Hybrid Genetic Search for the Dynamic Vehicle Routing Problem

Mohammed Ghannam
HTW Berlin
ghannam@htw-berlin.de

Ambros Gleixner
HTW Berlin / Zuse Institute Berlin
gleixner@zib.de

Abstract

This document describes *dynamo* team's submission[1] to the EURO Meets NeurIPS 2022 Vehicle Routing Competition. [1]

1 Introduction

This report presents our work in the *EURO Meets NeurIPS 2022 Vehicle Routing Competition*. The competition included two variants of the vehicle routing problem: the *static* variant, which is a standard capacitated vehicle routing problem with time windows (VRPTW), and the *dynamic* variant, where the customers are passed to the solver in epochs, the solver then decides which of these customers to dispatch and create routes for, and which to leave for future epochs. Consequently, a subset of the customers must be dispatched in their corresponding epoch, otherwise, their time window will have passed by the next epoch. For a detailed description of the variants, refer to the competition's website. Each participant in the competition was required to submit two solvers for each variant. The organizers provided baseline solvers for both variants and they are based on the work by Kool et. al [2] to adapt the hybrid genetic search open-source code by Vidal for the capacitated vehicle routing problem [3], to add support for time-windows and add other improvements to different aspects of the algorithm. Given the success of hybrid genetic search on many variants of the vehicle routing problem [4], the goal of this work is to adapt this code for the dynamic variant.

The report is structured as follows: Section 2 contains a brief description of the hybrid genetic search framework, Section 3 contains improvements to the static solver baseline, Section 4 contains the adaptation of the static baseline for the dynamic variant, and Section 5 contains results and conclusions.

2 Hybrid Genetic Search

The Hybrid Genetic Search (HGS) algorithmic framework introduced by Vidal [5] first for the multi-depot and the periodic vehicle routing problem has proven effective on other multiple problem variants [4]. It is a meta-heuristic that includes many techniques to balance solution quality and diversity. Being a genetic algorithm, it uses the corresponding metaphor of referring to solutions as *individuals*, a set of solutions as a *population*, and *fitness* to refer to a measure of solution's quality.

The algorithm works by managing two sets for feasible and infeasible solutions. In the beginning, an *initial population* is created using multiple construction heuristics. Then at each iteration, two individuals are chosen based on their fitness, and a *crossover operator* is applied on them to create another individual. This individual is then improved using *local search* operators, and added back to the corresponding population. The -penalized- cost of a solution is comprised of two parts, one for the total distance traveled in the routes, and the other for penalizing infeasibility of the routes. This is then combined with a diversity measure to represent the overall fitness of the solution.

[1] Code for this work is to be released at `https://github.com/mmghannam`

EURO Meets NeurIPS 2022 Vehicle Routing Competition Workshop.

For the purposes of a compact presentation, many of the details of the algorithm are skipped here, for a detailed description refer to [5]. In the following sections, both solvers for the static and dynamic variants are presented. One thing to note is that due to the small number of tested instances in the qualification phase, this might not be a good representative of their overall performance. Therefore, we mention some of the ideas that showed some improvement in local tests but did not work on the hidden test instances.

3 Static Variant Solver

This solver for the static variant is based on the HGS solver for the VRPTW provided in the competition, later referred to as HGSTW. We added a new crossover operator based on the HGreX operator described in [6], and the best of all crossover operators is used. HGreX works by first choosing a node at random and add it to the child solution, then from that node it checks outgoing arcs from it in both parent solutions and chooses the shortest one. If connected nodes from these arcs are assigned before a random unassigned node is chosen. This process is then repeated until the child solution contains all nodes.

Other ideas that did not show an improvement on the test instances were:

- A construction heuristic based on insertion by regret as mentioned in [7];
- Two local search operators: *or-opt*, that moves n subsequent clients in a specific route to after some other node in the solution, where $n \in [3, \frac{|customers|}{2}]$ and a *swap-tail* operator, that for two nodes in different routes, the following nodes that are visited after them are swapped.

4 Dynamic Variant Solver

The main contribution of our work is an extension of the static solver to natively handle the dynamic variant. The goal is to pass the must-dispatch data to HGSTW and harness its advanced solution pool management and improvement techniques in finding the best subset of customers to dispatch at each epoch, and find good routes for these customers.

To achieve this the following modifications were applied:

- **Partial solutions.** We allow for solutions to have variable length, meaning any subset of the customers is allowed to form a solution.
- **Fitness.** The natural next step is to modify the cost (fitness) calculation in order for it to be comparable between solutions with different number of customers, and to reflect the new feasibility criterion of visiting all *must-dispatched* customers. For feasibility, the cost is penalized by the number of *must-dispatched* customers that do not appear in the solution. To achieve comparability, one possibility is to add the number of non-dispatched customers multiplied by some parameter; and another is simply normalizing the calculated cost in HGSTW by the number of dispatched nodes it contains. We found the latter choice to work better, and required no extra parameter to optimize over.
- **Initial population.** Random orderings of all customers are generated, then optional customers are removed from these orderings with a certain probability, afterwards, the SPLIT [8] algorithm is run to divide each ordering into routes.
- **Future flexibility.** With these modifications, we have a working adaptation for the dynamic variant. However, for it to improve upon baselines, other additions are required. One idea that worked is including an addition to the cost that reflects how limiting a solution is for future epochs. Two values were tested: *lateness*, which is based on the intuition that customers whose time windows end earlier should be preferred so that it would be less likely to see them as *must-dispatched* in future epochs if they were skipped in the current one; the latest time for which the customer can be served is used to reflect this notion in the cost. The other is *intersections*, which is the number of times a customer's time window intersects with the time window of another. Favouring the customers with a higher number of intersections in earlier epochs could potentially add flexibility in later epochs. Only the

lateness measure was included in the final submission as the other did not work well on the test instances.

5 Results & Conclusions

In the qualification phase, the solvers came in 9th place, where the dynamic solver, despite not using machine learning, showed a significant improvement over the baselines. This approach could form the basis for a general solver for the dynamic vehicle routing problem, that can further be enhanced using machine learning techniques for example to learn a better cost function.

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] W. Kool, J. O. Juninck, E. Roos, K. Cornelissen, P. Agterberg, J. van Hoorn, and T. Visser, "Hybrid Genetic Search for the Vehicle Routing Problem with Time Windows: A High-Performance Implementation," p. 7.

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

[4] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A unified solution framework for multi-attribute vehicle routing problems," *European Journal of Operational Research*, vol. 234, pp. 658–673, May 2014.

[5] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems," *Operations Research*, vol. 60, pp. 611–624, June 2012.

[6] K. Puljić and R. Manger, "Comparison of eight evolutionary crossover operators for the vehicle routing problem," *Mathematical Communications*, vol. 18, Nov. 2013.

[7] M. Zhao and Y. Lu, "A heuristic approach for a real-world electric vehicle routing problem," vol. 12, no. 2, p. 45. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[8] T. Vidal, "Technical Note: Split Algorithm in O(n) for the Capacitated Vehicle Routing Problem," *Computers & Operations Research*, vol. 69, pp. 40–47, May 2016.