Report of solution to a battery-electric vehicle (BEV) variant of CVRP

Jinhan Gao

Submitted: Oct 22, 2023

Abstract

A solution for a simplistic battery-electric vehicle (BEV) variant of CVRP is presented in this report. In the solution, a new destroy heuristic method is proposed and implemented along with several Destroy Operators and Repair Operators in the ALNS algorithm.This report focuses on the design, implementation, and analysis of the results of the solution.

1 Design Of The ALNS Framework

According to the pre-assignnment, the solution uses the ALNS algorithm and the corresponding Python library. The ALNS framework is an extension of the Large Neighborhood Search (LNS) algorithm which employs a set of destroy and repair operators, allowing it to iteratively destroy parts of the current solution and then repair and improve it.

One of the keys to solving CVRP using ALNS is to design and implement problem-specific destroy and repair operators.The operators used in experiments are described below, and the final solution consists some of them after comparison.

1.1 Destroy Heuristics

1. Random Removal

   Random Removal removes a random portion of customers (user-defined number).

2. Worst Removal

   If the cost of the route drops a lot after removing a customer, it is considered that the customer brings a lot of cost. After sorting this change (the difference in cost after removing this customer) from largest to smallest, a customer is randomly selected (at the front of the queue) and removed.

3. Slack-induced Substring Removal

   SISR obtains state-of-the-art results using a destroy operator that, instead of removing random customers, removes partial routes (called strings) that are all located near each other.

4. Long Distance Removal

   The Long Distance Removal Heuristic is a new Destroy Heuristics proposed by this report, which is inspired by Forward Load Removal Heuristic proposed by Yu [1] and the Worst Removal Heuristic.

In the context of the problem discussed in this report, cost is not equal to distance but is still a unitary function of distance, so the main idea of this method is to destroy the paths over long distances. Unlike the Worst Removal Heuristic, this method does not consider the change of cost after removing a certain customer, but only removes the front point of the long distance edge (the point that is close to the depot in an edge), separates the original path, and then finds a connection that is closer in distance through the repair operation. The Figure 1 shows an example of how to choose the customer to remove. Assume that the edge between the customer A and B is very long, which resulting the customer B will be removed. Then, it is supposed to find routes for customer A and B with small distances. The alogrithm 1 represents the pseudocode of this removal method.

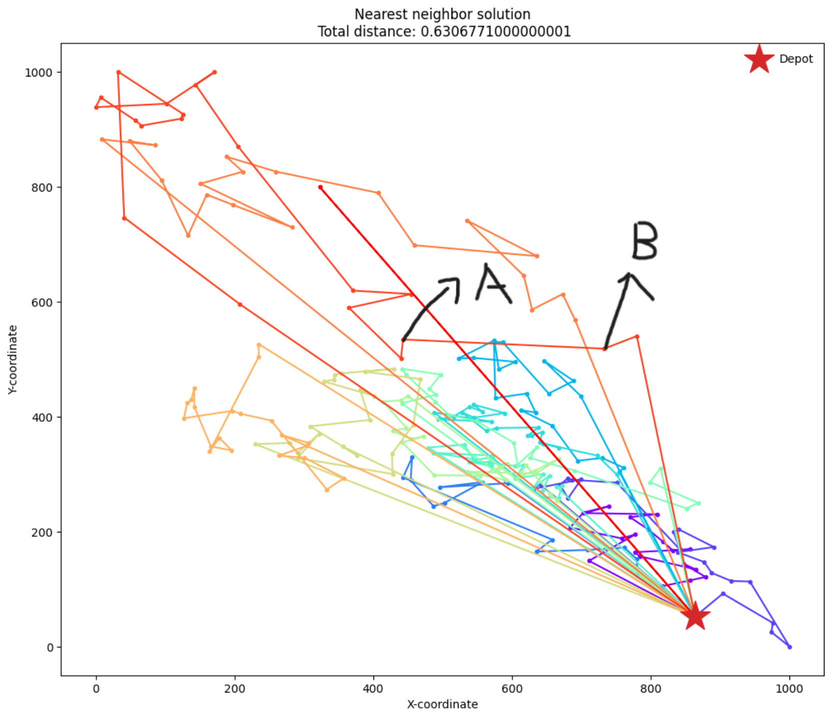


Figure 1: Illustration of a long-distance edge in the initial solution.

By looking at the initial solution, it is found that the route with the highest number of customers has several long distance edges, which is an unreasonable plan. Therefore, at each round of the loop, the length of an edge in the route with the most customers is randomly chosen as a criterion, and any front points (customers) with edges longer than this length are removed.

Algorithm 1 Long Distance Removal Heuristic.

```
Require: $Routes \leftarrow State.routes$
 1: $RouteWithMostCustomers \leftarrow CountNumOfCumstomer(Routes)$
 2: $RandomNum \leftarrow np.random.randint(0, len(RouteWithMostCustomers) - 1)$
 3: $ChosenCustomer \leftarrow RouteWithMostCustomers[RandomNum]$
 4: $DistanceCriterion \leftarrow Distance[ChosenCustomer][ChosenCustomer + 1]$
 5: $RouteIndex \leftarrow 0$, $CustomerIndex \leftarrow 0$
 6: for $RouteIndex \leq len(Routes)$ do
 7:     for $CustomerIndex \leq len(Routes[RouteIndex])$ do
 8:         if $Distance[CustomerIndex][CustomerIndex + 1] \geq DistanceCriterion$ then
 9:             $Del(CustomerIndex)$
10:         end if
11:         $CustomerIndex \leftarrow CustomerIndex + 1$
12:     end for
13:     $RouteIndex \leftarrow RouteIndex + 1$
14: end for
15: return $State$
```

1.2 Repair Heuristics

1. Greedy Insertion Heuristic Find a new, least-cost route for each customer in the order in which they were previously removed.

2. Regret-k Insertion Heuristic Find the route with the smallest cost for each unassigned

Baseline	Change of battery State-of-Health
Simple Method	0.6212446
Best Solution	0.6111875

Table 1: Two baselines for the experiment.

customer in order of regret value. Regret value refers to the difference between the cost of the optimal route and the cost of the second best route.

2 Result

2.1 Baselines

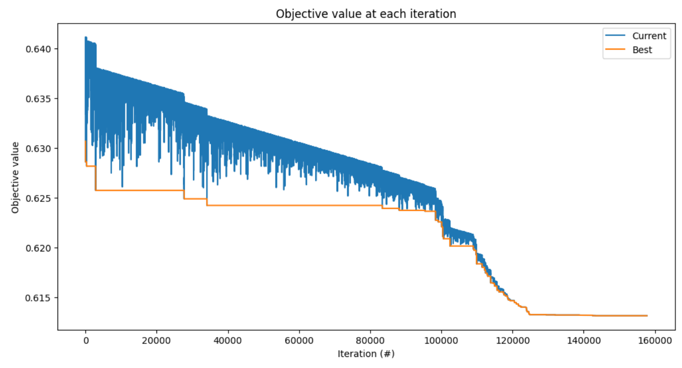
There are two baselines chosen for the experiment. The strong baseline is the objective value of the best solution and the weak baseline is the objective value of a simple method. These two baselines were used to measure the performance of the developed solution. The wek baseline is the result by using the Random Removal and Greedy Repair and iterated for 20000 times. The change of battery State-of-Health value of two baselines are showd in the table 1.

2.2 Best Solution

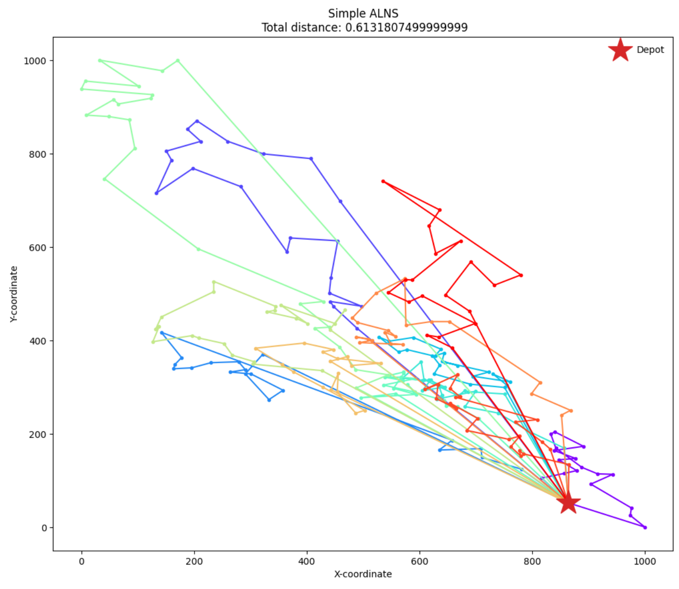
After extensive testing, Long Distance Removal can replace Worst Removal in the calculation of ALNS and performs better. The best results being obtained by applying Long Distance Removal, String Removal, Random Repair, and Greedy Repair. The implementation of the regret k repair method has yet to be optimized, and it now requires a significant amount of computational resources to perform this operation, therefore it is not used in the final results. The table 2 presents the comparison between my solution and baselines. The Figure 2 shows the objective value at each iteration and the vislualization of the final

Solutions	Change of battery State-of-Health
Simple Method	0.6212446
Best Solution	0.6111875
My Solution	0.6131807

Table 2: A comparison between my solution and baselines.



(a)



(b)

Figure 2: SOlution Visualization.

solution.

3 Literature Cited

References

[1] Zixuan Yu, Ping Zhang, Yang Yu, Wei Sun, and Min Huang. An adaptive large neighborhood search for the larger-scale instances of green vehicle routing problem with time

windows. *Complexity*, 2020:1–14, 2020.