Applied Intelligence
https://doi.org/10.1007/s10489-022-03555-8

# A greedy search based evolutionary algorithm for electric vehicle routing problem

**Vu Quoc Hien**[1] · **Tran Cong Dao**[1] · **Huynh Thi Thanh Binh**[1]

Accepted: 25 March 2022


**Abstract**

Over the years, there have been many variations of the Vehicle Routing Problem created to fit the actual needs of society, one of which is the Electric Vehicle Routing Problem (EVRP). EVRP is a more complex and challenging combinatorial optimization than the conventional vehicle routing problem. This paper considers a specific model for the tram routing problem and proposes a clustering-inspired greedy search algorithm GS. GS algorithm aims to cluster charging routes and greedily search charging stations for the optimal route output. In this paper, we purposely implement GS into a meta-heuristic genetic algorithm GA to utilize GA's finding a globally optimal, leading to the formulation of the GSGA algorithm. To evaluate performance, we use a benchmark dataset found in the CEC-12 Tram Routing Problem CEC-12 Competition at the World Congress on Computational Intelligence (WCCI) 2020. The experiment evaluates GS's effectiveness when applied to other algorithms such as genetic algorithms and simulated annealing. The experiments results show that our proposed algorithm has better solution quality than previous algorithms.

**Keywords** Electric vehicle routing problem · Greedy search · Meta-heuristic algorithm · Genetic algorithm

## 1 Introduction

Recently, the use of Electric vehicles (EVs) has become more crucial to transportation companies because of their low cost, low energy consumption, and environmental friendliness, contributing to reducing CO2 emission. EVs come with a rechargeable battery, but some issues may arise such as insufficient energy and low battery due to the limited number of charging stations. The need to manage EV's battery power raises the complexity and difficulty of the path planning problem for EVs.

In some recent studies on the problem of routing a fleet of EVs, M. Mavrovouniotis et al. [1] presents the Electric Vehicle Routing Problem (EVRP) for battery electric vehicles. He also presented the benchmark set to evaluate the proposed algorithms' effectiveness. The objective function of the EVRP is to find a set of routes that minimize the total distance traveled. The authors have proven that the EVRP is a $\mathcal{NP}$-hard combinatorial optimization problem.

As a challenging $\mathcal{NP}$-hard problem, the EVRP with a large number of customers is considered a complex and challenging. However, this problem can be solved effectively by a meta-heuristic algorithm. One of the most popular meta-heuristic algorithms that has been widely studied in the scientific community is the Genetic Algorithm (GA) [2, 3]. GA, inspired by the process of natural selection and genetics [4], begins with a population of individuals undergoing reproduction and mutation to create offspring. Procedure execution is repeated and terminated when a predefined condition is satisfied. Due to its powerful and easy search in use, over the past decades, GA has achieved remarkable successes to obtain an optimal or near-optimal solution on a plethora of complex real-world optimization problems, including combinatorial optimization, continuous optimization, and constrained optimization. Therefore, we propose a hybridization of a greedy search nature-inspired

✉ Vu Quoc Hien
hienvq.2000@gmail.com

Tran Cong Dao
daotc.bk@gmail.com

Huynh Thi Thanh Binh
binhht@soict.hust.edu.vn

[1] Hanoi University of Science and Technology, Hanoi, Vietnam

Published online: 14 May 2022

metaheuristic algorithm [5], namely Greedy search based Genetic Algorithm (GSGA), to solve the EVRP.

The main contributions of this work are described below:

- Propose a new greedy search algorithm GS for this problem based on the nearest neighbor clustering method, the balanced method, and the local search to achieve a reasonable solution.
- Propose a genetic algorithm GSGA implements the proposed greedy search algorithm to solve the EVRP problem efficiently. A novel encoding and decoding method are specifically designed for this problem. Furthermore, new initialization, hybridization, and mutation operators are also proposed.
- Conduct experiments with different scenarios to demonstrate the effectiveness of the proposed greedy search method on existing existing charging path optimization algorithms and of the greedy search inspired genetic algorithm for solving the EVRP.

The rest of this paper is organized as follows: Section 2 introduces the related works for EVRP, Section 3 describes the statement and formulation of the problem. The proposed greedy search algorithm (GS) and the proposed genetic algorithm (GSGA) are elaborated in Sections 4 and 5, respectively. Section 6 provides computational experiments and results. Conclusions and future extension of this research are given in Section 7.

## 2 Related works

The demand for Electric Vehicles is increasing, so the problem is to change variants and different constraints. One major downside of electric vehicles is their limited battery capacity; as a result, they are required to visit charging stations to recharge the battery. Therefore, the problem of finding optimal routing schemes of electric vehicles, i.e., EVRP, are attracting special attention from numerous of researchers and experts. Many recent related works have studied about EVRP. In [6], problem is introduced and mathematically formulated. The aim of this problem is to minimize the energy consumption of EVs. Moreover, the authors applied an ant colony (AC) algorithm based metaheuristics to solve the EVRP. In [7], Montoya et al. proposed a hybrid metaheuristic that combines simple components from the literature and components specifically designed for this problem. To assess the importance of nonlinear charging functions, they presented a computational study comparing our assumptions with those commonly made in the literature. Erdelic et al. reviewed the state-of-the-art exact, heuristic, and hybrid procedures applied for solving various EVRP variants [8]. In [9], the authors proposed a heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges.

In [10], Afroditi et al. proposed a mathematical model inspired by the known VRPTW. To solve Afroditi's model, an adaptive large neighborhood search (ALNS) algorithm enhanced with the fuzzy simulation method is proposed [11]. In the proposed ALNS algorithm, four new removal algorithms are designed and integrated to address the FEVRPTW. In [12], Yang et al. presented an EVs battery swap stations locations routing problem called BSS-EV-LRP. The problem's objective is to determine the location strategy of battery swap stations (BSSs) and the routing plan of an EV fleet within conditions that limit the battery driving range. Moreover, the authors also proposed two methods to solve the problem. The first proposed method is a four-phase heuristic called SIGALNS, in which the BSSs location stage and the vehicle routing stage are alternated iteratively. The second method is a two-phase Tabu Search-modified Clarke and Wright Savings heuristic (TS-MCWS). These proposed algorithms have been shown to be effective in finding good solutions without too many computations in medium and large instances when compared to MIP solver of CPLEX. Abdallah et al. solved this problem using a Lagrangian relaxation and proposed a new Tabu search algorithm [13]. They also presented the first results for the fully adapted Solomon instances. In the study [14], the authors pointed out the importance of the CVRP problem in practice and proposed a method based on the combination of the ant colony algorithm and the simulation annealing algorithm, called SACO. The SACO algorithm is compared with the methods based on the ant colony algorithm and shows better results than the baseline methods. However, a limitation of the proposed algorithm is the relatively long execution time, especially in cases where the size of the problem is enormous.

A problem similar to ours is studied in [15, 16], but the model formulation and optimization goals are different. In [16], the authors presented the first EVRP model to consider the vehicle load effect on battery consumption to find the optimal routing strategy with minimal travel time cost, energy cost, and number of EVs dispatched. In [15], the EVRP model with charging demands, energy consumption, range constraint, vehicle capacity constraint is presented. The model's optimization goal [15] is to find the minimum total cost, including fixed vehicle cost, travel cost, and charging cost.

In summary, with the related studies introduced, there are many variations of the VRP problem that have been proposed, including the EVRP with several models and proposed methods to solve that model. In this paper, we propose a greedy search algorithm and a genetic algorithm to persuasively solve the EVRP problem introduced in [1].

## 3 The electric vehicle routing problem

### 3.1 Problem statement

The problem is expressed with the use of a fully connected weighted graph $G = (V, E)$ where $V = \{C, O, S\}$ is a set of nodes and $E = \{(i, j), \forall i, j \in V, i \neq j\}$ corresponds to all possible arcs connecting vertices of $V$. The set $C = \{c_1, c_2, ..., c_{n_c}\}$ is a set of customer points, $O$ is the central depot, $S = \{s_1, s_2, ..., s_{n_s}\}$ is a set of charging stations. Each arc $(i, j)$ is associated with a non-negative Euclidean distance $d_{ij}$ between nodes $i$ and $j$.

The power recharging rule is defined as follows: the battery of an EV is fully charged each time the EV starts the route at the depot or after visiting a charging station. The EV can be recharged once or more at any charging stations in $S$ during routing.

The aim of the EVRP is to find a set of routes that minimize the total traveled distance that must satisfy the following conditions or constraints:

- Each EV is homogeneous and each route starts and finishes at the depot.
- Each charging station may be visited multiple times or not visited by any EV in a given strategy.
- Every customer is visited exactly once by exactly one EV.
- For every EV route, the total energy consumption does not exceed the EV's maximal battery charge level $Q_{max}$.
- For every EV route, the total demand of customers does not exceed the EV's maximal carrying capacity $P_{max}$.

Consider a feasible solution to EVRP $p = (R_1, R_2, ..., R_l)$. The objective fuction of the EVRP is below:

$$minimize: \sum_{t=1}^{l} \sum_{i,j \in R_t, i \neq j} d_{ij} x_{ij} \quad (1)$$
$$x_{ij} \in \{0, 1\}, \forall i, j \in R_t$$

where $R_t = (0, \pi_1, \pi_2, ..., \pi_k, 0)$ is a path of $EV_t$ in which $EV_t$ starts from the depot and visits customers and after recharging at charging stations, it returns to the depot; $d_{ij}$ is the Euclidean distance between nodes $i$ and $j$, $x_{ij} = 1$ if there is an arc connecting $i$ and $j$, or $x_{ij} = 0$ otherwise.

A simple EVRP example consisting of five customer $c_1$, $c_2$, $c_3$, $c_4$, $c_5$, one charging station $s_1$, and only one central depot $O$ is illustrated in Fig. 1a and b.

In Fig. 1b, a strategy without visiting any charging station and using two EVs is given. The first EV on route 1 leaves the depot $O$ to visit 3 customers $c_1$, $c_2$, $c_3$ then returns to the depot. Similarly, the second EV on route 2 leaves the depot to visit 2 customers $c_4$ and $c_5$ then returns to the depot. In the example in Fig. 1 , $p^1 = (R_1, R_2)$ with $R_1 = (O, c_1, c_2, c_3, O)$ and $R_2 = (O, c_4, c_5, O)$.

Figure 1b depicts a better strategy that involves visiting one charging station and using one EV (one route). The EV leaves the depot O to visit 3 customers $c_1$, $c_2$ and $c_3$ then visits charging station $s_1$ to fully recharge its battery. After recharging at $s_1$, the EV visits customer $c_4$ and $c_5$ before returning the depot. In this example, $p^2 = (R_1)$ with $R_1 = (O, c_1, c_2, c_3, s_1, c_4, c_5, O)$.

### 3.2 Problem formulation

The EVRP seeks to find a set of EV routes where each EV visits each customer once and only once so that the total distance is minimized. Notations of the EVRP problem are given in Table 1. Furthermore, the mathematical formulation of the EVRP in [1] is also represented below:

$$\sum_{j \in V, i \neq j} x_{ij} = \sum_{j \in V, i \neq j} x_{ji} = 1, \forall i \in C, S \quad (2)$$

$$\sum_{j \in V, i \neq j} x_{ij} \geqslant 1, \forall i \in O \quad (3)$$

$$\sum_{j \in V, i \neq j} x_{ij} \geqslant 0, \forall i \in S \quad (4)$$

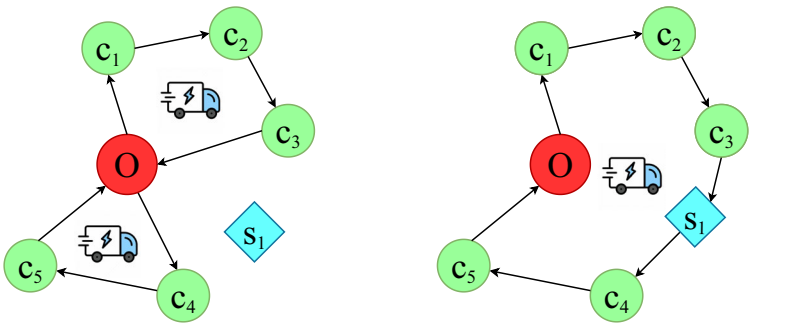



**Fig. 1** Two strategies of EVs in the EVRP

(a) Strategy without visiting any charging station (b) Strategy with visiting one charging station

**Table 1** Notations and parameters of the EVRP

| Notation, parameter | Description |
|---|---|
| $C$ | Set of customers |
| $S$ | Set of charging stations |
| $O$ | Depot |
| $n_c$ | Number of customer |
| $n_s$ | Number of charging station |
| $n$ | Size of problem$(1 + n_c + n_s)$ |
| $Q_{max}$ | Maximum battery |
| $P_{max}$ | Maximum capacity |
| $h$ | Consumption rate of the EV |
| $R_i$ | Route of the $i$-th EV |
| $l$ | Number of electric vehicles |
| $u_i$ | Remaining carrying capacity of an EV at node $i$ |
| $y_i$ | Remaining battery charge level of an EV at node $i$ |
| $b_i$ | Demand of customer $i$ |
| $\pi$ | The charging station or customer visited by the vehicle in the route |

$$\sum_{j \in V, i \neq j} x_{ij} - \sum_{j \in V, i \neq j} x_{ji} = 0, \forall i \in V \quad (5)$$

$$\sum_{i \in R_j} b_i \leq P_{max}, 0 \leq j \leq l \quad (6)$$

$$0 \leq u_i \leq P_{max}, \forall i \in C \quad (7)$$

$$y_j \leq y_i - h d_{ij} x_{ij} + Q_{max}(1 - x_{ij}), \forall i \in S, \forall j \in V, i \neq j \quad (8)$$

$$y_j \leq y_i - h d_{ij} x_{ij}, \forall i \in S \cup O, \forall j \in V, i \neq j \quad (9)$$

$$0 \leq y_i \leq Q_{max}, \forall i \in V \quad (10)$$

$$x_{ij} \in \{0, 1\}, \forall i \in V, \forall j \in V, i \neq j \quad (11)$$

Equation 2 guarantees every customer is visited exactly once by exactly one EV; (3) ensures each EV is homogeneous and each route starts and finishes at the depot; (4) implies that each charging station may be visited multiple times or not visited by any EV in a given strategy. Equation 5 establishes flow conservation by guaranteeing that at each node, the number of incoming arcs is equal to the number of outgoing arcs. Equations (6) and (7) guarantee demand fulfillment at all customers by assuring a non-negative carrying load upon arrival at any node including the depot, (8), (9), and (10) ensure that the battery charge never falls below 0, and (11) defines a set of binary decision variables which each one equal to 1 if an arc is traveled and 0 otherwise.

Figure 2 is an example solution, in which a red circle is a depot 0, green circles are customers (from customer 1 to customer 9), and blue circles are charging stations 10 and 11. The number on each edge is the distance between customer and customer or customer and charging station, and the number in red font-size inside a green circle is the remaining carrying capacity of an EV at that corresponding customer. This solution consists of three different routes: $R_1 = (0, 7, 5, 10, 2, 4, 0)$, $R_2 = (0, 9, 11, 8, 6, 0)$, and $R_3 = (0, 3, 1, 0)$. In the first route, the route length is calculated as: 4 + 8 + 7 + 4 + 4 + 6 = 33. In a similar way, the second one is 22, and the third one is 11.

In this case, we assume that this solution is valid, which means all the constraints (capacity and battery) are satisfied. Hence, the numerical fitness of the solution is 66 (= 33+22+11), which is the total length of three routes. In another case, if the solution is invalid, which exists one of the constraints is not satisfied, the fitness of the solution is set at positive fitness.

## 4 Greedy search algorithm

In this section, we present a greedy algorithm (GS) solving the EVRP problem followed by the combination of the algorithm and a genetic algorithm (GSGA) in Section 5. The flow of the GS is divided into two main periods, beginning with clustering customers into subroutes and ordering them,

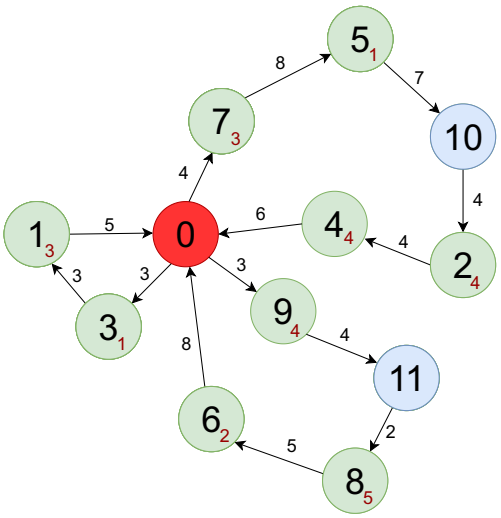



**Fig. 2** A sample solution

ending with finding the best set of charging stations for each route of each vehicle. The clustering strategy has three steps, each obtained cluster corresponding with each vehicle. Here, the evaluation function does not mention the optimal number of cars. However, the number of clusters also significantly affects the total length of all routes. In most cases, using a small number of vehicles will bring better performance than dividing customers' goods into too many ones. To do that, we use a greedy clustering method make significantly reduce the number of vehicles needed.

### 4.1 Clustering method

We implement the nearest neighbor method for clustering such that the cluster centers are uniformly distributed while the maximum power is not exceeded without exceeding the maximum carrying capacity of Ev $P_{max}$. The implementation steps as follows:

1. Randomly select a customer as seed point in a cluster
2. Add the nearest customers into that cluster until reaching max capacity without exceeding the EV's maximal carrying capacity $P_{max}$.
3. Repeat the process until all customers belong to one of the routes.

The customers, as ordered, tend to be close together, and the total capacity of each cluster will be close to the maximum capacity. Thus, the number of vehicles is significantly reduced, and the locations on each route that customer's goods are transported to will not be too far apart.

### 4.2 Balancing method

For the clustering at the before step, the customers assigned in the last route are the non-clustered customers. They are the remaining customers, so their geographical locations are not close-set. Moreover, the number of customers on this route will be less than other routes, even a single customer. To ensure the distance of customers and increase the number of customers in this, we use a balanced approach with the following steps:

1. Randomly select a customer (customer A) from the last route.
2. Select in turn the customers from the other routes such which is the closest one to A.
3. The chosen customers must satisfy the two following conditions:
   (a) The initial sum of the total capacity of the last route and the capacity of the chosen customer does not exceed the EV's maximal carrying capacity $P_{max}$.
   (b) Total capacity difference (delta) is less than before: consider two routes $R_i$ and $R_j$ ($A \in R_i$, $B \in R_j$), then $delta = |\sum_{x \in R_i} b_x - \sum_{y \in R_j} b_y|$ in which, $b_x$, $b_y$ is the demand of customer $x$ and $y$, respectively.
4. Repeat until the above conditions are not satisfied.

### 4.3 Local search

After dividing the customers into distinct clusters, the local search algorithm will rearrange them such that there are no intersecting paths. The detail of the local search is described in Algorithm 1.

**Algorithm 1** Local search 2-optimal.

**Input**: A route in a solution with $l$ and $r$ are the position of the first and the last customer in the route.
**Output**: A better route.

```
1  begin
2      Compute the distance matrix;
3      stop ← false;
4      while not stop do
5          stop ← true;
6          for i ← l to r do
7              for j ← r-1 downto i+1 do
8                  if the total length is better than before
                   then
9                      Swap the targets;
10                     stop ← false;
11                 end
12             end
13         end
14     end
15 end
```

General, this method will provide some benefits of solving the EVRP problem, such as:

- The total of an EV's capacity in a route does not exceed the EV's maximum carrying capacity $P_{max}$. Therefore, this method ensures that a solution is valid for capacity constraints.
- Each solution that is performed by the local search has an advantage in the total distance because customers on the same route are likely to be near each other.

### 4.4 Finding the charging station for earch route

After being routed, the vehicles will provide the goods in turn in the sorted order, but to ensure energy constraints is a more complicated problem. In many cases, despite the routes scheduled, it may not be possible to find a set of

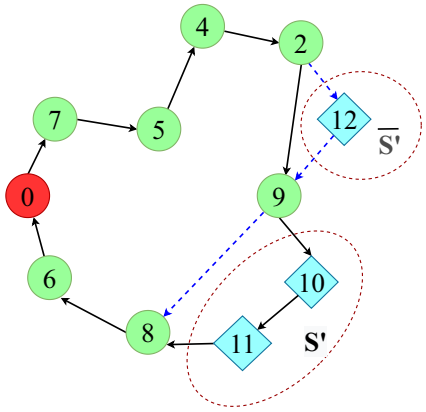



**Fig. 3** The optimize charging stations strategies of EV

charging stations that can deliver to all customers of that vehicle. To ensure enough traveling energy for each of them, we use a greedy strategy combined with the shortest path algorithm to find a set of best-charging stations among the collection of satisfying charging stations. In reality, moving between two locations requires a single charging station. However, to generalize for all cases, we found a set of charging stations between the two locations instead of using only one charging station. The detailed method is presented in Algorithm 2. For each route, the procedure is given as follows:

1. For each vehicle transfer in turn in scheduled order, if the energy necessary to go from customer $c_i$ to customer $c_{i+1}$ is not enough to move, it will go to the charging stations (not necessarily the only one) to recharge. The goal is to maximize the remaining energy of the vehicle when it reaches $c_{i+1}$. Let $e_c$ be the energy of the EV at $c_i$. And $S' = \{s_1, s_2, ..., s_k\}$ is the set of satisfy charging station, such that $s_k$ is nearest to $c_{i+1}$ and the energy consumption from $c_i$ to $s_1$ is less than $e_c$. If the vehicle cannot find such a set of charging stations, it will go back to find a station from customer $c_{i-1}$ to $c_i$. Repeat the process until it can find a satisfactory station or back to the depot.
2. For any customer $c_i$, we can prove that the number of times to find the set of charging stations from $c_i$ to $c_{i+1}$ does not exceed two because after finding the previous of them, the energy when reaching $c_i$ is maximum in any case. If we go back to the depot, there will be no way to insert charging stations to get a satisfying route and return $fitness = +\infty$.

**Algorithm 2** Insert charging stations to route $T$.

**Input**:
- A half-complete subroute $R$,
- $l, r$ are the position of the first and the last of depot in the half-complete subroute $R$,
- The set of charging stations.

**Output**: The complete route consist of set of customers from $l$ to $r$ with inserted charging stations. return $\emptyset$ if not found.

```
1  begin
2      T ← ∅;
3      for each customer c_i from l to r-1 do
4          if there is enough energy to go forward then
5              Insert c_{i+1} to tail of T;
6              Update energy;
7          else
8              while true do
9                  if found the satisfied set of charging
                   stations S' then
10                     Insert S' and c_{i+1} to tail of the
                       route T and break the while loop;
11                 else
12                     Back to find the satisfied set of
                       charging stations.
13                 end
14             end
15         end
16     end
17     return T;
18 end
```

After inserting the charging station as above, the remaining energy of the vehicle is the largest when it moves from the charging station to the next customer on the route. However, this set of charging stations is not good in terms of the distance cost of the way. Therefore, we will use one more stage to find another with shorter path costs. We can

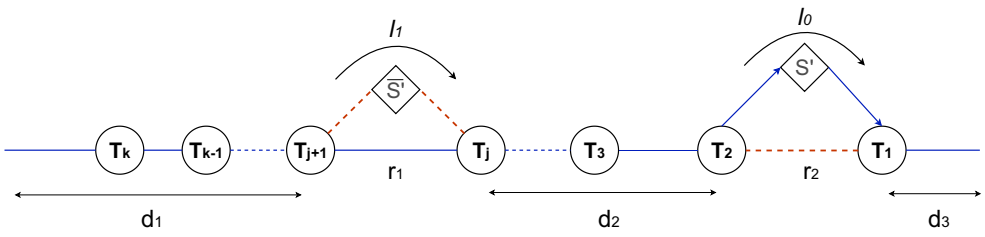



**Fig. 4** Replacing the current set of charging stations with the best other

prove that the proposed algorithm will give the optimal results if all of the considered sets have the same number of finding charging stations in a route. Suppose two different sets $S$ and $S'$ have k number of times to explore for charging stations along the way, in which $S'$ using the proposed algorithm will get better results than $S$, at least equal. The following steps are describe the process of finding a set of charging stations:

1. First, consider the route $R = \{O, c_1, c_2, .., c_{i-1}, S', c_i, ..., c_r, O\}$ and $\bar{R}$ as the reversed order route. In pre-arranged order $R$, $c_i$ is a customer right head to that the set of charging stations $S'$ and let $T$ be the set of customers $c_j$ in front of $c_i$. If the vehicle transfer with the reversed order $\bar{R}$, the energy from depot to $c_j$ is sufficient but without visiting any stations (*).

   An example is illustrated in Fig. 3, in which set $T$ including customer 8, customer 9, customer 2 and customer 4. $T$ not include the customer 5 because its energy not sufficient to travelling with the reversed order $\bar{R} = \{0 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 0\}$.
2. Let $\overline{S'} = \{s_k, s_{k-1}, ..., s_1\}$ be the set of charging stations found from $T_j$ to $T_{j+1}$ which satisfies the following conditions:

   (a) In $R$, EV has enough energy go from depot to $s_1$.
   (b) In $\bar{R}$, EV has enough energy go from depot to $s_k$.
   (c) The distance from $T_j$ to $T_{j+1}$ that through $\overline{S'}$ is the smallest.

   In Fig. 3, the tour length through the set of charging stations $\overline{S'}$ is smaller than before. After process reoptimization has been completed with **Djkstra's algorithm**, the route is $\{0 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 12 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 0\}$

As present on above, we can prove that if the set of charging stations $S'$ replaces with the set of charging stations $\overline{S'}$, the condition (*) is also satisfied.

*Proof* Let $delta_0$ be the distance from $T_1$ to $T_2$ through $S'$, and $delta_i$ is the distance from $T_i$ to $T_{i+1}$ through $\overline{S'}$. We will replace $S'$ from $T_1 \rightarrow T_2$ with the new set of charging stations $\overline{S'}$ from $T_i \rightarrow T_{i+1}$ if $delta_i$ is less than $delta_0$. If $delta_i$ is the smallest, it is considered as the best set of charging stations in the satisfied sets of charging stations. For each EV route with arranged order, the best set of charging stations will provide enough energy for the vehicle to visit all customers and return to the depot with minimum travel distance. The explanation is as following:

For two routes: ($O_l$, .., $T_k$, $T_{k-1}$,..., $T_{j+1}$, $T_j$,..., $T_2$, $S'$, $T_1$, ...$O_r$) (1) and ($O_l$, $T_k$, $T_{k-1}$,..., $T_{j+1}$, $\overline{S'}$, $T_j$,..., $T_2$, $T_1$,..., $O_r$) (2) (as in Fig. 4). Let's consider the following expressions:

$$\begin{cases} d_1 = sumDistance(O_l, ..., T_{j+1}) \\ d_2 = distance(T_j, T_2) \\ d_3 = sumDistance(T_1, ..., O_r) \\ r_1 = distance(T_{j+1}, T_j) \\ r_2 = distance(T_2, T_1) \\ l_1 = distance(T_{j+1}, \overline{S'}) + distance(\overline{S'}, T_j) \\ l_2 = distance(T_2, S') + distance(S', T_1) \end{cases}$$

- Route length (1) : $L_1 = d_1 + r_1 + d_2 + l_0 + d_3$
- Route length (2) : $L_2 = d_1 + l_1 + d_2 + r_2 + d_3$
- Let $L_2 < L_1 \Rightarrow d_1 + r_1 + d_2 + l_0 + d_3 < d_1 + l_1 + d_2 + r_2 + d_3 \Leftrightarrow l_1 - r_1 < l_0 - r_2 \Leftrightarrow delta_j < delta_i$, which completes the proof.

□

## 5 Evolutionary algorithm

In this section, a genetic algorithm combined with the proposed greedy search algorithm, called GSGA, is presented in detail. The flow of GSGA is described in Fig. 5. The algorithm includes the basic steps of a traditional genetic algorithm: individual representation, initialization, and reproduction. However, the novelty of implementing the proposed greedy search strategies into GA is clarified in the new two-level representation method, the greedy initialization method, the new hybrid and mutation operators. The details are presented in the following subsections.

### 5.1 Representation

An essential step in designing genetic algorithms is to find an appropriate chromosome representation. For this EVRP problem, a solution involves tram routes, in which the electric vehicles depart from a depot to visit customers and possibly pass through intermediate charging stations. Therefore, a reasonable solution representation requires information about customers, charging stations, and depots. This paper proposes a solution representation including new encoding and decoding methods, which are described in detail in the following subsections.

#### 5.1.1 Encoding

In the proposed encoding method, the initial representation of a solution needs only represent the customer's information and the order of the route that the customer is visiting. Thus, encoding for an individual is an array of fixed-size integers $n_c$, where $n_c$ is the total number of customers in the

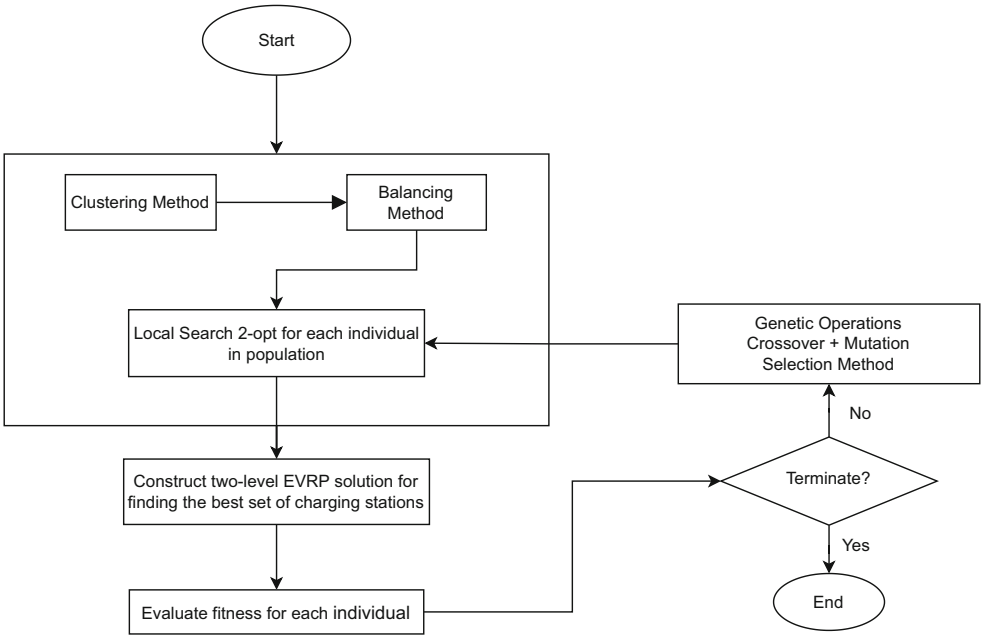



**Fig. 5** Proposed Algorithm Schema (GSGA)

problem. The value of each element corresponds with the customer node. The order of the element values in the array is the order of customers' visits in the solution.

Assume that a given EVRP model solution begins from the depot and needs to visit nine customers and go through some intermediate charging stations. The solution encoding method for this case is described in detail in Fig. 6. Figure 6a illustrates the solution mentioned above, including three routes: route 1: $0 \rightarrow 7 \rightarrow 5 \rightarrow 10 \rightarrow 2 \rightarrow 9 \rightarrow 0$, route 2: $0 \rightarrow 4 \rightarrow 11 \rightarrow 8 \rightarrow 6 \rightarrow 0$, route 3: $0 \rightarrow 3 \rightarrow 1 \rightarrow 0$. Hence, according to the proposed encoding method, a corresponding representation is shown in Fig. 6b.

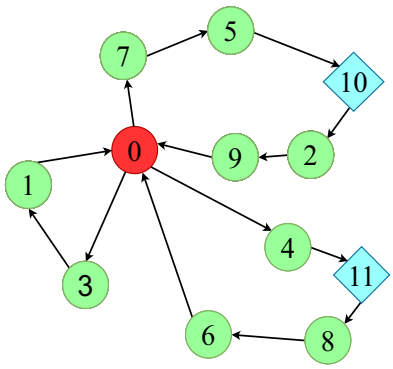
#### 5.1.2 Decoding

According to the solution encoding described above, a simple decoding method to obtain the corresponding solution is to insert possible charging stations into each existing route. Therefore, the proposed strategy of finding a set of charging stations presented in Section 4.4 is applied to build a complete path, including the order of visiting customers and charging stations. This finding charging stations strategy from Section 4.4 will ensure the set of charging stations is best from each existing route.

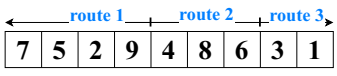
### 5.2 Initialization

According to the proposed encoding, an individual is represented by a permutation array of $n_c$ elements, where $n_c$ is the number of customers. Note that an individual encoding only needs the customer's information and the order of the route to visit that customer. Therefore, we use the greedy strategy proposed, including steps Sections 4.1, 4.2, and 4.3 in Section 4, to initialize an individual.

**Fig. 6** Encoding of a candidate solution




(a) A possible solution of the EVRP




(b) A chromosome representation of the solution

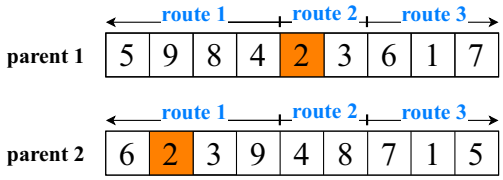



**Fig. 7** One customer is randomly selected

This greedy initialization method is expected to be more efficient than the random initialization method. Moreover, this method always generates a valid individual for capacity constraints because it applies the local search introduced in Section 4.3. This will be demonstrated in the experimental results section.

### 5.3 Crossover operator

In this paper, a new hybridization method is proposed in the GSGA. This method uses the distance heuristic, suitable for multi-vehicle problems. Details of the proposed crossover method are shown in Algorithm 3. An example of the proposed crossover operator including four steps is described as follows:

**Step 1**: Randomly select a customer (customer 2) in the parent individuals as in Fig. 7.

**Step 2**: $Sub_1$ is a set of customers in the route that contains customer 2 of $parent_1$. $Sub_2$ is a set of customers in the route that contains customer 2 of $parent_2$ as in Fig. 8.

**Step 3**: Customers who do not belong to $sub_1$ and $sub_2$ will be inherited from parent to children respectively that keep relative order.

**Step 4**: Customers in $sub_1$ and $sub_2$ will be partitioned into children satisfying the following rules: partition concatenate ($sub_2$, $sub_1$) respectively into $child_1$ and partition concatenate (reverse($sub_1$), reverse($sub_2$)) respectively into $child_2$ at the positions where elements are in $sub_1 \cup sub_2$ as in Fig. 9.

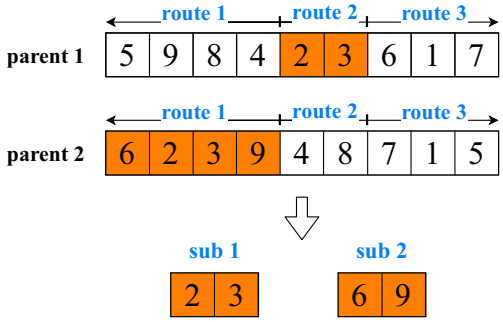



**Fig. 8** $sub1$ and $sub2$ obtained in route contain customer 2 from two parents

**Algorithm 3** Proposed crossover operator.

**Input**: Two individuals $parent_1$, $parent_2$
**Output**: $child_1$, $child_2$.

```
begin
    c_rd ← Choose a random customer;
    V1, V2 ← Set of customers which consists of c_rd
    from parent1, parent2 respectively;
    child1, child2 ← ∅;
    sub1 ← V1;
    sub2 ← (V1 ∪ V2) \ V1;
    SC1 ← Concatenate(sub2, sub1);
    SC2 ← Concatenate(Reverse(sub1),
    Reverse(sub2));
    Distribute the chromosomes from SC1, SC2 and
    parent to child1 and child2;
    return child1 and child2;
end
```

The proposed crossover operator is quite similar to two-point hybridization. However, we still use a heuristic approach by selecting a random seed point to swap the chromosomes in two different subroutes belonging to two separate parents. When exchanging chromosomes in these two subroutes, we find that the crossover operator only switches the positions of adjacent customers of different subroutes by both subroutes close to the previously selected seed point. Therefore, this will minimize the number of customers located far away from other customers in a route. However, this algorithm always ensures the abundance of new features.

### 5.4 Mutation operator

Mutation aims to introduce new genetic material into an existing individual, adding diversity to the population's genetic characteristics. Herein, we propose two types of mutation operators independently with a mutation probability for each of them:

1. Heuristic-swap mutation (HSM): Choose a random customer $c_i$ and exchange its position with the customer $c_j$ from different routes that has the shortest distance to the customer $c_i$.
2. Heuristic-move mutation (HMM): Choose a random customer $c_i$, find the customer $c_j$ from a different route that has the shortest distance to the customer $c_i$,

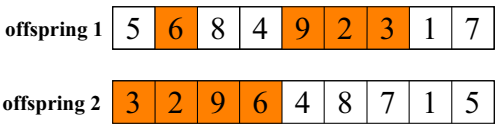



**Fig. 9** Two new children after crossover processing

and insert customer $c_j$ into the route containing the customer $c_i$.

An example of the HSM method is shown in Fig. 10. In this example, a random customer (customer 4) is chosen, and the nearest customer from other routes is customer 9. Then, two customers, 4 and 9, are swapped. Figure 11 depicts the HMM method by choosing a random customer (customer 11), finding the nearest customer from a different route (customer 2), and inserting it into the route containing customer 11.

The proposed mutation operators use heuristic methods similar to the proposed crossover operator. We can see that the mutation operators have sizable effects on the redistribution route and changing the genetic abnormalities. In the Heuristic-swap mutation operator, switching positions of two points near each other will be effective for two intersection routes. Meanwhile, the Heuristic-move mutation operator will be effective when an abnormal point needs to be moved through another route. The mutation operators described may produce the solution that violates the maximum load, we mark it as an invalid individual and return the positive infinity fitness. To evaluate clearly the effectiveness, we present the experimental result in Section 6. Overall, the use of mutation operators may be proposed for many different routing problems, and the EVRP is one of them.

### 5.5 Selection method

Natural selection is an important step in the evolution theory. Individuals struggle to survive in the wild, and only the fittest can survive. In this selection method, we choose the best individual by the wheel selection method. The details of the method are described in Algorithm 4. This method ensures that individuals with good fitness will have a high probability of choosing and vice versa. Specifically, the global best individual is added to the new population to maintain the best existing traits.

**Algorithm 4** Selection method.

**Input**: The population consists of parent and child individuals.
**Output**: New population.

```
begin
    for all individuals do
        Normalized and cumulated normalized fitness
        and compute accumulated probability array;
    end
    for all individuals do
        Choose a random probability p_r in the range
        [0, 1];
        The individual corresponding to this
        probability p_r in the accumulated probability
        array selected to the next generation;
    end
end
```

## 6 Experiment results and performance evaluation

### 6.1 Problem benchmark dataset

The EVRP benchmark set consists of two groups of problems:

1. Group 1: consists of 7 small problem instances (up to 100 customers) in which their optimal upper bound values are provided in [1].
2. Group 2: consists of 10 larger problem instances (up to 1000 customers) in which their upper bound values are not provided.

The first group of EVRP instances is generated by extending the well-known instances of the conventional vehicle routing problem from Christofides and Eilon [17].

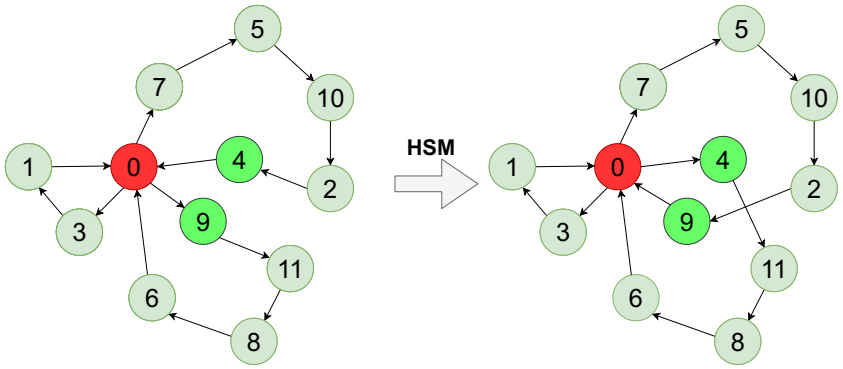



**Fig. 10** Heuristic-swap mutation

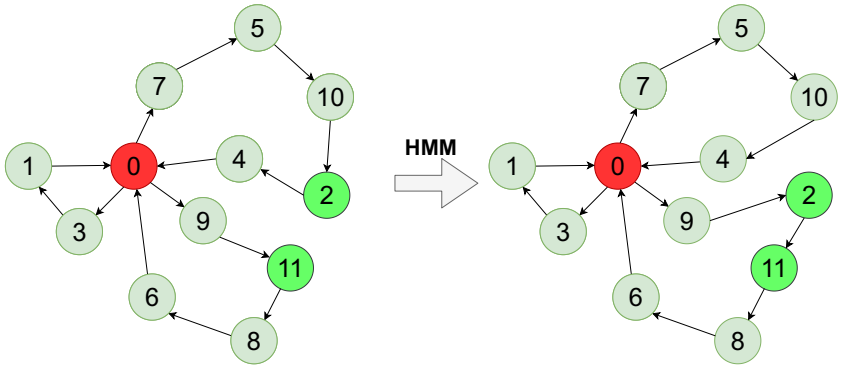



**Fig. 11** Heuristic-move mutation

The second group is an extension of the recent instances of the conventional vehicle routing problem from Uchoa et al. [18].

## 6.2 Experimental setup

To evaluate the performance of the GSGA for the EVRP, we implement the proposed algorithm on Windows 10 with 8.0 GB RAM, CPU 2.2 GHz, with a population size of 200 individuals and a mutation rate of 0.1. The maximum number of evaluations is 25000n, where n = $|C| + 1 + |S|$ is the size of the problem instance. Our source code is written in the C++ language.

In this study, we try to set the parameter for different ranges to find the best results. The fixed $POP\_SIZE$ parameter equals 200 individuals, but it is not ideal to work in large instances because the population is not diverse enough to get a good local optimal solution. However, it is good for mean and small problem size. Besides, the mutation rate also significantly affects the performance, so we experiment with different parameters (from 0.025 to 0.1) to choose the best one. The crossover rate that gives the best experimental results is 0.9, and the smaller the crossover rate, the worse the results tend to be. Details are listed in Table 2.

## 6.3 Experimental criteria

The effectiveness of the proposed algorithm is evaluated based on several criteria: speed convergence of population, best result, worst result, average result, execution time, and stabilization. Details of the criteria are presented in Table 3.

## 6.4 Experimental results

### 6.4.1 Greedy search algorithm analysis

To evaluate the effectiveness of the proposed greedy search algorithm, the given upper bound on all instances belonging to the group 1 (small) dataset are compared with the average objective values found by the greedy search algorithm on all runs. Notably, we calculate a value called approximation ratio given by the following formula:

$$Approximation\ ratio = \frac{GSA_i}{UB_i} \tag{12}$$

In the formula, $UB_i$ and $GSA_i$ are the lower bound value provided and the average objective value found by the proposed greedy search algorithm of $i$ instance. This approximation ratio is the ratio between the result obtained by the proposed greedy search algorithm and the given lower bound value. In this paper, the closer the approximation ratio is to 1, the better result is.

### 6.4.2 Greedy Seach implemented algorithms analysis

In this study, the number of runs is set to 20 (with random seed from 1 - 20). According to the execution time chart, as shown in Fig. 12, the execution time depends on the number

**Table 2** The experimental parameters

| Parameter | Description | Value |
|---|---|---|
| $POP\_SIZE$ | number of individuals | 200 |
| $p_c$ | crossover rate | 0.95 |
| $p_m$ | mutation rate | 0.1 |
| $Ev_m$ | maximum number of evaluations | 25000n |

**Table 3** Criteria for assessing the performance of the proposed algorithm over all runs

| | |
|---|---|
| Best | The best objective function value achieved |
| Worst | The worst objective function value achieved |
| Avg | The average objective function value achieved |
| Stdev | The standard deviation |
| Time | Time for execution the proposed algorithm |

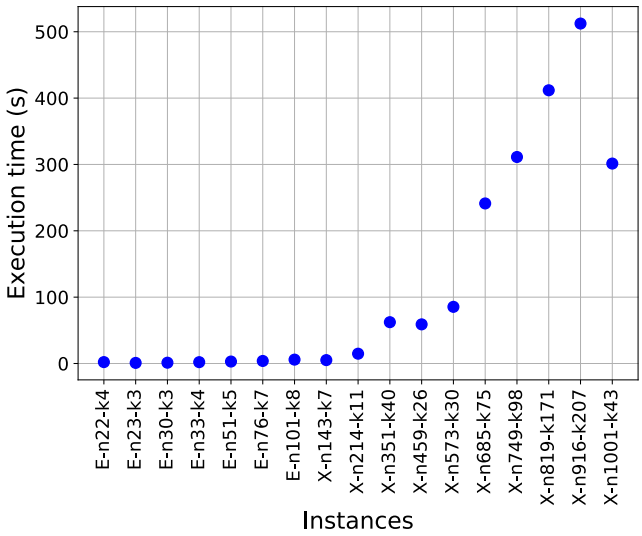



**Fig. 12** Execution time chart of the proposed GSGA algorithm over all runs in all instances

of customers, charging stations, and the number of routes. In the X-n1001-k43 instance, the execution time of the GSGA is smaller than the X-n916-k207 instance. Therefore, the number of routes in each instance proportional the execution time of the proposed algorithm. In general, the execution time of the algorithm is relatively fast, about 500 seconds for large instances.

Furthermore, we also evaluate the effectiveness of the proposed algorithm with the given upper bound value. As shown in Table 5, the experiment results show that the best average results obtained by the GSGA are lower than the upper bound value in most instances by about 1.56%.

During the experiment, the best objective values were obtained when the population size ranged from 150 to 200. With such a number of individuals, populations can adapt quickly. However, when the problem size is quite large, the number of customers is large, the population's diversity will be reduced. Comparing the Tables 4 and 5 results, the model has a much better execution time when the greedy search algorithm is implemented to GA.

**Table 4** The given upper bound values, average objective values of the proposed greedy search algorithm, and approximation ratios on 30 random runs in the small dataset

| Instance | Upper bound | Greedy search | Approximation ratio |
|---|---|---|---|
| E-n22-k4 | 384.67 | 490.21 | 1.27 |
| E-n23-k3 | 573.13 | 760.03 | 1.33 |
| E-n30-k3 | 511.25 | 626.28 | 1.22 |
| E-n33-k4 | 869.89 | 976.16 | 1.12 |
| E-n51-k5 | 570.17 | 661.01 | 1.15 |
| E-n76-k7 | 723.36 | 868.96 | 1.20 |
| E-n101-k8 | 899.88 | 1065.43 | 1.18 |

**Table 5** The upper bound values are provided and best result of GSGA

| Instance | Upper Bound | GSGA |
|---|---|---|
| E-n22-k4 | 384.67 | 384.67 |
| E-n23-k3 | 573.13 | 571.94 |
| E-n30-k3 | 511.25 | 509.47 |
| E-n33-k4 | 869.89 | 845.62 |
| E-n51-k5 | 570.17 | 542.08 |
| E-n76-k7 | 723.36 | 717.30 |
| E-n101-k8 | 899.88 | 872.69 |

### 6.4.3 Comparison with baselines

In this paper, we conduct three baseline algorithms: a normal genetic algorithm (GA), a simulated annealing algorithm combined the proposed greedy search algorithm (GSSA), and a hybrid ACO algorithm (SACO) mentioned in the related works. Notably, the baseline method GA do not use the initial balanced distribute method and only use the Dijkstra's algorithm to find the possible nearest station. GSSA also uses initial proposed methods and search energy station schema similar to GSGA. SACO is based on the framework of ACO and combined with the Simulated Annealing. The combination of the ACO and SA really produces an excellent outcome. ACO is used to find a viable and stable solution, but it is difficult to find a globally optimal solution while SA will rely on ACO's solution to find a globally optimal solution. We evaluate four algorithms based on the same duration (25000n evaluations).

Overall, GSGA performs the best for all data instances with the shortest path and average execution time. The GS algorithm can improve GA's performance by $10\% - 25\%$ on average if implemented, compared to GA's results. The GSGA algorithm incorporates the proposed GS strategy and outperforms GSSA and the state-of-the-art SACO in all instances of datasets in average objective value.Specifically, GSGA algorithm is approximately 5% better than SACO on the small dataset and 23.5% on the large dataset. Compared to GSSA, GSGA is 5.7% better on small dataset and 4% on large dataset. This is because the proposed Greedy Strategy, combined with a genetic algorithm, can significantly reduce the cost of finding optimal solutions as the data scales. Table 6's findings show the greedy search algorithm GS's potential in improving any genetic algorithm's performance for finding the optimal charging routes with GSGA remains the best performing algorithm.

Next, we run Wincoxon rank test on three different pairs: GSGA vs GA, GSGA vs GSSA, and GSGA vs SACO. Performance of the above algorithms when compared with the proposed GSGA algorithm on two type datasets for 30

**Table 6** The travel distance results obtained from the implemented GA, GSGA, GSSA and SACO (the best results are indicated in bold font)

| | GA | | GSGA | | GSSA | | SACO | |
|---|---|---|---|---|---|---|---|---|
| Instances | Best | Avg | Best | Avg | Best | Avg | Best | Avg |
| E-n22-k4 | 385.44 | 391.78 | **384.67** | **384.67** | **384.67** | 407.93 | **384.67** | 385.48 |
| E-n23-k3 | 582.60 | 583.32 | **571.94** | **571.94** | **571.94** | 636.09 | **571.94** | 582.62 |
| E-n30-k3 | 516.31 | 517.12 | **509.47** | **509.47** | 515.61 | 526.02 | 511.01 | 542.20 |
| E-n33-k4 | 860.21 | 860.92 | 844.25 | **845.62** | 845.33 | 890.81 | **840.56** | 857.86 |
| E-n51-k5 | 551.47 | 568.62 | **529.90** | **542.08** | 556.92 | 590.06 | 558.46 | 595.79 |
| E-n76-k7 | 725.84 | 742.03 | **697.27** | **717.30** | 701.87 | 745.06 | 707.29 | 773.58 |
| E-n101-k8 | 879.28 | 902.82 | **852.69** | **872.69** | 867.22 | 906.60 | 877.40 | 958.47 |
| X-n143-k7 | 16876.24 | 17389.54 | **16488.60** | **16911.50** | 16845.47 | 17496.27 | 17813.49 | 19256.56 |
| X-n214-k11 | 11794.23 | 12277.79 | **11762.07** | **12007.06** | 12116.68 | 12600.04 | 14929.50 | 15491.52 |
| X-n351-k40 | 28435.32 | 28973.42 | **28008.09** | **28336.07** | 28575.47 | 28768.73 | 36450.11 | 38427.21 |
| X-n459-k26 | 26305.41 | 27330.85 | **26048.21** | **26345.12** | 26899.03 | 27620.47 | 30114.04 | 33675.74 |
| X-n573-k30 | 55678.62 | 56810.22 | **54189.62** | **55327.62** | 57906.94 | 59878.48 | 62175.89 | 64544.27 |
| X-n685-k75 | 75065.39 | 76357.98 | **73925.56** | **74508.03** | 76029.72 | 78155.81 | 117122.33 | 120876.62 |
| X-n749-k98 | 85258.97 | 86501.16 | **84034.73** | **84759.79** | 85850.66 | 88641.28 | 126675.87 | 128481.11 |
| X-n819-k171 | 174693.48 | 175382.71 | **170965.68** | **172410.12** | 175527.73 | 177915.03 | 203541.34 | 210170.65 |
| X-n916-k207 | 361402.06 | 365416.88 | **357391.57** | **360269.94** | 367617.75 | 371027.32 | 408692.18 | 412554.09 |
| X-n1001-k43 | 79513.74 | 80918.71 | **78832.90** | **79163.34** | 80907.12 | 81569.93 | 115439.23 | 115439.15 |

independent runs in the Wilcoxon signed-rank test at alpha = 0.05 is shown in Tables 7 and 8. Where, -, ≈, + indicate that the results obtained by the base algorithms are worse, similar, and better than the proposed algorithm, respectively.

It can be observed that the results of our algorithm GSGA outperform other algorithms on both small and large datasets. Specifically, GSGA outperformed GA and GSSA in all instances; GSGA outperformed SACO in 6 out of 7 small instances, and it outperformed SACO in all instances on the large dataset.

Consequently, illustration routes of the best recorded solutions of instances are shown in Fig. 13a–d. Note that the symbols: a red circle symbol, a green triangle symbol, and a blue square symbol represent a depot, a customer, and a charging station, respectively.

**Table 7** Wilcoxon signed-rank test on small dataset

| Instances | GA | GSSA | SACO |
|---|---|---|---|
| E-n22-k4 | – | – | ≈ |
| E-n23-k3 | – | – | – |
| E-n30-k3 | – | – | – |
| E-n33-k4 | – | – | – |
| E-n51-k5 | – | – | – |
| E-n76-k7 | – | – | – |
| E-n101-k8 | – | – | – |
| Total (-/≈/+) | 0/0/7 | 0/0/7 | 0/1/6 |

### 6.4.4 Ablation studies

To further examine the effectiveness of each component in the proposed genetic algorithm (GSGA), including the initialization method, the crossver method, and the mutation method, we carry out the experiments by removing each of these components in the GSGA. Table 9 shows the experimental results with *%improv* being the relative improvement of versions GSGA-I (replacing the greedy initialization by random one) and GSGA-M (without mutation), and GSGA-C (without crossover) to the version GSGA.

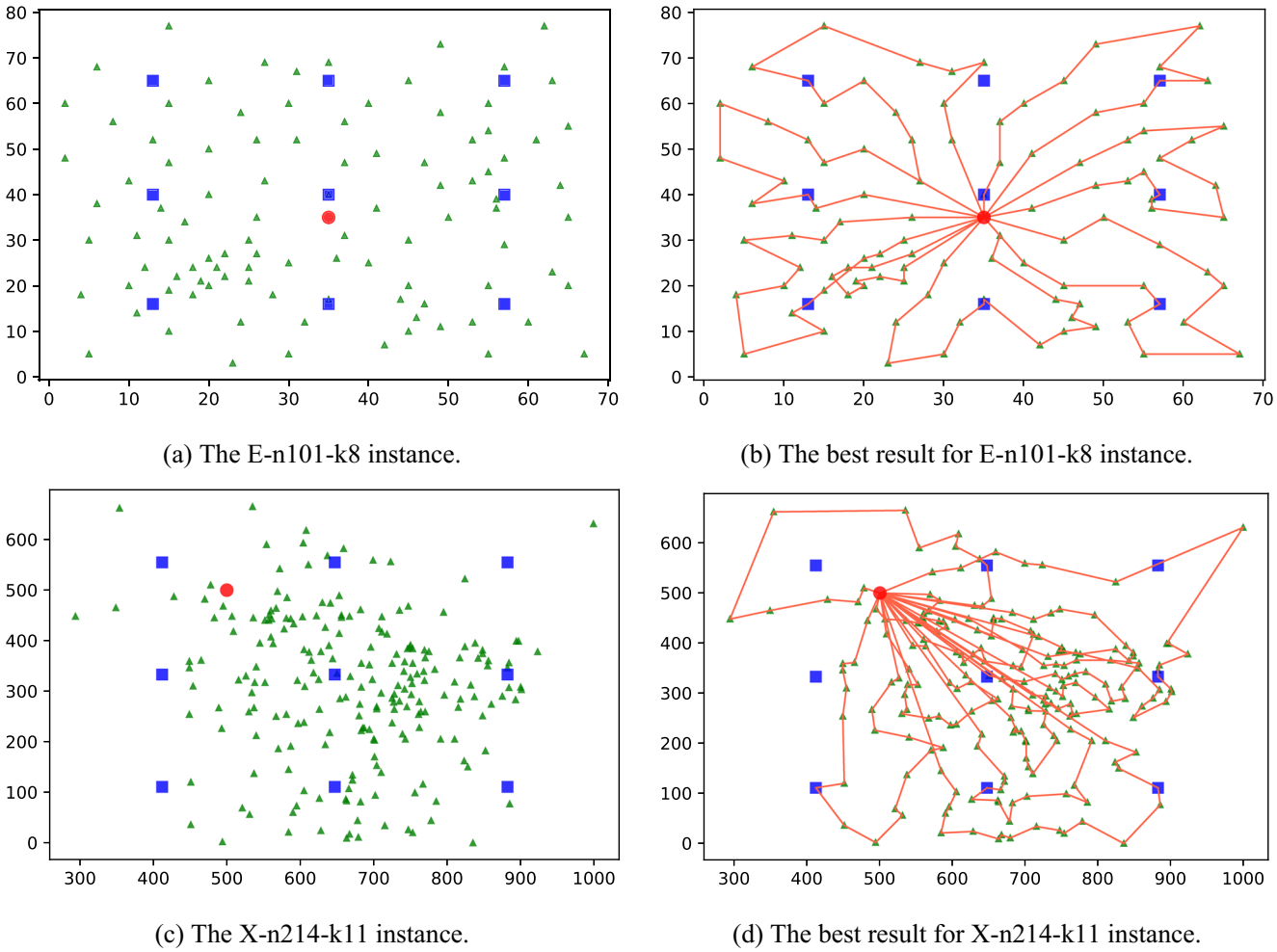
**Table 8** Wilcoxon signed-rank test on large dataset

| Instances | GA | GSSA | SACO |
|---|---|---|---|
| X-n143-k7 | – | – | – |
| X-n214-k11 | – | – | – |
| X-n351-k40 | – | – | – |
| X-n459-k26 | – | – | – |
| X-n573-k30 | – | – | – |
| X-n685-k75 | – | – | – |
| X-n749-k98 | – | – | – |
| X-n819-k171 | – | – | – |
| X-n916-k207 | – | – | – |
| X-n1001-k43 | – | – | – |
| Total (-/≈/+) | 0/0/10 | 0/0/10 | 0/0/10 |

**Table 9** Ablation analysis of each component in the proposal: initialization, crossver, mutation

| Instances | | GSGA-I | GSGA-C | GSGA-M |
|---|---|---|---|---|
| Small | E-n22-k4 | −10,76 % | −2,14 % | −1,56 % |
| | E-n23-k3 | −0,69 % | −0,40 % | −0,16 % |
| | E-n30-k3 | −10,02 % | −0,54 % | −0,30 % |
| | E-n33-k4 | −3,89 % | −0,12 % | −0,12 % |
| | E-n51-k5 | – | −3,49 % | −2,79 % |
| | E-n76-k7 | – | −2,22 % | −4,84 % |
| | E-n101-k8 | – | −1,69 % | −4,68 % |
| Large | X-n143-k7 | – | −1,74% | −4,31% |
| | X-n214-k11 | – | −0,77 % | −6,62 % |
| | X-n351-k40 | – | −1,76 % | −5,48 % |
| | X-n459-k26 | – | −2,10 % | −6,30 % |
| | X-n573-k30 | – | −2,01 % | −5,10 % |
| | X-n685-k75 | – | −1,39 % | −4,65 % |
| | X-n749-k98 | – | −1,39 % | −3,92 % |
| | X-n819-k171 | – | −1,37 % | −2,20 % |
| | X-n916-k207 | – | −1,31 % | −1,93 % |
| | X-n1001-k43 | – | −1,47 % | −5,04 % |

The results are *%improv* averaged over 10 runs of the GSGA-I, GSGA-M, GSGA-C algorithms to the GSGA algorithm



(a) The E-n101-k8 instance.

(b) The best result for E-n101-k8 instance.

(c) The X-n214-k11 instance.

(d) The best result for X-n214-k11 instance.

**Fig. 13** Routes of the best recorded solutions of instances in the benchmark datasets

First, we observe that greedy initialization gives better solutions than random initialization on all datasets. The result is inadequate when not using the greedy initialization strategy, even finding invalid solutions (except for four small instances). The reason is that the greedy initialization method can produce valid solutions and the solutions are superior in quality to the random initialization.

Second, we observe that the results of GSGA-C (without crossover) are also not good. Besides, GSGA-M (without mutation) results are much worse than GSGA-C on all large instances. As a result, both crossover and mutation operators play a significant role in improving the solution's quality, in which mutation is more efficient than the crossover.

## 7 Conclusion

This paper presents an Electric Vehicle Routing Problem (EVRP) and proposes an efficient algorithm, namely GSGA, based on a genetic algorithm and greedy search algorithm to solve the EVRP. In our approach, the new solution representation, including the new encoding and decoding method, is proposed for GSGA. Furthermore, this paper proposes the greedy initialization, the new crossover, and mutation operators for the problem. The proposed algorithm is evaluated on EVRP benchmark instances in EVRP Competition at WCCI 2020. The experimental results show that when implemented to a genetic algorithm, our greedy search algorithm GS can find much better clustered charging routes with more optimal travel distances for vehicles.

For the future, we plan to further study the proposed GSGA on more variations of electric vehicle routing problems. The design of multifactorial evolutionary algorithms to solve vehicle routing problems is also a promising research direction.

**Acknowledgements** This work is supported by the Ministry of Education and Training of Vietnam under Contract Number B2021-TTB-01. Tran Cong Dao is funded by Vingroup JSC and supported by the Master, PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), Institute of Big Data, code VINIF.2021.ThS.BK.08.

## References

1. Mavrovouniotis M, Menelaou C, Timotheou S, Panayiotou C, Ellinas G, Polycarpou M (2020) Benchmark set for the ieee wcci-2020 competition on evolutionary computation for the electric vehicle routing problem. KIOS CoE, University of Cyprus, Cyprus, Tech. Rep
2. Thanh PHAMDinh, Binh HTT, Lam BT (2013) A survey on hybridizing genetic algorithm with dynamic programming for solving the traveling salesman problem. In: 2013 International conference on soft computing and pattern recognition (SoCPaR). IEEE, pp 66–71
3. Thanh PD, Binh HTT, Lam BT (2015) New mechanism of combination crossover operators in genetic algorithm for solving the traveling salesman problem. Knowl Syst Eng 326(1):367–379
4. Back T, Hammel U, Schwefel H-P (1997) Evolutionary computation: Comments on the history and current state. IEEE Trans Evol Comput 1(1):3–17
5. Yang X-S (2010) Nature-inspired metaheuristic algorithms. Luniver press
6. Zhang S, Gajpal Y, Appadoo SS, Abdulkader MMS (2018) Electric vehicle routing problem with recharging stations for minimizing energy consumption. Int J Prod Econ 203:404–413
7. Montoya A, Guéret C, Mendoza JE, Villegas JG (2017) The electric vehicle routing problem with nonlinear charging function. Transp Res B Methodol 103:87–110
8. Erdelić T, Carić T (2019) A survey on the electric vehicle routing problem: variants and solution approaches. J Adv Transp
9. Felipe A, Ortuño MT, Righini G, Tirado G (2014) A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. Transp Res Part E: Logist Transp Rev 71:111–128
10. Afroditi A, Boile M, Theofanis S, Sdoukopoulos E, Margaritis D (2014) Electric vehicle routing problem with industry constraints: trends and insights for future research. Transp Res Procedia 3:452–459
11. Zhang S, Chen M, Zhang W, Zhuang X (2020) Fuzzy optimization model for electric vehicle routing problem with time windows and recharging stations. Expert Syst Appl 145:113123
12. Yang J, Sun H (2015) Battery swap station location-routing problem with capacitated electric vehicles. Comput Oper Res 55:217–232
13. Abdallah T (2013) The plug-in hybrid electric vehicle routing problem with time windows. Master's Thesis, University of Waterloo
14. Sun X, Fu Y, Liu T (2017) A hybrid aco algorithm for capacitated vehicle routing problems. In: 2017 IEEE 2nd advanced information technology, electronic and automation control conference (IAEAC). IEEE, pp 510–514
15. Shao S, Guan W, Bi J (2017) Electric vehicle-routing problem with charging demands and energy consumption. IET Intell Transp Syst 12(3):202–212
16. Lin J, Zhou W, Wolfson O (2016) Electric vehicle routing problem. Transp Res Procedia 12:508–521
17. Christofides N, Eilon S (1969) An algorithm for the vehicle-dispatching problem. J Oper Res Soc 20(3):309–318
18. Uchoa E, Pecin D, Pessoa A, Poggi M, Vidal T, Subramanian A (2017) New benchmark instances for the capacitated vehicle routing problem. Eur J Oper Res 257(3):845–858