	9.52%
2	11	quadratic	TRUE	quadratic	1500	0.6	0.3	9.53%
3	3	quadratic	TRUE	regular	1500	0.6	0.3	9.57%
4	42	quadratic	FALSE	cubic	1500	0.3	0.6	9.82%
5	34	quadratic	FALSE	quadratic	1500	0.3	0.6	9.85%
6	66	cubic	TRUE	cubic	1500	0.3	0.6	10.00%
7	32	quadratic	FALSE	regular	3000	0.6	0.6	10.14%
8	40	quadratic	FALSE	quadratic	3000	0.6	0.6	10.14%
9	43	quadratic	FALE	cubic	1500	0.6	0.3	10.15%
10	88	cubic	FALSE	quadratic	3000	0.6	0.6	10.16%

Table 2. Large instances experimentations

Ranking								
Rank	Experiment	Darwinian		eval insertion	Genetic			Gap
		pen	length r	pen	pop. size	c. rate	m. rate	
1	44	quadratic	FALSE	cubic	1500	0.6	0.6	52.05%
2	42	quadratic	FALSE	cubic	1500	0.3	0.6	53.34%
3	34	quadratic	FALSE	quadratic	1500	0.3	0.6	53.70%
4	38	quadratic	FALSE	quadratic	3000	0.3	0.6	54.41%
5	78	cubic	FALSE	regular	3000	0.3	0.6	54.43%
6	86	cubic	FALSE	quadratic	3000	0.3	0.6	54.62%
7	92	cubic	FALSE	cubic	1500	0.6	0.6	54.75%
8	68	cubic	TRUE	cubic	1500	0.6	0.6	55.08%
9	45	quadratic	FALSE	cubic	3000	0.3	0.3	55.09%
10	46	quadratic	FALSE	cubic	3000	0.3	0.6	55.25%

Table 3. Small instances results.

Inst.	CPLEX			Hybrid Genetic Algorithm			
	#EV	F. O	t	# EV	F. O	t	gap
C101-5	**2**	**257.75**	**81**	**3**	**250.04**	**0.13**	**-2.99%**
C103-5	1	176.05	5	1	185.01	0.13	5.09%
C206-5	1	242.55	518	1	254.13	0.09	4.77%
C208-5	1	158.48	15	1	201.35	0.12	27.05%
R104-5	2	136.69	1	2	137.01	0.11	0.23%
R105-5	2	156.08	3	2	174.03	0.14	11.50%

R202-5	1	128.78	1	1	144.67	0.1	12.34%
R203-5	1	179.06	5	1	241.42	0.11	34.83%
RC105-5	**2**	**241.3**	**764**	**3**	**239.46**	**0.11**	**-0.76%**
RC108-5	1	253.93	311	2	274.1	0.15	7.94%
RC204-5	1	176.39	54	2	257.38	0.1	45.92%
RC208-5	1	167.98	21	1	174.38	0.13	3.81%
C101-10	3	393.76	171	4	446.26	0.37	13.33%
C104-10	2	273.93	360	2	292.48	4.35	6.77%
C202-10	**1**	**304.06**	**300**	**2**	**280.41**	**0.33**	**-7.78%**
C205-10	2	228.28	4	2	265.5	3.02	16.30%
R102-10	3	249.19	389	4	262.92	0.29	5.51%
R103-10	2	207.05	119	2	208.31	10.63	0.61%
R201-10	**1**	**241.51**	**177**	**2**	**228.36**	**13.26**	**-5.44%**
R203-10	1	218.21	573	1	235.95	0.28	8.13%
RC102-10	4	423.51	810	5	436.05	0.47	2.96%
RC108-10	3	345.93	399	3	347.9	2.26	0.57%
RC201-10	**1**	**412.86**	**7200**	**2**	**381.05**	**2.35**	**-7.70%**
RC205-10	2	325.98	399	2	361.04	1.76	10.76%
C103-15	3	384.29	7200	4	394.8	63.76	2.73%
C106-15	3	275.13	17	3	334.65	23.58	21.63%
C202-15	2	383.62	7200	3	409.37	11.57	6.71%
C208-15	**2**	**300.55**	**5060**	**2**	**300.55**	**2.73**	**0.00%**
R102-15	5	413.93	7200	6	439.22	32	6.11%
R105-15	4	336.15	7200	4	345.95	3.52	2.92%
R202-15	2	358	7200	2	412.1	50.89	15.11%
R209-15	1	313.24	7200	1	476.69	0.55	52.18%
RC103-15	4	397.67	7200	5	422.03	42.74	6.13%
RC108-15	3	370.25	7200	3	385.02	4.75	3.99%
RC202-15	2	394.39	7200	2	454.8	14.78	15.32%
RC204-15	**1**	**407.45**	**7200**	**2**	**354.91**	**2.86**	**-12.89%**
					AVG	8.18	8.71%

Table 4. Large instances results

Inst.	BKS		Hybrid Genetic Algorithm			
	#EV	f	# EV	F. O	t	gap
c101	12	1053.83	15	1559.56	470.73	0.4799
c102	11	1056.47	16	1656.67	445.08	0.5681
c103	10	1002.03	15	1435.05	417.22	0.4321
c104	10	979.51	14	1313.04	324.94	0.3405
c105	11	1075.37	14	1465.86	200.61	0.3631
c106	11	1057.97	15	1465.88	247.58	0.3856
c107	11	1031.56	14	1424.22	472.38	0.3806

c108	10	1015.73	13	1291.57	475.94	0.2716
c109	10	1036.64	12	1269.76	283.73	0.2249
c201	4	645.16	6	965.99	352.45	0.4973
c202	4	645.16	6	1076.65	469.88	0.6688
c203	4	644.98	6	1116.37	299.28	0.7309
c204	4	636.43	5	1079.74	387.02	0.6966
c205	4	641.13	5	941.21	357.8	0.468
c206	4	638.17	6	1055.3	412.97	0.6536
c207	4	638.17	5	966.7	467.33	0.5148
c208	4	638.17	5	879.21	279.3	0.3777
r101	18	1670.8	25	2034.72	480.33	0.2178
r102	16	1495.31	22	1853.23	266.48	0.2394
r103	13	1299.17	19	1668.78	277.75	0.2845
r104	11	1088.43	15	1428.47	409.53	0.3124
r105	14	1401.24	19	1703.58	313.42	0.2158
r106	13	1334.66	19	1695.58	461.84	0.2704
r107	12	1154.52	16	1546.32	450.86	0.3394
r108	11	1050.04	14	1392.61	453.02	0.3262
r109	12	1294.05	17	1619.73	396.59	0.2517
r110	11	1126.74	16	1491.03	196.61	0.3233
r111	12	1106.19	16	1545.87	142.44	0.3975
r112	11	1026.52	14	1426.49	435.5	0.3896
r201	3	1264.82	6	1705.92	371.56	0.3487
r202	3	1052.32	6	1437.18	471.66	0.3657
r203	3	895.91	5	1343.71	481.75	0.4998
r204	2	790.57	4	1026.94	281.33	0.299
r205	3	988.67	5	1379.6	373.73	0.3954
r206	3	925.2	5	1340.9	301.3	0.4493
r207	2	848.53	5	1326.95	217.11	0.5638
r208	2	736.6	4	1088.48	472.36	0.4777
r209	3	872.36	5	1284.32	468.06	0.4722
r210	3	847.06	5	1179.63	296.88	0.3926
r211	2	847.45	4	1135.37	408.27	0.3397
rc101	16	1731.07	22	2191.23	415.2	0.2658
rc102	15	1554.61	19	2072.39	472.33	0.3331
rc103	13	1351.15	17	1808.33	151.06	0.3384
rc104	11	1238.56	14	1639.88	304.89	0.324
rc105	14	1475.31	19	1988.38	228.94	0.3478
rc106	13	1437.96	17	1896.05	314.88	0.3186
rc107	12	1275.89	15	1675.09	395.11	0.3129
rc108	11	1209.61	14	1608.76	469.34	0.33
rc201	4	1444.94	7	1918.15	281.39	0.3275
rc202	3	1412.91	6	1751.1	452.52	0.2394
rc203	3	1073.98	5	1556.55	433.25	0.4493

rc204	3	885.35	5	1316.44	391.5	0.4869
rc205	3	1321.75	6	1620.98	319.33	0.2264
rc206	3	1190.75	5	1727.45	245.67	0.4507
rc207	3	995.52	4	1275.96	389.67	0.2817
rc208	3	837.82	4	1150.71	211.33	0.3735
				AVG	361.95	38.63%

In applied logistics, the demand for immediate and efficient solutions is paramount. The complexity of real-world logistics problems often necessitates timely decision-making to optimize operations and ensure customer satisfaction. In this context, relying solely on exact mathematical modeling techniques may prove impractical due to their computational demands and inability to provide rapid solutions.

Our HGA, on the other hand, offers a powerful alternative for tackling logistics optimization problems. In Table 3, the small instances, the HGA improves the BKS based on distances, between 0.7% and 12%, where our function objective is minimizing distances. We can see the distances in six of the 36 instances. By leveraging principles inspired by natural evolution, genetic algorithms can efficiently explore a vast solution space and adaptively converge toward high-quality solutions. These algorithms find finding near optimal or satisfactory solutions within reasonable time frames.

5. Conclusions

The inherent ability of metaheuristics as genetic algorithms to handle large-scale, complex logistics problems and provide timely results makes them a preferred choice in practice. Their capacity to deliver satisfactory solutions within practical time constraints empowers logistics professionals to make informed decisions and swiftly adapt to dynamic environments.

Furthermore, genetic algorithms offer advantages in terms of flexibility and robustness. They can easily accommodate diverse problem structures, adapt to changing requirements, and handle uncertainties, making them well-suited for real-world logistics scenarios' inherent complexities and uncertainties.

References

[1] Negri, M., Cagno, E., Colicchia, C., & Sarkis, J. Integrating sustainability and resilience in the supply chain: A systematic literature review and a research agenda. Business Strategy and the Environment, (2021). 30 (7), 2858– 2886. https://doi.org/10.1002/bse.2776

[2] Pelletier, S., Jabali, O., & Laporte, G. 50th anniversary invited article—goods distribution with electric vehicles: review and research perspectives. Transportation science, (2016). 3-22.

[3] Cordeau, J. F., Laporte, G., Savelsbergh, M. W., & Vigo, D. Vehicle routing. In Handbooks in operations research and management scienc. (2007). (pp. 367-428.)

[4] Li, H., & Lim, A. Local search with annealing-like restarts to solve the VRPTW. European journal of operational research, (2003). 115-127.

[5] Schneider, M., Stenger, A., & Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. Transportation science, 500-520.

[6] Gendreau, M. G., Cordeau, J.-F., & Laporte, G. A tabu search heuristic for the electric vehicle routing problem with time windows. Transportation Science, (2004). 38(1), 30-43.

[7] Prins, C., Van Hentenryck, J. M., & Wagelmans, L. A. A genetic algorithm for the electric vehicle routing problem with time windows. Operations Research, 50(6), (2002). 1110-1124.

[8] Yang, H., Yang, S., Xu, Y., Cao, E., Lai, M., & Dong, Z. Electric vehicle route optimization considering time-of-use electricity price by learnable partheno-genetic algorithm. IEEE Transactions on smart grid, (2015), 6(2), 657-666.
[9] Zhang, Y., Zhou, S., Ji, X., Chen, B., Liu, H., Xiao, Y., & Chang, W. The Mathematical Model and an Genetic Algorithm for the Two-Echelon Electric Vehicle Routing Problem. In Journal of Physics: Conference Series (Vol. 1813, No. 1, p. 012006). (2021). IOP Publishing.
[10] The Electric Vehicle Routing Problem with Time Windows and Multiple Recharging Options, in IEEE Access, vol. 8, pp. 114864-114875, 2020, doi: 10.1109/ACCESS.2020.3003000.
[11] Wang, L., Gao, S., Wang, K., Li, T., Li, L., & Chen, Z. Time-dependent electric vehicle routing problem with time windows and path flexibility. Journal of Advanced Transportation, (2020), 1-19.
[12] Zang, Y., Wang, M., & Qi, M. A column generation tailored to electric vehicle routing problem with nonlinear battery depreciation. Computers & Operations Research, (2022), 137, 105527.
[13] Yilmaz, Y., & Kalayci, C. B. Variable Neighborhood Search Algorithms to Solve the Electric Vehicle Routing Problem with Simultaneous Pickup and Delivery. Mathematics, (2022). 10(17), 3108.
[14] Fateme Attar, S., Mohammadi, M., Reza Pasandideh, S. H., & Naderi, B. Formulation and exact algorithms for electric vehicle production routing problem. (2022).
[15] Löffler, M., Desaulniers, G., Irnich, S., & Schneider, M. (2020). Routing electric vehicles with a single recharge per route. Networks, 76(2), 187-205.
[16] Arias, A., Sanchez, J., & Granada, M. Integrated planning of electric vehicles routing and charging stations location considering transportation networks and power distribution systems. International Journal of Industrial Engineering Computations, (2018), 9(4), 535-550.
[17] Jiang, Z., Bao, F., & Wang, N. The Electric Vehicle Routing Problem with Time Windows and Three Charging Options. Available at SSRN 4263893.