Structured Learning for the EURO Meets NeurIPS 2022 Vehicle Routing Competition

Team `Kléopatra` **(Kai Jungel[1], Léo Baty[2], Patrick Klein[1])**
[1] School of Management, Technical University of Munich, Munich, Germany
kai.jungel@tum.de, patrick.sean.klein@tum.de
[2] CERMICS, École des Ponts, Marne-la-Vallée, France
leo.baty@enpc.fr

1 Introduction

This paper summarizes our approach to solving the *static* and *dynamic* variants of the *vehicle routing problem with time windows* (VRPTW) considered in the *EURO Meets NeurIPS 2022 Vehicle Routing Competition* [1]. We tackle the static VRPTW with an improved version of the Hybrid-Genetic-Search (HGS) algorithm supplied by the organizers [2]. For the dynamic variant, we introduce a novel structured learning pipeline, where we learn to represent the dispatching problem raised in each epoch as a deterministic prize-collecting vehicle routing problem with time windows (PC-VRPTW) which can be solved using a combinatorial algorithm.

In what follows, we first focus on the static problem, summarizing our changes to the HGS algorithm. We then introduce a structured learning pipeline to solve the dynamic problem. We conclude by benchmarking our approach against the baseline approaches made available by the organizers. We refer to our supplemental material [3] for a more thorough discussion of our approaches.

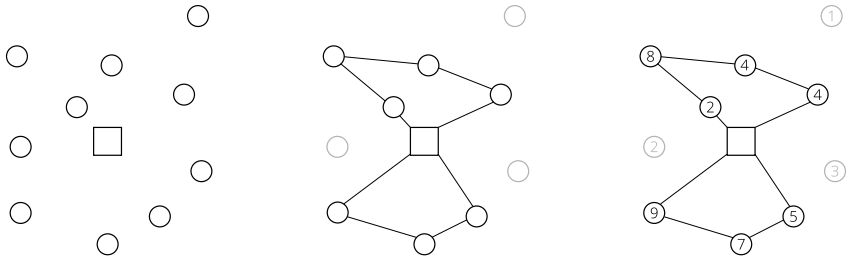
2 Static Problem: Hybrid Genetic Search improvements

We add several performance optimizations to improve the convergence of the HGS. Wouter et al. [2] implement an efficient local search caching mechanism that keeps track of route modifications during local search, avoiding superfluous neighborhood evaluations of unchanged routes. We extend this mechanism to track route changes outside the local search. This improved performance in practice as, among others, the selective route exchange operator tends to modify only a small subset of routes.

The HGS algorithm's local search implements two mechanisms to improve computational performance. First, the procedure accepts moves based on a first-improvement criterion [2], [4]. Second, the algorithm limits neighborhood size according to a granularity criterion, i.e., considers only the Γ most promising moves, according to spatial and temporal proximity, for each vertex. To balance the greediness resulting from this design, the HGS randomizes the order of moves considered across local search invocations. Furthermore, the algorithm periodically shuffles each vertex's promising arc set. Our experiments indicate that this additional diversification mechanism is detrimental in scenarios with limited runtime as it, due to the first improvement criterion, often introduces non-optimal arcs into the solution. We thus maintain ordered sets of promising arcs at all times. We account for this modification in our parameter setting and restart early, i.e., after 6000 iterations without improvement.

Finally, we prune superfluous data from central local search data structures and reorganize the local search control flow to increase the likelihood of cache hits.

EURO Meets NeurIPS 2022 Vehicle Routing Competition Workshop.



(a) Epoch instance with requests x_t.

(b) Routes y_t^* dispatched by an optimal strategy in epoch t.

(c) Solution y_t of the PC-VRPTW instance constructed from correctly predicted prizes.

Figure 1: Illustration of our approach. Solving a PC-VRPTW yields optimal routes $y_t = y_t^*$ if our predictor assigns high prizes to requests dispatched in the optimal solution, and low prizes to postponed requests.

3 Dynamic Problem

The optimal solution to any epoch instance can be obtained by first deciding on a set of requests to postpone and dispatch respectively, and then solving a VRPTW on the set of dispatched requests. This however requires to learn a binary decision, i.e. to dispatch or postpone a request, which is difficult due to the combinatorial structure of the problem. Instead, our approach to solving the dynamic problem bases on the idea illustrated in Figure 1: for any epoch instance with requests x_t, there exists a prize vector θ_t such that including only those requests where the detour is smaller than the prize collected by serving them equals the optimal dispatched routes. This reduced the learning problem to predicting prizes, and the optimization problem to the well-known prize-collecting VRPTW (PC-VRPTW), a variant of the VRPTW that drops customer covering constraints and instead pays a request-specific prize θ_v for each served request v. Intuitively, the optimal choice of request prizes θ_v corresponds to the lowest cost caused by serving request v in a later epoch.

However, due to the stochastic nature of the dynamic problem, prizes θ_v are uncertain and thus need to be estimated. We follow a machine learning based approach for this purpose, which leads to the structured learning pipeline with an integrated combinatorial optimization layer as illustrated in Figure 2:

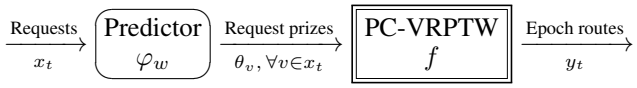


Figure 2: Our pipeline.

At the start of each epoch, we predict for each request v the cost of serving v in a later epoch. We then construct and solve a PC-VRPTW instance using these values as request prizes. Solving this instance yields a set of routes containing only profitable requests, which we dispatch without further modification.

In what follows, we first detail the layers of our pipeline, that is, the prediction layer φ_w and the combinatorial optimization layer f in sections 3.1 and 3.2, respectively. Then, we detail our learning approach in Section 3.3.

3.1 The prediction layer

In general, our pipeline allows using any differentiable predictor φ_w, which receives the set of requests x_t as input and outputs a corresponding vector of prizes θ_t. Formally,

$$\theta_t = \varphi_w(x_t). \tag{1}$$

The predictor φ_w we use in this challenge first transforms the set of requests x_t into a set of feature vectors each representing request and epoch attributes, e.g., normalized request time windows, the number of must-dispatch requests in close vicinity, and the number of remaining epochs, and then feeds each feature vector into a neural network to retrieve the prize for the respective requests.

3.2 The combinatorial optimization layer

We propose a state-of-the-art metaheuristic, which we will refer to as PC-HGS, based on the HGS algorithm [4] to solve the PC-VRPTW. In what follows, we briefly describe our main extensions to the algorithm and refer to our supplemental material [3] for a thorough discussion.

Accounting for optional requests. Solutions in the PC-HGS carry a *request set*. Requests contained in this set must be served. The converse applies to requests not in the set.

Mutation mechanic. We introduce two new mutation operators that modify the request set explicitly. The first operator either prunes 15% of the requests set or inserts 15% of the currently unserved requests into the request set based on the result of a coin toss. The second operator (`optimize_requests`) optimises the request set of a given solution. Specifically, this operator first prunes any unprofitable requests from the solution, and then inserts any unserved but profitable request. The operator perturbs insertion and removal costs with a random factor drawn uniformly from $[0.8, 1.2]$.

Local Search. We extend the local search procedure with two additional operators that remove and insert a request into the request set, respectively.

Initialisation. We apply a pre-processing technique to account for extreme weights encountered during the learning procedure. Specifically, we maintain a set of certainly profitable and certainly unprofitable requests based on the maximum and minimum detours required, respectively. This procedure allows to account for must-dispatch requests during the evaluation phase.

We seed the initial population with solutions of previous learning epochs. Specifically, we add a new construction heuristic that generates additional candidates by applying `optimize_requests` to the solutions of previous learning epochs. The construction heuristic further improves these candidates with a local search procedure.

3.3 Learning approach

In order to achieve good performance with our pipeline, we need to find weights w of the neural network in φ_w, such that the routes y_t that our pipeline outputs provide good solutions, even for instances not considered during training. For this purpose, we propose an approach that appears counterintuitive from the perspective of stochastic optimization and try to imitate the decisions taken by the (anticipative) oracle strategy provided by the organizers.

Training dataset Our training dataset comprises a set of 20 dynamic instances comprising between 4 and 7 epochs generated from a randomly sampled set of static instances. To keep the training time reasonably short ($\approx$ one day), we limit the computational time of the PC-HGS to 60 seconds and modify the instance generation procedure slightly: we sample 50 instead of 100 requests at the start of each epoch. This ensures that the PC-HGS converges within the set time limit even for large epoch instances, i.e., those obtained from poor choices of predictor weights w causing excessive postponing. Our results indicate that pipelines trained on these smaller instances generalize well to larger instance sizes.
To compute the loss function outlined in the following section, we further solve each dynamic instance with the oracle strategy and label each epoch with the routes output by the oracle.

Learning problem and loss function Classic supervised learning settings formulate the learning problem based on a loss function $\mathcal{L}$ that measures output quality. Finding the best $w^\star$ based on the known training data then corresponds to minimizing $\mathcal{L}$. This optimization problem is usually solved using a gradient algorithm that relies on automatic differentiation to backpropagate gradients through the pipeline. In our setting, a natural loss corresponds to the difference of the obtained solutions to the oracle's solutions with respect to the combinatorial optimization layer. More precisely, this natural loss captures the difference in the objective value of the obtained routes and the oracle's routes, evaluated with the predicted prizes θ. However, this natural loss is not smooth and allows for an optimal

solution with $\theta_t = 0$. To tackle this problem, we use a regularization approach for combinatorial optimization layers of the form $\theta \mapsto \operatorname{argmax}_y \theta^\top y$ recently proposed in Berthet et. al [5] and perturb the input θ_t to the combinatorial optimization layer with an additive gaussian noise $Z \sim \mathcal{N}(0, 1)$:

$$\widehat{f}_\varepsilon : \theta_t \mapsto \mathbb{E}[f(\theta_t + Z)] \tag{2}$$

In practice, the expectation in (2) is intractable, but can be approximated by Monte-Carlo sampling. This perturbed combinatorial layer allows to define a Fenchel-Young loss [6], which is smooth, convex, and has meaningful gradients. We extend the approach from Berthet et. al [5] to account for heuristic solutions in the combinatorial layer.

Experiments To calculate the gradient of one learning step we calculated the expectation over 20 perturbation vectors, such that each training epoch requires to solve a total of 20×110 PC-VRPTW instances. We used the *ADAM* optimizer to optimize neural network weights w.

4 Results

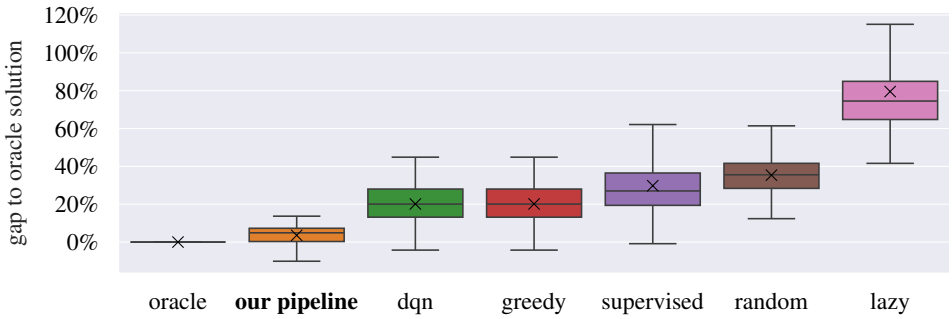


Figure 3: Box plot comparing the gap in % to the oracle solution. Crosses give mean values.

Figure 3 shows the performance of our pipeline in comparison to the provided benchmarks. Our pipeline solution has an average gap of 3.55% to the oracle's solution. The second best approaches, i.e., greedy and dqn, achieve a gap of 20.15%.

Acknowledgments We'd like to thank Axel Parmentier and Maximilian Schiffer for guidance and fruitful discussions.

References

[1] W. Kool, L. Bliek, D. Numeroso, *et al.*, "The EURO meets NeurIPS 2022 vehicle routing competition," 2022.

[2] W. Kool, J. O. Juninck, E. Roos, *et al.*, "Hybrid genetic search for the vehicle routing problem with time windows: A high-performance implementation," 2022.

[3] K. Jungel, L. Baty, and P. Klein, *Team kléopatra: Supplemental material*. [Online]. Available: https://github.com/tumBAIS/euro-meets-neurips-2022.

[4] T. Vidal, *Hybrid Genetic Search for the CVRP: Open-Source Implementation and SWAP* Neighborhood*, arXiv:2012.10384 [cs], Oct. 2021. DOI: 10.48550/arXiv.2012.10384. [Online]. Available: http://arxiv.org/abs/2012.10384 (visited on 09/30/2022).

[5] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach, "Learning with Differentiable Perturbed Optimizers," en, *arXiv:2002.08676*, Jun. 2020. [Online]. Available: http://arxiv.org/abs/2002.08676 (visited on 02/01/2022).

[6] M. Blondel, A. F. Martins, and V. Niculae, "Learning with Fenchel-Young losses.," *J. Mach. Learn. Res.*, vol. 21, no. 35, pp. 1–69, 2020.