Large Scale Robust Energy Capacitated Vehicle Routing Problem (ECVRP) Solved with Clustering Heuristics

Mark Pustilnik and Francesco Borrelli

July 2023

1 ABSTRACT

The Routing problem is a well known and hard to solve problem in practice. The recent developments in car technology, specifically the introduction of Electric Vehicles (EV) introduces a new type of complexity. The limited battery capacity of EV makes the Energy Capacitated Vehicle Routing Problem (ECVRP) - a finite number of customers, each with its own demand, have to be serviced by a electric fleet trucks while ensuring that none of the truck would run out of energy. This paper suggests to solve the ECRP exactly with a MIP and by clustering Heuristics to speed up the solution. Using the fast solutions achieved by a clustering heuristics, a closed loop planing can be done by re-planning the tours on every time step.

2 Introduction

The Vehicle Routing problem (VRP) is a well-established combinatorial optimization challenge with significant practical implications in transportation and logistics management - Visiting N nodes by M cars such that every car starts and finishes at K central nodes (Depots) and each node is visited a single time while minimizing a cost function. The cost function can be a simple total tour time, distance travelled by the trucks or can be something more complex like the total financial cost (which is the function of time and energy consumption). The ECVRP is an extension of the RP - in addition to the constraints of the RP problem, the energy constraints are added. Every drive between customers takes some amount of energy that is subtracted from the vehicle's state-of-charge(SOC). The vehicle SOC must always be positive, and at the end of the tour, each truck has to return with enough energy so that it can be charged fully overnight for the next day tour. There are Charging Stations (CS) that can be visited during the tour to charge the vehicle battery with a known time charging profile that can be nonlinear. The cost function to minimize is a function of the total time travelled by all trucks and total charging time. In addition, each customer has some demand to be fulfilled by the truck and the sum of demands of all

customers in a single tour can't exceed the capacity of the truck. Additional constraint may be added such as - limit the maximal number of nodes visited by a single car. The ECVRP, similarly to the classic RP problem, is NP-hard and is impractical to solve to optimally for large scale problems [1]. There are many approximated methods for solving this problem ([2, 3]). Another important extension to consider is the stochastic nature of the problem. As far as the authors know, the ECVRP has never been defined in it's stochastic form which take into account the uncertain properties of the parameters defining the problem - the time and energy it takes to travel between customers. The solution of the routing problem should take into account the probability properties of the each tour. Meaning, if the energy consumption of travelling between some 2 customers has a small mean but high variance, we should take it into account when calculating the optimal tour. By assuming that the energy consumption of travelling on an edge between 2 customers has some distribution (for simplicity and practicality we assume normally distributed) we can make sure, that the probability of any vehicle's SOC to reach the minimum allowed is less than a desired probability - this is the Robust Electric Capacitated Vehicle Routing Problem (RECVRP). The main contribution of this paper is as follows: 1) develop the Mixed Integer Program (MIP) that potentially can solve the RECVRP problem optimally, 2) develop a heuristic method that first clusters the data into M groups and than solves the problem for each group by assigning a truck to serve the customers in this group - this becomes M Robust Electric Capacitated Travelling Salesman Problem (RECTSP), each one smaller than the original problem. Choosing the clustering method is non trivial [4]. There has been attempt to solve the VRP with clustering hueristics [5, 6, 7]. Figure 1 shows an example of a RECVRP tour. It shows the customers on a graph and the tour of each truck. Each edge has the mean and standard deviation of the Time of travel in red, and the mean and standard deviation of the energy consumption in blue. It also shows the nominal (solid) and robust (dashed) SOC of each vehicle during the tour. This cluster algorithm is fast and produces good quality tours (compared to other algorithms) - this can be used for a closed loop control of the tour, in case the conditions is changes along the tour in real time.

3 Problem Statement

The problem is described with a fully weighted connected graph $G = (V, E)$, where $V = (D, C, S)$ is all the nodes on the graph that represent the depots $D = \{d_1, ..., d_{n_d}\}$, the customers $C = \{c_1, ..., c_{n_c}\}$ and charging stations $S = \{s_1, ..., s_{n_s}\}$ (S may contain duplicate virtual charging station to allow more than a single visit in the same charging station). Every depot has a known number of trucks m_i for $i \in D$, where $M = \sum_{i \in D} m_i$ is the total number of vehicles. Each edge on the graph that connects 2 nodes is the set $E = \{(i, j) \forall i, j \in V, i \neq j\}$. Every tour has to start and end from the same depot. The maximal Customers any vehicle can visit along the tour is defined by $N_{max} \in \mathbf{N}$. Every customer has to be visited once and only once. Every charging station can be visited as many times as needed. Every edge $(i, j) \in E$ has an associated non-negative time and energy consumption associated with travelling along the edge represented by the mean $(T^{\mu}_{i,j}, E^{\mu}_{i,j})$ and variance $(T^{\sigma}_{i,j}, E^{\sigma}_{i,j})$. Every

customer has a demand to be fulfilled, q_i for $i \in C$, and the sum of demands along any tour has to be less or equal to the capacity of the truck (Q). Each truck starts the tour with fully charged battery and it has to complete the tour in a probability greater than the design goal P_E - meaning, the SOC will not be less the minimum allowed at any point along the tour with probability greater that P_E. Each truck may visit any charging station as many times as needed and charge its battery to any SOC up to full SOC according to some charging profile. Any time spent in a charging station is part of the tour time. The objective function is to find the tour for each truck that minimizes the total tour time with a probability of P_T. This means that if we produce a large number of randomized instances for each feasible tour of the problem, we would like to choose the tour that will be the fastest with a probability greater than P_T.

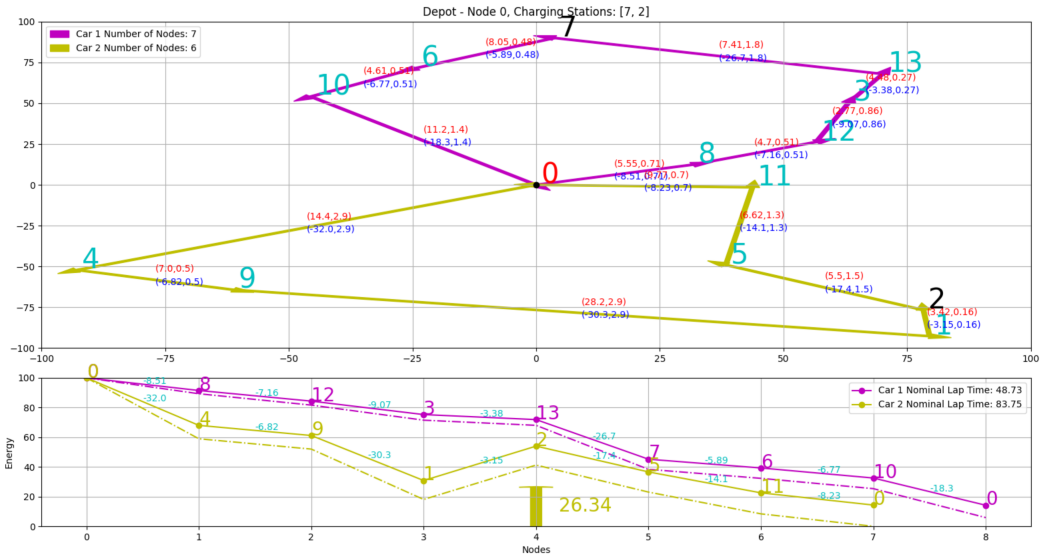

Figure 1: Example of a Energy Constrained Routing Problem with 20 Nodes and 2 Vehicles

4 Mixed Integer Program Formulation

The ECVRP can be described as a Mixed Integer Program (MIP) and solved using any Mixed integer solver. To formulate the MIP - First, define the decision variable matrix $X \in \mathbf{B}^{NxN}$. Every term in the matrix, $x_{i,j}$ is a Boolean variable which is 1 if a tour goes from node i to j, and 0 otherwise. The first constraint of the MIP is that the number of exits leaving depot i is as the number of trucks in this depot:

$$\sum_{j \in (C,S)} x_{i,j} = m_i, \forall i \in D \tag{1}$$

The second constraint ensures that every customer has single entrance:

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

The third set of constraints ensures that every charging station has no more than a single entrance:

$$\sum_{j \in S, i \neq j} x_{i,j} <= 1, \quad \forall i \in V \tag{3}$$

To ensure that every node has the same number of exits as entrances and eliminate self loop:

$$\begin{aligned} x_{i,i} &= 0, \quad i \in V \\ \sum_{j \in V,, i \neq j} x_{i,j} &= \sum_{j \in V,, i \neq j} x_{j,i}, \quad \forall i \in V \end{aligned} \tag{4}$$

Constraints (1) to (4) are not enough to define a valid tour. Sub-tour elimination constraints are needed. There are 2 main types of sub-tour elimination types. The first one is known as the Miller–Tucker–Zemlin (MTZ) formulation, and the second one is the Dantzig–Fulkerson–Johnson (DFJ) formulation. Both formulations are valid but each has it's own pros and cons. The MTZ formulation is defined by $\mathcal{O}(N^2)$ constraints on a N auxiliary variable $u_i \in \mathbf{R}$:

$$u_i = 1 + i * (N_{max} + 1), \quad i \in 0, \ldots, |D| - 1 \tag{5a}$$
$$u_j \geq u_i + 1 - (U_{max} - 1) \cdot x_{i,j}, \quad i \in V, j =\in (C, S), \quad i \neq j \tag{5b}$$
$$u_i \geq 1 + j \cdot (1 + N_{max}) - U_{max} \cdot (1 - x1, j), \quad i \in (C, S), j \in 0, \ldots, |D| - 1 \tag{5c}$$
$$u_i \leq (1 + j) \cdot (1 + N_{max}) + U_{max} \cdot (1 - x1, j), \quad i \in (C, S), j \in 0, \ldots, |D| - 1 \tag{5d}$$

Where U_{max} is a large number that makes (5) trivial if $x_{i,j} = 0$. The concept of this formulation is that u_i is increased every time a node is visited. When $x_{i,j} = 0$ the constraint becomes trivial, when $x_{i,j} = 1$ the constraint makes sure the tour is Hamiltonian. (5a) initializes the values for all vehicles leaving node i, (5b) describes the dynamics of u_i, (5c)and (5d) make sure that the returning vehicle originated from the same depot. The MTZ formulation is a practical formulation to use for large scale problem. This formulation is essentially a integrator that increases by 1 every time a node is visited along the tour. This technique will be used later again. The Second formulation (DFJ), makes the sub-tour eliminations in more direct approach. For any unwanted sub-tour, there is a constraint that eliminates it. There are 2 types of tours eliminations. The first, any tour that doesn't start from the depot is eliminated. For example, the tour $(4) - (5) - (6)$ should be eliminated and it is done by:

$$x_{4,5} + x_{5,6} + x_{6,4} < 3 \tag{6}$$

These set of constraints are in general written as:

$$\sum_{i \in Q} \sum_{j \neq i, j \in Q} x_{ij} < |Q| - 1, \quad \forall Q \subsetneq \{C, S\} \quad 2 \leq |Q| \leq N/2 \tag{7}$$

The second type is any tour that is longer than allowed:

$$\sum_{i \in Q} \sum_{j \neq i, j \in Q} x_{ij} < |Q| - 1, \quad \forall Q \subseteq V, \quad N_{max} + 1 < |Q| \tag{8}$$

The DFJ formulation that is defined by (7)-(8) is a stronger formulation than the MTZ for this problem, but it's also exponential in size and therefore impractical to used for large scale problem. In this paper the MTZ formulation is used. The next constraint set will deal with is the energy constraints of each vehicle. Defining the auxiliary variables $\{\epsilon_i\}_{i=0}^{N}$ and $\{e_i\}_{i=0}^{N}$ as the battery SOC when entering and exiting node i, respectively. $\epsilon_i = e_i$ for any node that is not a charging station and $\epsilon_i \leq e_i$ for $i \in S$. The requirement of the problem is:

$$\mathbb{P}(\epsilon_i \geq SOC_{min}) \geq P_E \quad \forall i \in V \tag{9}$$

Where SOC_{min} is the minimal SOC allowed. This can be some constant for all nodes ($SOC_{min} = 0$ is the most trivial) or it can be different for each node (for example, the energy it takes to return to the depot) Since the energy consumption of travelling on any edge between nodes is a normally distributed variable and independent on other edges then the total energy consumption to travel up until some node along the tour is also a normally distributed with the following properties:

$$\epsilon_k \sim \mathcal{N}(\mu = \sum_{(i,j) \in W_k} E_{i,j}^{\mu}, \quad \sigma^2 = \sum_{(i,j) \in W_k} (E_{i,j}^{\sigma})^2) \tag{10}$$

Where W_k represent the set of edges that lead from the depot to node k. To ensure (9) the following constraints need to be met:

$$\epsilon_j = SOC_0, \quad \forall j \in D \tag{11a}$$
$$e_j^{\sigma} = 0, \quad \forall j \in D \tag{11b}$$
$$ef_j^{\sigma} = 0, \quad \forall j \in D \tag{11c}$$
$$\epsilon_j \geq e_i + E_{i,j}^{\mu} - E_{max} \cdot (1 - x_{i,j}), \quad i, j \in V, \quad i \neq j \tag{11d}$$
$$(e_j^{\sigma})^2 \geq (e_i^{\sigma})^2 + (E_{i,j}^{\sigma})^2 - (x_{i,j} - 1) \cdot E_{max}, \quad i, j \in V \quad i \neq j \tag{11e}$$
$$(ef_i^{\sigma})^2 \geq (e_i^{\sigma})^2 + (E_{i,j}^{\sigma})^2 - (x_{i,j} - 1) \cdot E_{max}, \quad i \in (C, S), j \in D \tag{11f}$$
$$e_i = \epsilon_i, \quad i \in \{D, C\} \tag{11g}$$
$$e_i \geq \Phi(P_E) \cdot e_i^{\sigma} + SOC_{min}, \quad i \in (C, S) \tag{11h}$$
$$e_i \geq \Phi(P_E) \cdot ef_i^{\sigma} + SOC_{min} + E_{i,j}^{\mu} - E_{max} * (1 - x_{i,j}), \quad i \in (C, S), j \in D \tag{11i}$$

Where E_{max} is a large number that makes the constraint trivial if $x_{i,j} = 0$. $\{e_i^\sigma | i \in V\}$ is the standard deviation of the energy consumption from the depot to node i, $\{ef_i^\sigma | i \in V\}$ represent the standard deviation of the energy consumption after returning to the depot and $\Phi(\cdot)$ is the standard normal CDF function. Equations (11a)-(11c) are variable initialization for the energy and energy consumption standard deviation at the depots. Equation (11d) is the SOC change along the tour when moving along edges. Equation (11e) is the summation of the energy consumption covariance along the tour. Equation (11f) is the summation of the energy consumption covariance along the tour for the return to depot edge. Equation (11g) represents the fact that the SOC when entering and exiting a non charge station node are equal. Equations (11h) and (11i) enforces that the SOC when entering a node and returning to the depot would be more than SOC_{min} at a desired probability, respectively.

The connection between the entering SOC and exiting SOC at a charging station node depends on the charging model. For a linear charging model:

$$\begin{aligned} &e_i = \epsilon_i + \tau_i \cdot c, \quad i \in S \\ &\tau_i \geq 0, \quad i \in V \end{aligned} \tag{12}$$

Where $\{\tau_i\}_{i=0}^n$ is the charging time at each node. This formulation (11)-(12) is essentially an integrator on the SOC state of each truck. The charging model can be written as a more realistic non linear charging model where the charging rate is SOC dependent, as presented in [8] for piece-wise liner function. The last set of constraints are the load capacity. As mentioned, each customer has a some known demand and each vehicle has a known maximal capacity. The sum of loaded of all customers visited by the same vehicle should be less than the vehicle maximal capacity. This can be guaranteed by summing the demand along every vehicle tour and constraining the sum to be less than allowed. As before, this can be achieved by using the impractical DFJ formulation or the MTZ formulation. Introducing the variable $\{c_i \| i \in V\}$ which represent the sum of demand from the depot to node i along the tour:

$$c_i = 0, i \in D \tag{13a}$$

$$c_j \geq c_i + Q_j - Q_{max} \cdot x_{i,j}, \quad i \neq j, \quad i,j \in V, i \neq j \tag{13b}$$

$$q_i <= Q, i \in C \tag{13c}$$

Where, Q_max is a large number that makes (13b) trivial. (13a) is the initialization of the dispatched load in the depots, (13b) describes the accumulated dispatched load after visiting node i and (13c) constraints the dispatched load to be less than the vehicle capacity.

Finally, the cost function to be minimized is the tour total time (including the travel and charging time). The total tour time is the sum of the travelling time along all the edges in the tour. Since, the travelling time along any edge is normally distributed and independent of all other edges, the total travelling time is therefore a normally distributed variable with the following properties:

$$T \sim \mathcal{N}(\mu = \sum_{i,j \in V, i \neq j} x_{i,j} \cdot T_{i,j}^\mu, \quad \sigma^2 = \sum_{i,j \in V, i \neq j} (x_{i,j} \cdot T_{i,j}^\sigma)^2) \tag{14}$$

To minimize the total tour time at a desired probability, the cost function is:

$$\min_{X,\epsilon,e,\epsilon,ef,e^{\sigma},ef^{\sigma},\tau} \sum_{i\in S} \tau_i + \sum_{i,j\in V, i\neq j} x_{i,j} \cdot T^{\mu}_{i,j} + \Phi(P_T) * \sqrt{\sum_{i,j\in V, i\neq j} (x_{i,j} \cdot T^{\sigma}_{i,j})^2} \tag{15}$$

Where $\Phi(\cdot)$ is the value of the CDF function of a normal distribution. This is a non-convex cost function which makes this program Non Linear Mixed Integer Problem (NLMIP). This program can be solved using many solvers like Bonmin, BARON or Gurobi. For example, if $P_T = 0.5$ then the minimization function depends only on the mean values, and as P_T increases the weight of the variance in the cost function increases. One can approximate the nonlinear non-convex term by a taylor series around the an approximated value of the variance:

$$\sqrt{\alpha} = \sqrt{\alpha - \hat{\alpha} + \hat{\alpha}} = \hat{\alpha}\sqrt{1 + \frac{\alpha - \hat{\alpha}}{\hat{\alpha}}} \approx \hat{\alpha}(1 + 0.5 \cdot \frac{\alpha - \hat{\alpha}}{\hat{\alpha}}) \tag{16}$$

Where $\hat{\alpha}$ is the best estimation for the variance. For example, it can be the average of all the terms in T^{σ} multiplied by the number of edges in the tour: $2M + N - 2$. The final and complete formulation of the MIP is (1)-(5),(11),(12),(13) with (15) as the cost function. The full formulation is given in **??**

5 Clustering Heuristics

Solving the robust ECVRP is NP hard and there no polynominal time algorithm that solves it, even with the best commercial solvers. For fast solution times, heuristic solving methods can be introduced. In this paper a clustering heuristic is developed to divide the nodes to M groups. Each group is served by a single truck and an optimal Robust Energy Capacitated Travelling Salesman Problem (ECTSP) is solved. Since the ECTSP is smaller in size compared to the original ECVRP, it's optimal solution can be found faster, maybe even by solving the MIP. The main idea in clustering the nodes is the understanding that a single vehicle should serve nodes that are close to each other. When dividing the nodes into groups, it basically turns the big matrices that define the problem (with size $\mathbf{R}^{NxN}$) into smaller matrices, and if the clustering is done wisely many large numbers would disappear (for example, the 2 most distant nodes would probably end up in different groups and the edge between them will not be considered). Each group should include the Depot node and each group's total customers demand should be less than the maximal capacity of a single vehicle. It is also possible to limit the number of nodes in each group to the maximal number of nodes a single truck can visit in a tour - N_{max}. This leads to the constrained clustering problem - divide the graph into groups that minimize some cost function, while each group meets a set of defined constraints. Defining the measure of what is close and how to divide the graph into, so that the resultant tours would come close to the optimal solution is not trivial. In this paper we would examine a few techniques and analyze the pros and cons of each method. First, lets formulate the clustering problem. The nodes in (D, C) should be divided into M groups such that some cost function will be minimized. The cost matrix

describing the egde costs on the graph is given. Since the cost between nodes is not always linear with distance, it's not always possible to calculate a new cost matrix or add new auxiliary nodes - K-mean method would no be possible to implement with available data (when the weights on the edges are not linearly dependent on the 2D distance). The most obvious example is the edge that connecting 2 nodes with an obstacle between them (for example, a river). The distance is small but the time of travel is long and not trivial. The charging station nodes will be added after the clustering process is completed - further discussion in the clustering algorithm section. The formal program for the clustering problem:

$$\min \sum_{i=0}^{m} f(C_i) \cdot |G_i| \tag{17a}$$

$$s.t. \quad |G_i| <= N_{max}, \forall i = 1, ..., M \tag{17b}$$

$$\sum_{j \in G_i} q_j \leq Q, \forall i = 1, ..., M \tag{17c}$$

$$G_i \bigcap G_j \subseteq \{d_k\}, \quad i, j = 1, ..., M, i \neq j, k \in D \tag{17d}$$

$$\bigcup_i G_i = \{D, C\} \tag{17e}$$

Where, G_i is the set of all nodes in the i-th group, $d(\cdot)$ is some function of that takes a square matrix as input and outputs a scalar, $C_i \in \mathbf{R}^{N_i x N_i}$ is the i-th group cost matrix that represent the cost of travelling from any 2 nodes in the group. (17b) limits the size of each group, (17c) makes sure that the sum of demand in each group is less than the capacity of each vehicle, (17d) ensures that the intersection between any 2 groups can be a depot node at most (this can only be if both group start a the same depot) and (17e) is the union of all groups is the set of all depots and customers. The cost function to be minimized needs to represent the distance between the points in the group. The cost function should be invariant to the order the nodes are arranged in the group. Possible cost function can be: 1) The sum of the Frobenius norm of the M cost matrices - this cost function is easy to calculate but is dominated by the large numbers in the cost matrix that are usually not part of the solution, 2) sum of all the eigenvalues of all M matrices - this is a good measure of the size of the matrix but this also can create outlier points and is hard to practically cluster and the calculation of all the eigenvalues of large matrices can be time consuming. 3) sum of the max term in a row of each of the m cost matrices. 4) The maximal eigenvalue of the cost matrix. Choosing the cost function for the clustering problem is not trivial. The cost matrix should reflect in some way the cost f travelling between nodes and other constraints that affect the problem (energy consumption, in our case). if both parameters have the same cost trend (time of travel and energy consumption usually have positive correlation) then it is, usually, enough to use only one of them in the cost function. The cost function used in this paper is:

$$Z = T_\mu + \Phi(P_T) \cdot T_\sigma + (E_\mu + \Phi(P_E) \cdot E_\sigma)/R \tag{18}$$

Where, $Z \in \mathbf{R}^{NxN}$ and R is some tuning parameter that represent the average charging rate of the charging profile. This cost function takes into account the time minimization

problem, but doesn't take into account the energy problem. This assumes that the energy constraints of the problem are in some correlation with time and therefore minimizing time would also lower the energy. The clustering problem is NP hard. The algorithm used in this paper to cluster the groups is a simple and straight forward as follows: At first, there is an initialization process that divides the nodes to M groups with feasible solution. Secondly, loop over all nodes and check if any of them should move to a different group (does moving the node to different group lower the cost function). Thirdly, loop over all pair of node in different groups and check if they should switch groups. fourthly, loop on all options of 2 nodes in some group and 1 node on another group - check if the switch lowers the cost function. This search can continue on (loop over all k nodes in the some group and l nodes in a different group to check if they should switch places, and so on...) but it was found that searching up until this point gives very good results. Before changing any of the nodes labels, it should be checked that the new solution is still feasible. The clustering algorithm is described in 1. After the clustering of the customers into groups is completed, each group is added a number of charging stations - first the number of station for each group is evaluated by a rough estimate of the number of recharges needed, and then the relevant stations for each group are selected - This algorithm described in 2

6 Solving The Robust ECTSP

After the large Robust ECVRP has been divided into smaller Robust ECTSP problem, a numerical solver can be used to solve for the optimal tour for each vehicle. It can be done by solving a MIP or by a search algorithm like Dynamic Programming (DP), Divide and Conquer (D&C) or a combination of the 2. Since the ECTSP problem may still be large, we can use the same clustering algorithm that was described in 5 to reduce the dimension of the problem - divide the nodes in the group to sub-groups such that moving from group to group is possible only if all the node in the subgroup has already been visited. Furthermore, due to the additional energy constraints, many trajectories can be eliminated early in the search, thus a recursive search makes sense in this case. Remainder - the load capacity problem is irrelevant in this point since the initial clustering of the group made sure that the capacity constraint is met. The number of subgroups can be determined by the tuning. The lower the number of subgroups the faster the search algorithm will find a solution. The higher the number of subgroups, it's more likely the solution will get close to the optimal ECTSP. The main idea of the solver is to search recursively all possible routes with breaking points whenever the current best potential route is worst than the best route found so far or it's infeasible due to energy constraints. By using the sub groups technique described above, we can calculate the best route for each sub group as a function of the entering and exiting node from the group. This can be used later in the search algorithm to reduce calculations time. The search is done using the greedy method to sort the order of search to speed up finding good feasible solution and eliminate many potential tours as early as possible. The cost of each feasible route is calculated with a general charging profile with the optimal charging times for each visited charging station. The time and energy variance is sum of variances of

all edge along the tour. This recursive algorithm can be paralleled in a multi core processor with a shared memory. The algorithm can be stop with a timeout condition if some feasible solution is found. The algorithm is described in 3.

7 Results

This section presents the parameter tuning of the clustering algorithm and the results of solving the Robust ECVRP using the presented methodology on 2 types of scenarios: The first are a random generated scenarios - in each scenario the total number of nodes and vehicles are given. The position of all nodes are randomized from uniform distribution on a 100×100 map. Each Customer node demand is randomized from uniform distribution $q_i \sim \mathcal{U}[3, 15]$. The maximal vehicle load capacity is $Q = 100$. The energy consumption is linear function of the travel distance $e_{i,j} = c \cdot r_{i,j}$, where $c = 1.85$ where the maximal battery capacity is $SOC_{max} = 100.0$. The charging profile is linear with charging rate of 3. The standard deviation of the time and energy is randomized at 5% to 30% of the nominal value for each edge. The second data-set used is the data-set proposed in [9]. This a data-set for a small (up to 100 nodes) and large (up to 1000 nodes) EVRP instances with many attempts of solving them. The requirement of the energy faliure probability is $P_E = 0.999$ and time robustness probability is $P_T = 0.9$. The results are compared to the Best Known Solution (BKS) for each instance. We compare 2 main parameters: the quality of the solution (how close can it get to the BKS) and the time of calculation. All resulted calculated for this paper have been done on a Laptop equipped with Intel® Core™ i7-1260P 3.4 GHz and 16Gib of RAM. First, the tuning of the cost function is presented. In this section, the randomized scenarios were used together with the small data-set of (data-set E). data-set E contains data for a deterministic ECVRP. To convert it to a stochastic ECVRP the time and energy standard deviation of each node was taken a 20% of the nominal value. For these small data-sets a solution can be found using a MIP solver to get an estimate for good solution. Cost (19a) takes the sum of the maximal eigenvalue of the cost matrix multiplied by the number of nodes in the group (this is done using the Power Method for numerical efficiency). Cost (19b) is the sum of the frobenius norm of all the cost matrices. Cost (19c) is the sum of the absolute value of all the eigenvalues of the cost matrices:

$$f_1 = \sum_{i=1}^{M} \lambda_{max}(Z_i) \cdot |G_i| \tag{19a}$$

$$f_2 = \sum_{i=1}^{M} |G_i|_F \tag{19b}$$

$$f_3 = \sum_{i=1}^{M} \sum_{j=1}^{|G_i|} \lambda_j(Z_i) \tag{19c}$$

Where, Z_i is the cost matrix of group i for which the clustering is done. Table 1 shows the results for the tuning of the cost function type for the clustering algorithm. It shows the

cost function of the solution produced by solving the MIP with Gurobi solver (Gurobi was run with a timeout of 10 min), and solution with using one of the clustering cost function (19). All the cost functions presented in the table are the same for all cases and is the robust total tour time. The name of the scenario suggests it structure - The first letter is D if the scenario is deterministic ($T_\sigma = E_\sigma = 0$) or S for stochastic. the second letter is E (or X) if the scenario is from [9] or R if the scenario is randomized. the letter (than a number) represent the number of customers in the scenario. The letter k(and a number) represent the number of vehicles in the scenario. All the scenarios in E are single depot and R are multiple depots. It can be seen that as the number of nodes increase, cost function (19c) gets better and better results. All cost function types have similar calculation times. The chosen cost function type is (19c). 2 shows an example of a tour calculated on a randomized scenario with 3 vehicles a 2 depots. 3 shows a Monte-Carlo(MC) of the tour. Each MC runs randomized the time and energy of each edge along the tour. The upper plot is the total tour time of each vehicle. The lower plot is the minimal SOC of each vehicle along the tour. It can be seen that the probability of tour failure is as required. The sum of the 90% time of the vehicles is about the cost function.

Next, we solve the problem with the suggested algorithm and chosen cost function for all instances in [9]. The instances are solved twice - once as a deterministic problem and then as a stochastic. The deterministic case is compared to the BKS in terms of quality and time of solution. The BKS and best calculation time is taken form the best solution from [3, 10, 11] - not necessarily the same paper. The times presented of the BKS is sometimes approximated, since not all papers have accurate numbers. The results for the stochastic case will be a benchmark for future research. Table 2 shows the results of the tuned algorithm. It can be seen that the algorithm is faster than the fastest algorithm that has calculation time statistics. For the largest instances, if has a factor of 3-4 in calculation times. In terms of the quality of solution, as the instances get bigger, the algorithm gets very close to the BKS. This can be explained by the intuition that in large instances the tours are more compact so that a clustering heuristics works very well. As a remainder, the cost function presented is the robust total tour time:

$$Cost = T_\mu + \Phi(P_T) \cdot T_\sigma \tag{20}$$

Where, $P_T = 0.9$. The energy constraint are calculated with robustness probability of $P_E = 0.999$.

8 Conclusions

Solving the Robust ECVRP can be done quickly with clustering heuristics that produce sub optimal solution with short computation times. The clustering heuristics is a natural and intuitive way of solving the large scale vehicle routing problem. The added complexity of the energy constraints and robustness makes the corresponding MIP extremely hard to solve, but on the other hand makes it possible for the introduction of a simple iterative search algorithm that utilized the many tours eliminated by the energy constraints. Due to the fast solving times, this algorithm can be used for a closed loop

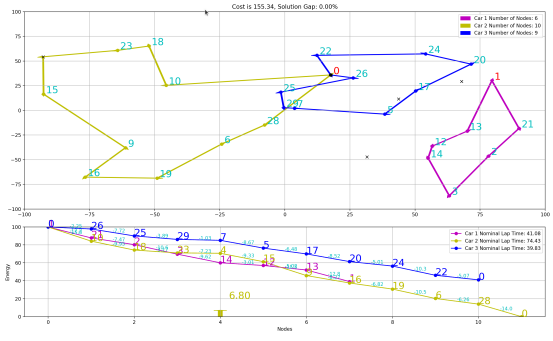

Figure 2: Randomized Scenario- Tour of 30 Customers, 3 Vehicles, 2 Depots

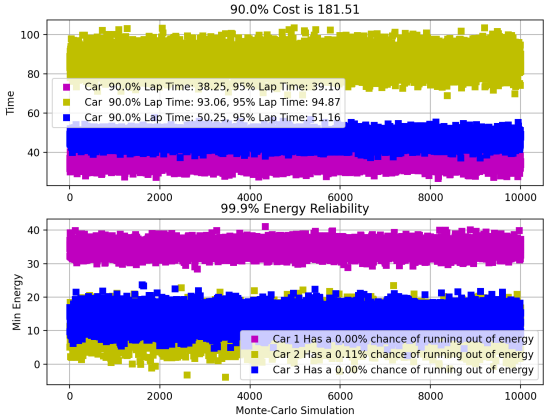

Figure 3: Randomized Scenario- MC of each vehicle Total time and Energy

feedback loop of the whole process. All the code used in this paper can be found in - `https://github.com/pukmark/EVRP_Optimization.git`

References

[1] G. Pataki, ‘‘Teaching integer programming formulations using the traveling salesman problem,’’ *Society for Industrial and Applied Mathematics*, vol. 45, pp. 116--123, 2003.

[2] G. M. P. J.-Y. Laporte, G. and F. Semet, ‘‘Classical and modern heuristics for the vehicle routing problem,’’ *International Transactions in Operational Research*, vol. 7, pp. 285--300, 2000.

[3] M. Mavrovouniotis, C. Menelaou, S. Timotheou, G. Ellinas, C. Panayiotou, and M. Polycarpou, ‘‘A benchmark test suite for the electric capacitated vehicle routing problem,’’ pp. 1--8, 2020.

[4] S. E. Schaeffer, ‘‘Graph clustering,’’ *Computer Science Review*, vol. 1, no. 1, pp. 27--64, 2007.

[5] M. D. W. Rizkallah, F. Ahmed, ‘‘A clustering algorithm for solving the vehicle routing assignment problem in polynomial time,’’ *International Journal of Engineering amp; Technology*, vol. 9, p. 1{8, Jan. 2020.

[6] T. Vidal, M. Battarra, A. Subramanian, and G. Erdog˘an, ‘‘Hybrid metaheuristics for the clustered vehicle routing problem,’’ *Computers Operations Research*, vol. 58, pp. 87--99, 2015.

[7] C. Le, D. D. Nguyen, J. Oláh, and P. Miklós, ‘‘Clustering algorithm for a vehicle routing problem with time windows,’’ *Transport*, vol. 37, p. 17{27, 05 2022.

[8] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas, ‘‘The electric vehicle routing problem with nonlinear charging function,’’ *Transportation Research Part B: Methodological*, vol. 103, pp. 87--110, 2017. Green Urban Transportation.

[9] S. T.-C. P. G. E. M. P. M. Mavrovouniotis, C. Menelaou, ‘‘Benchmark set for the ieee wcci-2020 competition on evolutionary computation for the electric vehicle routing problem,’’ pp. 1--8, March 2020.

[10] D. Woller, V. Kozák, and M. Kulich, ‘‘The grasp metaheuristic for the electric vehicle routing problem,’’ pp. 189--205, 2021.

[11] Y.-H. Jia, Y. Mei, and M. Zhang, ‘‘A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem,’’ *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10855--10868, 2022.

Algorithm 1 Clustering Algorithm For ECVRP with multiple depots

```
Groups = []
for i in range(M) do                          ▷ initialize the relevant depot to each group
    Group.append(d_i)                         ▷ d_i ∈ D, is the depot node from which car i exits
end for
for i in range(|C|) do                        ▷ initialize each customer to its cost minimizer
    for j in range(M) do
        iCost[j] = CalcCost([Group[j],i])     ▷ CalcCost(·) return cost of the group
    end for
    for j in argmin(iCost) do                 ▷ For loop on groups with ascending cost function
        if sum(Demand(Group[j]))+Demand[i] ¡ Q then
            Group[j].append[i]
        end if
        if j == M then
            return []                         ▷ Can't find feasible solution. add more trucks.
        end if
    end for
end for
PrevTotalCost = ∞
TotalCost = CalcTotalCost(Groups)
while TotalCost ¡ PrevTotalCost do
    PrevTotalCost = TotalCost
    for i in range(|C|) do                    ▷ Insert each customer in the group that minimizes Cost
        argMinGroup = CalcCostReplaceNode(Groups, C_i)
        iGroup = FindGroupOfNode(Groups,C_i)
        Groups[iGroup].remove(C_i)
        Groups[argMinGroup].append(C_i)
    end for
    for i in range(M) do ▷ Try to switch Customers between groups with 1/1 ratio, only if feasible
        for j in range(i, M) do
            for (k,l) in premuts(Group[i],Group[j],1,1) do ▷ premuts(G1,G2,i,j) returns all unique permutations with i terms from G1 and j terms from G2
                Group[i], Group[j] = CalcGroupsSwitch(Group[i], Group[j],C_k,C_l)   ▷ CalcGroupsSwitch return feasible groups that minimize cost by switching node k and l
            end for
        end for
    end for
    for i in range(M) do ▷ Try to switch Customers between groups with 2/1 ratio, only if feasible
        for j in range(M) do
            for (k1,k2,l) in premuts(Group[i],Group[j],2,1) do
                Group[i], Group[j] = CalcGroupsSwitch21(Group[i], Group[j],C_k1,C_k2,C_l)   ▷ CalcGroupsSwitch return feasible groups that minimize cost by switching node k_1 and k_2 with l
            end for
        end for
    end for
    TotalCost = CalcTotalCost(Groups)
end while
```

Algorithm 2 Clustering Algorithm For ECVRP with multiple depots

for i in range(M) **do**
  for iNode in Group[i] **do** iClosestCS = argmin(CostMatrix[iNode,CSs]) ▷ CSs is group of all charging stations
    if CSs[iClosestCS] not in Group[i] **then** Group[i].append(CSs[iClosestCS])
    end if
  end for
end for

Table 1: Results for Algorithm Parameter Tuning

Scenario	MIP Solver/BKS	MIP Gap [%]	f_1	f_2	f_3	Best$_f$
SE-n22-k4 Cost	733.57	25.0	421.0	414.1	414.1	f_2, f_2
Clustering Time [s]	-	-	0.1	0.1	0.1	f_1
SE-n23-k3 Cost	733.57	25.0	712.5	700.0	700.0	f_2, f_3
Clustering Time [s]	-	-	0.3	0.2	0.3	f_2
SE-n33-k4 Cost	733.57	25.0	993.9	943.2	944.2	f_1, f_2
Clustering Time [s]	-	-	0.2	0.3	0.3	f_1
DE-n22-k4 Cost	384.7	0.0	393.6	390.3	390.3	f_2, f_3
Clustering Time [s]	-	-	0.1	0.1	0.1	f_1
DE-n23-k3 Cost	571.9	0	665.5	665.5	665.5	f_1, f_2, f_3
Clustering Time [s]	-	-	0.3	0.2	0.3	f_2
DE-n33-k4 Cost			910.4	885.3	883.5	f_3
Clustering Time [s]	-	-	0.3	0.3	0.3	f_1
SE-n51-k5 Cost	733.57	25.0	595.4	570.5	577.6	f_2
Clustering Time [s]	-	-	0.7	0.9	1.2	f_1
SE-n101-k8 Cost	733.57	25.0	939.4	935.5	946.7	f_2
Clustering Time [s]	-	-	7.5	14.2	14.5	f_1
SE-n351-k8 Cost	27065	-	29820	29542	28568	f_3
Clustering Time [s]	-	-	38.6	41.3	47.5	f_1
SE-n351-k8 Cost	-	-	32254	31876	30556	f_3
Clustering Time [s]	-	-	37.6	40.1	45.5	f_1
SR-n30-k4 Cost	173.6	18.0	198.0	167.9	177.4	f_2
Clustering Time [s]	-	-	0.4	0.4	0.4	f_1, f_2, f_3
SR-n50-k4 Cost	223.7	19.7	213.3	228.2	227.0	f_1
Clustering Time [s]	-	-	1.3	1.4	1.3	f_1, f_3
SR-n100-k7 Cost	-	-	353.5	364.8	360.4	f_1
Clustering Time [s]	-	-	2.1	5.4	4.8	f_1
SR-n164-k18 Cost	223.7	19.7	768.2	701.6	699.4	f_3
Clustering Time [s]	-	-	36.5	31.9	28.4	f_1

Algorithm 3 ECTSP solving algorithm

procedure BESTROUTE, BESTCOST = ECTSP$_s olver(CurNode, CurRoute, CurTime, CurSOC, Tim$
procedure ALLNODESVISITED(CURROUTE) == TRUE(▷)Check if traveling to depot is feasible
CurRoute.append(depot)
CurTime, CurSOC, TimeVar, EnergyVar = UpdateRouteParams(CurTime, CurSOC, TimeVar, EnergyVar, CurNode, depot)
if CurSOC - Phi$_E * sqrt(EnergyVar)$¡0 **then** $returnCurRoute, inf$
if
thenreturn CurRoute, CurTime + Phi$_T$*$sqrt(TimeVar)$ **end if**

if CurNode in ChargeStationsNodes **then** ▷ If current node is a charging station
ChargingPotential = SOC_{max} - CurSOC
end if
NextPossibleNodes = NodesInSubGroup(CurNode) ▷ This function return the nodes in the subgroup of CUrNode
NextPossibleNodes - setNextPossibleNodes - setCurRoute
if NextPossibleNodes == [] **then** ▷ If all nodes in subgroup were visited - move to next subgroup
NextPossibleNodes = setallNodes - setCurRoute
SortedNodes = sort(TimeMatrix[CurNode,NextPossibleNodes] ▷ sort nodes by distance from CurNode
end if
for iNode in SortedNodes **do**
NextTime, NextSOC, NextTimeVar, NextEnergyVar = UpdateRouteParams(CurTime, CurSOC, TimeVar, EnergyVar, CurNode, iNode)
if NextTime + Phi$_T * sqrt(TimeVar)$≥ BestCost **then**
continue
end if
MinPossibleTimeToCompleteTour = MinTimeEstimate(CurTime, NodeLeftToVisit)
MinPossibleTimeVarToCompleteTour = MinTimeVarEstimate(TimeVar, NodeLeftToVisit)
MinPossibleEnergyToCompleteTour = MinEnergyEstimate(CurEnergy, NodeLeftToVisit)
MinPossibleEnergyVarToCompleteTour = MinEnergyVarEstimate(EnergyVar, NodeLeftToVisit)
if MinPossibleTimeToCompleteTour + Phi$_T$ * $sqrt(MinPossibleTimeVarToCompleteTour) \geq BestCost$ **then** *continue*
if
if **then****then**MinPossibleEnergyToCompleteTour + Phi$_E$ * $sqrt(MinPossibleEnergyVarToCompleteTour)$ + $MinPossibleChargingPotential$ < 0 *continue*
if R **then**oute, Cost = ECTSP$_s olver(iNode, CurRoute, NextTime, NextS$
$BestCostRoute[-1] == depot$ **then**
if B **then**estCost = Cost
BestRoute = Route
end if

Table 2: Results for Clustering Algorithm On Large Scale instances

Scenario	BKS Cost	Solution Time [s]	Clustering Cost	Clustering Time [s]	Sol Gap [%]
SX-n143-k7	-	-	19849	104	-
DX-n143-k7	16028	104	17993	101	13.0
SX-n214-k11	-	-	13856	124	-
DX-n214-k11	11133	250	13152	114	18.0
SX-n351-k40	-	-	13856	144	-
DX-n351-k40	26478	500	28574	139	7.0
SX-n459-k26	-	-	30547	635	-
DX-n459-k26	24763	1900	28896	601	16.0
SX-n573-k30	-	-	62067	1776	-
DX-n573-k30	51929	2500	57906	1503	11.5
SX-n685-k75	-	-	81683	674	-
DX-n685-k75	70834	5511	75175	654	6.1
SX-n749-k98	-	-	-	-	-
DX-n749-k98	80299	7258	83366	438	3.8
SX-n819-k171	-	-	-	-	-
DX-n819-k171	164289	6612	167075	236	1.7
SX-n916-k207	-	-	-	-	-
DX-n916-k207	341649	6774	344346	294	0.7
SX-n1001-k43	-	-	-	-	-
DX-n1001-k43	77476	-	81875	3615	5.7