Customer Penalty-Based Heuristic Algorithm for the EURO Meets NeurIPS 2022 Vehicle Routing Competition

Yunhao Li [*]**,Junjie Zhang** [*]**,Yuxuan Wang ,Wenhan Shao ,Zhouxing Su ,Zhipeng Lü**
School of Computer Science and Technology
Huazhong University of Science and Technology
suzhouxing@hust.edu.cn

Abstract

This document describes HustSmart's submission to the EURO Meets NeurIPS 2022 Vehicle Routing Competition[1]. There are two different heuristic algorithms for two different variants of VRPTW. For the static variant, our solver is improved based on the solver Router, which won the first place in the VRPTW track of the 12th DIMACS competition. Our team won the second place in the 12th DIMACS competition. For the dynamic variant part, we design a hybrid algorithm. This algorithm assigns a penalty to each customer that can be postponed to the next epoch, and uses Google's OR-Tool and the static solver in baseline as components.

1 Introduction

From the input of the solver, we can judge whether the current instance is a static variant or a dynamic, and then use different algorithms to solve it. Our code is open source at https://github.com/liyunhaocn/npsvrp

1.1 Static Variant

We improved solver based on the algorithm in baseline/hgs_vrptw. The main improvements are as follows.

1. A crossover operator EAX is added to the algorithm, which is considered together with the two existing crossover operators, and then a the penalty function is used for scoring.
2. Each time a new optimal solution is found, the SISR algorithm is used to refine the solution.
3. After $nbIter$ consecutive local searches of the algorithm do not improve, use the DLLSMA [2] algorithm to improve the population.

1.2 Dynamic Variant

We design a penalty function based on the characteristics of each customer's time window $[e, l]$, demand, service time, and time window width $|l - e|$. Then we call OR-Tool to determine which customers are deferred to be postponed to next epoch, Finally, the algorithm of baseline/vrptw is called to minimize the cost of remaining customers.

[*]These authors contributed equally to this work.

EURO Meets NeurIPS 2022 Vehicle Routing Competition Workshop.

2 Algorithm for Static Variant

Algorithm baseline/hgs_vrptw is designed based on HGS [3], which is a state-of-the-art algorithm to solve CVRP. We make some improvements based on this baseline. First of all, we found that the baseline algorithm converges very quickly, but the results of each run are not stable enough. Secondly, the final running result of the algorithm has a great correlation with the initial population. We reduce the $nbIter$ in the baseline, so that the population can get multiple opportunities for reconstruction. then, we mobilize the DLLSMA [2] algorithm to optimize the population before resetting population. In DLLSMA solver, the SISR[4] method is used.

In order to increase the diversity of search, we add the EAX crossover [5] to the total algorithm, Compete with the two crossover operators in the baseline, and select the best offspring from the offspring generated by the three crossover operators.

We try to improve the ox crossover operator [3] and call it ox*. The main operation process of the ox crossover operator is to first inherit a continuous segment of customers from a parent, The remaining customers are then filled into the children in their relative order in the second parent. The ox* operator randomly inherits several consecutive fragments of the first parent, Then fill the children with the remaining customers in the order they were in the second parent.

3 Algorithm for Dynamic Variant

At the beginning of the algorithm, the penalty value corresponding to each node is calculated based on the information returned by the environment, and the penalty values of these nodes represent the cost of removing these points from the solution. If the penalty value of a node is higher, the less the node should be deleted. The important thing to note here is to set the penalty values corresponding to the nodes in the must-go set to infinity, which ensures that the nodes in the must-go set will not be dropped from the solution. The penalty value of a node is calculated as follows.

$$penalty[i] = a * latest\,arrival[i] + b * duration[depot, i] + c$$

where $penalty[i]$ denotes the penalty value of node i. $latest\,arrival[i]$ denotes the latest time of the time window of node i. $duration[depot, i]$ denotes the distance of node i to the depot. a, b, c are constants.

We have tried to calculate the penalty value by taking the service time and the demand size of the nodes, but the effect is not obvious from the experimental results. Therefore, we did not use this information to aid in the calculation of the penalty values. How to determine these constant values is often a tricky problem. We used the differential evolution algorithm to solve this problem. The chromosome of the differential evolution algorithm is composed of these constants (a, b, c), and the fitness function value is the average score obtained from multiple simulation experiments on all instances of these parameters.

Then, the known information and penalties are input into OR-Tools, and solve the problem with the method Penalties and Dropping Visits. The feasible solution and a collection of deleted points can be obtained from OR-Tools.

After that , the latest departure time corresponding to each route in the obtained feasible solution is calculated. If the latest departure time is greater than a constant t, the route is deleted. The reason for this is that if a route has a late latest departure time, then this route can be delayed. The nodes on this route can be combined with nodes from the next round to produce a higher quality solution.

Next, the remaining routes are input as initial solutions to our static solver. Since the performance of OR-Tools on the VRPTW problem is not as high as that of our static solver, this allows further optimization of the solution quality. Experiments show that this step is effective.

Finally, the feasible solutions obtained from the static solver are processed using the same method as before. In other words, the routes that are not urgent in the feasible solution are deleted. The remaining route after deletion is the final solution.

References

[1] W. Kool, L. Bliek, D. Numeroso, R. Reijnen, R. R. Afshar, Y. Zhang, T. Catshoek, K. Tierney, E. Uchoa, T. Vidal, and J. Gromicho, "The EURO meets NeurIPS 2022 vehicle routing competition," 2022.

[2] "http://dimacs.rutgers.edu/events/details?eid=2091,"

[3] T. Vidal, "Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood," *Computers & Operations Research*, vol. 140, p. 105643, 2022.

[4] J. Christiaens and G. Vanden Berghe, "Slack induction by string removals for vehicle routing problems," *Transportation Science*, vol. 54, no. 2, pp. 417–433, 2020.

[5] Y. Nagata, O. Bräysy, and W. Dullaert, "A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows," *Computers & operations research*, vol. 37, no. 4, pp. 724–737, 2010.