Solving Electronic Vehicle Routing Problem using general meta-heuristic optimizer

Jan Švrčina

Abstract

This project was focused on solving the NP-hard Electronic Vehicle Routing Problem [1], which is derived from the Vehicle Routing Problem [2], introducing additional constrains which require the vehicle to recharge in designated charging stations after traveling a certain distance. The goal of this project is to transform this problem into formulation for a provide generic meta-heuristic solver. Since most instances of the EVRP have proven too hard for the generic solver, an additional repair procedure, transforming current solution into a feasible solution, has been introduced to the generic solver to combat this problem. The performance of the generic solver, with and without the repair procedure, has been compared with tailored EVRP solver.

Index Terms

Electronic Vehicle Routing Problem, ERVP, Meta-heuristic, Discrete Optimization

1 Introduction

Electronic Vehicle Routing Problem [1] is derived from the Vehicle Routing Problem [2], introducing additional constrains which require the vehicle to recharge in designated charging stations after traveling a certain distance. These additional constrains are motivated by real world applications, since electronic vehicles have limited range before having to recharge. The main goal of this project is to compare the performance of a provided general metaheuristic solver [3] (https://github.com/wolledav/permutator) with a solver tailored for EVRP (https://github.com/wolledav/VNS-EVRP-2020).

1.1 Mathematical model

The mathematical model of the EVRP model is the following [4].

$$\min \sum_{i \in V, j \in V, i \neq j} d_{ij} x_{ij} \tag{1}$$

s. t.

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

$$\sum_{j \in V, i \neq j} \leq 1, \forall i \in F' \tag{3}$$

$$\sum_{j \in V, i \neq j} x_{ij} - \sum_{j \in V, i \neq j} x_{ji} = 0, \forall i \in V \tag{4}$$

$$u_j \leq u_i - b_i x_{ij} + C(1 - x_{ij}), \forall i \in V, \forall j \in V, i \neq j \tag{5}$$

$$0 \leq u_i \leq C, \forall i \in V \tag{6}$$

$$y_j \leq y_i - h d_{ij} x_{ij} + Q(1 - x_{ij}), \forall i \in I, \forall j \in V, i \neq j \tag{7}$$

$$y_j \leq Q - h d_{ij} x_{ij} + Q(1 - x_{ij}), \forall i \in F' \cup 0, \forall j \in V, i \neq j \tag{8}$$

$$0 \leq y_i \leq Q, \forall i \in V \tag{9}$$

$$x_{ij} \in \{0,1\}, \forall i \in V, \forall j \in V, i \neq j \tag{10}$$

This is an ILP model, the binary variable x_{ij} denotes if there is path from vertex i to j, set vertices $V = I \cup F' \cup 0$ is composed of depot 0, customer I and charging stations F' with necessary amount of copies (charging stations can be visited multiple times, at most $2|I|$ as in worst case we travel through one station back and forth to each customer). The object is to minimise sum of total distances, d_{ij} is distance from vertex i to j, while satisfying the conditions for customer demands, b_i is demand of customer i, when the carrying capacity of the vehicle Q is recharged in the depot, u_i is the current load in the vertex i. We also need to satisfy the charge condition, limiting the distance the vehicle can travel, the maximum charge Q is refilled in every charging station and depot, the current charge in vertex i is denoted by y_i, its consumption is defined as distance times energy consumption rate h.

1.2 Dataset

We are using a dataset with 17 instances ranging from 30 to 1010 vertices [4]. This dataset has and is being used for benchmarking algorithms solving the EVRP.

1.3 Generic solver

The provided generic solver works representing the solution as a sequence / permutation of vertices with upper and lower bound count of each vertex, every other condition has to be encoded into the fitness function, consisting of optimization goal and unmet conditions penalization. This solution iteratively improved using a metaheuristic, alternating between random perturbation, move far away from the neighbourhood of the current solution but retain some of its properties, and local search, exhaustively applying local operators, elementery changes in the solution like insert, swap or remove and selecting those that decrease the value of fitness function (assuming minimization of objective). Detailed description of the metaheuristics, perturbations and local operators is in "Metaheuristic Solver for Problems with Permutative Representation" by D. Woller et. al. [3] and a derived masters thesis "Metaheuristic Algorithms for Optimization Problems Sharing Representation" by J. Hrazdíra [5].

2 Implementation

There are 4 parts of this project that had to be implemented, extending the current project of the generic solver published on https://github.com/wolledav/permutator. There a new branch ERVP has been created and the following features have been implemented.

2.1 ERVP interface

A complete interface for the ERVP problem feasible for the generic solver has been implemented. The `evrp` files are converted to `json` using a python script `evrp_create_json.py`. These files are loaded into the solver, setting up a ERVP instance structure with all the parameters described in the mathematical model section, distances between nodes are saved in a distance matrix. This instance has a fitness function, which returns a numeric value based on optimization criterion and condition penalization, and the information about feasibility of the provided solution (have all conditions been met).

2.2 Repair function

Since during the initial experiments, the EVRP problem has proven to be too difficult for the solver, because it took very long to find an initial feasible solution or a feasible improvement and for bigger instances a feasible solution was never even found, a repair procedure has been introduced into the solver, providing a feasible solution based on the order of customer in the initial solution. This function is called after the initial solution has been generated and after every perturbation. The pseudocode of the repair procedure is the following, it is inspired by a repair procedure in "A meta-heuristic for capacitated green vehicle routing problem" by Zhang, Gajpal and Appadoo, thus named a ZGA repair procedure.

Data: any tour of customers T (starting and finishing in depot), depot D
Result: feasible EVRP tour T_{EVRP}
T_{EVRP}.push_back(T.pop_front());
$current \leftarrow T_{EVRP}$.back();
$next \leftarrow T$.front();
while *T not empty* **do**
  if *remaining_load(T_{EVRP}) $\geq$ demand($next$)* **then**
    if *charging station can be reached via $next$* **then**
      T_{EVRP}.push_back($next$);
      $current \leftarrow next$;
      T.pop_front();
      $next \leftarrow T$.front();
    else
      $next \leftarrow$ charging station closest to $next$ reachable from $current$;
      T.push_front(next);
    end
  else
    $next \leftarrow D$;
    T.push_front(next)
  end
end

Algorithm 1: ZGA repair procedure

2.3 Irace configuration optimization

Iterated Racing for Automatic Algorithm Configuration (irace) [6] is a tool for hyperparameter optimization, it has been used to select the best configuration of metauheristic, perturbation and local operators. The optimization has been setup analogically to previous problems in the generic solver project, can be started by command `irace` in `tuning_EVRP_repair` and `tuning_EVRP_no_repair` folders respectively. The selected configurations are saved in `EVRP_repair.json` and `EVRP_no_repair.json`, they are the same except for the repair function toggle.

Problem with double bridge perturbation

During testing, a bug in the provided solver has be detected. The double bridge pertubation function (`void LS_optimizer::double_bridge()` method on line 570 in file `src/LS_optimizer.cpp`) is outputting vertex indexes which are higher than the number of vertices, this causing memory errors in the fitness function. Because of this, the Double Bridge and Random Double Bridge perturbations have been excluded from the search.

2.4 Experiment setup

Bash script `evrp_run_experiment.sh` runs the generic optimizer for all 17 instances 20 times with seed $s \in \{1, \ldots, 20\}$ and timeout $t_{max} = \frac{|I|+|F|}{c}\nu$, where $|I|$ is number of customers, $|F|$ is the number of charging stations, ν is complexity coefficient, 1 for instances E-n22-k4 to E-n101-k8, 2 for X-n143-k7 to X-n916-k207 and 3 for X-n1001-k43, c is a scaling constant so that the whole experiment runs for approximately 16 hours for each (repair and no repair) method.

3 Results

All three methods, reference tailored solver, generic solver with repair function and generic solver without repair function have been tested on all 17 instances for 20 seeds, each method was given 16 hours (with spread according to Experiment setup) of computation time on Intel(R) Core(TM) i5-8250U (1.6 GHz) processor, the general solver was allowed parallelization on all 8 cores. The final results table 1 includes

mean and minimal value of the found solutions, percentile diff between minimum and mean and for the generic solver without repair shows the percentage of found feasible solutions. Note: Calculating difference between minimum and mean for no repair method is not relevant as these statistics are calculated at most from 1 to 3 results and only for 6 instances. For repair and no repair method of the generic solver, the percentile gap to reference is presented in table 2.

	reference			repair			no repair		
instance	min	mean	diff	min	mean	diff	min	mean	feasible
E-n22-k4	385	385	0%	401	445	11%	485	491	10%
E-n23-k3	572	572	0%	584	647	11%	668	701	15%
E-n30-k3	509	509	0%	517	556	8%	623	623	5%
E-n33-k4	840	840	0%	929	975	5%	1007	1007	5%
E-n51-k5	530	543	2%	549	609	11%	620	620	5%
E-n76-k7	693	679	1%	724	770	6%	799	832	10%
E-n101-k8	839	846	1%	959	1005	5%	N/A	N/A	0%
X-n143-k7	16028	16549	3%	18001	18645	4%	N/A	N/A	0%
X-n214-k11	11324	11482	1%	14038	14841	6%	N/A	N/A	0%
X-n351-k40	27065	27218	1%	34796	36506	5%	N/A	N/A	0%
X-n459-k26	25371	25582	1%	39191	42676	9%	N/A	N/A	0%
X-n573-k30	52182	52582	1%	63150	64554	2%	N/A	N/A	0%
X-n685-k75	71359	71818	1%	257584	272642	6%	N/A	N/A	0%
X-n749-k98	81417	81677	0%	237847	250798	5%	N/A	N/A	0%
X-n819-k171	165377	165883	0%	433054	443677	2%	N/A	N/A	0%
X-n916-k207	344243	344881	0%	634932	648988	2%	N/A	N/A	0%
X-n1001-k43	77684	78079	1%	392798	406322	3%	N/A	N/A	0%

TABLE 1
Performance results

	repair		no repair	
instance	min	mean	min	mean
E-n22-k4	4%	16%	26%	28%
E-n23-k3	2%	13%	17%	23%
E-n30-k3	2%	9%	22%	22%
E-n33-k4	11%	16%	20%	20%
E-n51-k5	4%	12%	17%	14%
E-n76-k7	4%	10%	15%	19%
E-n101-k8	14%	19%	N/A	N/A
X-n143-k7	12%	13%	N/A	N/A
X-n214-k11	24%	29%	N/A	N/A
X-n351-k40	29%	34%	N/A	N/A
X-n459-k26	54%	67%	N/A	N/A
X-n573-k30	21%	23%	N/A	N/A
X-n685-k75	261%	280%	N/A	N/A
X-n749-k98	192%	207%	N/A	N/A
X-n819-k171	162%	167%	N/A	N/A
X-n916-k207	84%	88%	N/A	N/A
X-n1001-k43	406%	420%	N/A	N/A

TABLE 2
Gaps to reference

4 Discussion

4.1 Conclusion

The ERVP extension of the provided generic solver has been implemented, its hyperparameters optimized and compared to the reference method. From the presented results we can draw several conclusions.

- Generic solver performs notably worse than the tailored solver and with increasing size of the problem, the difference becomes more apparent.

- Generic solver without the repair procedure can only be used on small problems, for large problems it becomes too computationally demanding to find a feasible solution.
- Generic solver has to ”luck out” to find an improving solution, there is much smaller probability of improvement between metaheuristic cycles, compared to the tailored solver. This is apparent from the mean-minimum difference, all solutions of the tailored solver have almost equal objective value (at maximum 3%), while the generic solver ranges from 2% to 11%, which is a noticeable disparity.

4.2 Potential improvements

The most apparent is the inclusion of the double bridge perturbation as the ERVP problem is similar to TSP, we wanna go through to all customers and thus rerouting is better motivated than just blind swapping. Second would be to extend the computation time for the solver or use better hardware to leverage parallelism. Smaller improvements could be done by optimizing the calculation of fitness function or optimizing the repair function (it does not scale well with large problems).

REFERENCES

[1] J. Lin, W. Zhou, and O. Wolfson, “Electric vehicle routing problem,” *Transportation Research Procedia*, vol. 12, pp. 508–521, 2016, tenth International Conference on City Logistics 17-19 June 2015, Tenerife, Spain. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146516000089

[2] G. B. Dantzig and J. H. Ramser, “The truck dispatching problem,” *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[3] D. Woller, J. Hrazdíra, and M. Kulich, “Metaheuristic solver for problems with permutative representation,” in *Intelligent Computing & Optimization*, P. Vasant, G.-W. Weber, J. A. Marmolejo-Saucedo, E. Munapo, and J. J. Thomas, Eds. Cham: Springer International Publishing, 2023, pp. 42–54.

[4] M. Mavrovouniotis, C. Menelaou, S. Timotheou, C. Panayiotou, G. Ellinas, and M. Polycarpou, “Benchmark set for the ieee wcci-2020 competition on evolutionary computation for the electric vehicle routing problem,” 03 2020.

[5] J. Hrazdíra, “Metaheuristic algorithms for optimization problems sharing representation,” 2022. [Online]. Available: https://dspace.cvut.cz/bitstream/handle/10467/102112/F3-DP-2022-Hrazdira-Jan-Hrazdira_Metaheuristic_Algorithms_v2.pdf

[6] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari, “The irace package: Iterated racing for automatic algorithm configuration,” *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.