The EURO Meets NeurIPS 2022 Vehicle Routing Competition

Wouter Kool[*] WOUTER.KOOL@ORTEC.COM
ORTEC

Laurens Bliek L.BLIEK@TUE.NL
Eindhoven University of Technology

Danilo Numeroso DANILO.NUMEROSO@PHD.UNIPI.IT
Università di Pisa

Yingqian Zhang YQZHANG@TUE.NL
Eindhoven University of Technology

Tom Catshoek T.CATSHOEK@TUDELFT.NL
Delft University of Technology

Kevin Tierney KEVIN.TIERNEY@UNI-BIELEFELD.DE
Bielefeld University

Thibaut Vidal THIBAUT.VIDAL@POLYMTL.CA
Polytechnique Montréal

Joaquim Gromicho JOAQUIM.GROMICHO@ORTEC.COM
ORTEC

Editor: Marco Ciccone, Gustavo Stolovitzky, Jacob Albrecht

Abstract

Solving vehicle routing problems (VRPs) is an essential task for many industrial applications. Although VRPs have been traditionally studied in the operations research (OR) domain, they have lately been the subject of extensive work in the machine learning (ML) community. Both the OR and ML communities have begun to integrate ML into their methods, but in vastly different ways. While the OR community primarily relies on simplistic ML methods, the ML community generally uses deep learning, but fails to outperform OR baselines. To address this gap, the *EURO Meets NeurIPS 2022 Vehicle Routing Competition* brought together the OR and ML communities as a joint effort of several previous competitions to solve a challenging VRP variant on real-world data provided by ORTEC, a leading provider of vehicle routing software. The challenge focuses on both a "classic" deterministic VRP with time windows (VRPTW) and a dynamic version in which new orders arrive over the course of a day. Over 50 teams submitted solutions over a 13-week submission period, battling for not only the best performance on the competition problems, but also for the longest dominance of the leaderboard. The goals of the competition were achieved, with both state-of-the-art techniques in OR and ML playing a significant role in several of the winning submissions.

Keywords: Machine learning, combinatorial optimization, stochastic optimization, vehicle routing problem with time windows, dynamic dispatch problem

[*] Corresponding author

1. Introduction

Every day, millions of trucks and other vehicles perform deliveries or services around the globe. Finding optimal routes for these vehicles is critical for companies to reduce their costs and environmental impact. The vehicle routing problem (VRP), a generalization of the well-known traveling salesperson problem (TSP), has thus been the focus of significant research activity. In the VRP, the goal is to find routes for a set of vehicles and a set of customers such that every customer is visited exactly once by one of the vehicles and the total distance traveled (or travel cost incurred) is minimized. Numerous extensions of the VRP have been studied in the literature (Vidal et al., 2020), for example, including time windows (known as the VRPTW), which require deliveries at customers to happen during specific periods of time, as well as stochastic variants (e.g., with uncertain travel durations between customers) or dynamic variants that reveal requests over time (Pillac et al., 2013).

VRPs have been traditionally studied in the field of operations research (OR), and solution methods primarily rely on searching the space of feasible solutions, either using exact methods or heuristics (Laporte et al., 2014; Poggi and Uchoa, 2014; Costa et al., 2019). In recent years, the problem has also attracted the attention of machine learning (ML) researchers (Nazari et al., 2018; Kool et al., 2018; Kwon et al., 2020; Hottung et al., 2020; Xin et al., 2021b; d O Costa et al., 2020; da Costa et al., 2021; Xin et al., 2021a; Kool et al., 2022b; Bai et al., 2021; Choo et al., 2022), who have developed a variety of methods using deep neural networks (DNNs) to produce VRP solutions by training either on (near-)optimal solutions or using reinforcement learning. While these techniques vary in their problem-specific components and general search schemes, most of them do not currently outperform state-of-the-art OR solution strategies, which are typically based on fast local searches with additional metaheuristic strategies to guide the exploration of the solution space.

The ML and OR communities have progressed quite differently in their work on vehicle routing problems, sometimes using different solution evaluation practices, hardware, and benchmarks, putting different emphases on speed versus solution quality, with different viewpoints regarding application-oriented versus method-oriented work. Given this, the *EURO Meets NeurIPS 2022 Vehicle Routing Competition* was designed to unite the two communities to work on the same routing problems and benchmarks, thus fostering cross-pollination between the areas. The competition was a joint effort of the VeRoLog competition (Gromicho et al., 2019) of the EURO working group on Vehicle Routing and Logistics Optimization (VeRoLog), the IJCAI AI4TSP competition (Zhang et al., 2023) of the EURO working group on Data Science meets Optimisation (DSO), and the DIMACS implementation challenge (Archetti et al., 2021) organized by the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS). The competition involved solving both a static and dynamic version of the VRPTW with the goal of offering both the OR and ML communities opportunities to exploit learning algorithms and advanced methods. Besides the three previous competitions, examples of other related recent competitions are the Amazon Last-Mile Routing Challenge (Merchán et al., 2022) and the CEC-12 Competition on Electric Vehicle Routing Problem (Mavrovouniotis et al., 2020).

This paper provides an overview of the competition and the lessons learned. We define the competition problem in Section 2, followed by a description of the competition setup in

Section 3. We include a summary of the approaches by the finalist teams in Section 4. The final results are discussed in Section 5, and a retrospective is presented in Section 6.

2. Problem statement

The VRPTW has long been the focus of OR researchers (Kallehauge et al., 2005) due to its appearance in multiple application settings and problem variants, such as home healthcare routing (Fikar and Hirsch, 2017), technician routing (Paraskevopoulos et al., 2017), and integrated truck-drone deliveries (Chung et al., 2020) to name a few. We selected the VRPTW and a dynamic variant of it for the competition as these problems are canonical and easy to model, but locating low-cost solutions is challenging due to the time-window constraints on customer visits and the vehicle's capacity restrictions. Besides this, static and dynamic versions of the VRPTW had not received significant work from the ML community, making them prime targets for bringing the ML community closer to real-world applications and encouraging the OR community to use more advanced ML methods. In the following, we formalize the VRPTW, starting with the standard (static) version, followed by its dynamic variant.

Static problem setting The VRPTW is a constrained variant of the capacitated VRP (CVRP) in which each customer requires delivery within a specified interval of time, called a *time window*. The static problem is defined as follows. A VRPTW problem instance considered in this competition consists of n locations (a depot and a set of $n-1$ customers), an $n \times n$ matrix D specifying the time (in seconds) to travel between each pair of locations (it represents actual road driving times in seconds and is not Euclidean nor symmetric), product demand quantities q_i for each customer i, and the maximum quantity Q that a vehicle can carry, also known as the vehicle's capacity.

Each customer, represented by a node i, is also characterized by a service duration s_i, denoting the time it takes to serve customer i, and a time window $[e_i, l_i]$, where $e_i < l_i$, within which service/delivery must begin. A vehicle is allowed to arrive at a customer location before the beginning of the time window, but it must wait until the beginning of the delivery window to start the delivery. A delivery cannot be started after the time window closes. In this way, the competition considers the time-window constraints to be hard constraints. There is no limit on the number of available vehicles.

A feasible solution to the VRPTW consists of a set of routes that start and end at the depot, such that each customer is visited on exactly one route within its specified time window, and the total demand assigned to a route does not exceed the vehicle capacity Q. The objective is to find a feasible solution that minimizes the total driving duration of all routes. Note that the number of vehicles, the service times and waiting times are not part of the objective in this competition. This problem definition follows the convention used in the recent DIMACS VRPTW challenge, except that the matrix D is not Euclidean, and the number of vehicles is unlimited. With these conventions, the feasibility of the problem is guaranteed even in the dynamic setting (see next section).

Dynamic problem setting For the dynamic variant of the problem, requests arrive at different one-hour epochs during the day. The solver can choose which requests to dispatch in the current epoch, for which it must create feasible routes, and which to postpone to consolidate with new requests that may arrive later. Some requests are "must-go" requests that must be dispatched during the current epoch, as delaying them until the next epoch will render their time window infeasible. In the last epoch, all requests are must-go. Locations, demands, service times, and time windows of requests are sampled for each epoch uniformly from the data of a static VRPTW instance, after which requests with infeasible time windows are filtered.

3. Competition details

The competition consisted of a qualification phase, where results were shown on a public leaderboard, and a final phase, where the top 10 submissions from the qualification phase were tested on 100 hidden test instances. Submissions in the qualification phase had a time limit of 3 to 8 minutes for the static problem, depending on the size of the instance, and 1 minute per epoch for the dynamic problem. In the final phase, this time limit was increased to 5 to 15 minutes and 2 minutes per epoch respectively.

In the quickstart repository[1], a public dataset with 250 static VRPTW instances was provided. These are real instances provided by ORTEC, using data from a US-based grocery delivery service that have been anonymized. The instances contain between 200 and 900 customers. From these static instances, dynamic instances are created by the environment by sampling 100 requests (after which requests with infeasible time windows are filtered) from the static set of customers during a number of epochs that is between 5 and 9, depending on the instance. The public instances and the provided environment can be used to develop a solver or train a machine-learning model.

All instances were provided in the VRPLib format, following the convention used by LKH3[2]. As the static problem is a special case of the dynamic problem, solvers were evaluated on static and dynamic problem variants using the same protocol. For this protocol, we used an environment that we implemented in Python and which follows the OpenAI gym interface (Brockman et al., 2016). Here, the environment accepts an action and returns an observation, a reward, a flag indicating whether the environment is done, and extra information. This terminology comes from reinforcement learning, where the solver can be seen as an agent interacting with an environment. In this terminology, the observation defines the VRPTW/request selection problem during each epoch, an action is a solution defining a set of routes to dispatch, and the reward is the negative of the cost (driving duration) of the solution (set of dispatched routes) for that epoch. Solving a single problem to completion (with multiple epochs) can be referred to as an episode of the environment.

3.1. Baselines

We provided a number of baselines for the participants to use while developing their solver.

Static baseline. As a baseline for the static variant, we provided an implementation of the Hybrid Genetic Search (HGS) algorithm (Vidal et al., 2012; Vidal, 2022), which combines a genetic algorithm with local search. HGS and its extensions have been used to produce state-of-the-art results for many vehicle routing problem variants (Vidal et al.,

1. https://github.com/ortec/euro-neurips-vrp-2022-quickstart
2. http://webhotel4.ruc.dk/~keld/research/LKH-3/

2013, 2014; Vidal, 2022). Specifically, the baseline that we provided was an extension of HGS-CVRP (Vidal, 2022), that was used by Kool et al. (2022a) to win the VRPTW track of the DIMACS implementation challenge and can thus be considered state-of-the-art. This implementation uses the strategy of Vidal et al. (2013) to efficiently evaluate time-window constraints through the search. Additionally, compared to HGS-CVRP, it includes an additional crossover from Nagata et al. (2010), dynamic parameters for population- and neighborhood-size management, along with some code optimizations.

Dynamic policies. All the vehicle routing variants solved with HGS until now were static. Extending a static solver to deal with dynamic requests required, in our situation, choosing which requests should be served on any given epoch. Such an extension is non-trivial and requires additional policies for customer selection.

To select the requests to be dispatched in each epoch, we provided three initial simple policies in the quickstart code base: (1) **Greedy**: every available request is going to be dispatched within the current epoch. This means that all requests are included in HGS route optimization; (2) **Lazy**: only must-go requests will be dispatched. Must-go requests are requests that cannot be delayed to further epochs because of time windows violation (indicated by the environment); and (3) **Random**: requests to be dispatched in the current epoch are selected randomly: each request is dispatched with 50% probability, independently. Note that must-go requests will always be included in order to have feasible solutions.

Additionally, during the competition, we added two additional baselines using machine learning: (1) **Supervised**: an oracle solver (using future information) was used to create examples to train a neural network to predict requests dispatched by the oracle, and (2) **DQN**: a Deep Q-Network (DQN) (Mnih et al., 2013) agent was trained using reinforcement learning to select requests, by repeatedly interacting with the environment. The machine learning baselines were for illustrative purposes and did not outperform the greedy baseline. Together, the baselines provided the initial results for comparison and were included in the public leaderboard, though not as official participants.

3.2. Scoring and submission system

Teams were asked to submit code that could generate a solution given an instance: for example, a hand-designed heuristic, or a policy trained with reinforcement learning. This allowed us to test the submitted code on hidden test instances, which we used to determine the final winners based on the solution quality on those instances. As described in the problem statement, the score for each instance is calculated as the total time that all vehicles spent driving (i.e., excluding service or waiting times). Given this, smaller scores were preferable.

The scores per instance were averaged for the hidden instances of the static problem for all teams, and also for the hidden instances of the dynamic problem for all teams, leading to a separate ranking of the teams on both problem variants. Then, the two rankings of each team were averaged, leading to a final ranking for each team for the whole competition. This ensured both problems were weighted equally regardless of the actual scores. In case of a tie, the scores themselves were averaged instead of the rankings.

For enabling code submission, we used the Codalab Competition[3] platform, and we provided participants with instructions in the quickstart repository. The code was executed inside an Apptainer/Singularity container, based on a Dockerfile[4] to ensure the same environment for all participants that they could reproduce for testing. The environment had a single CPU core (Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz), a single GPU (Nvidia TitanRTX 24GB) and 32 GB of RAM.

3.3. Prizes

We provided the following awards to winning participants, in total over 5000 €.

1. Best performance: 2022/750/500 € for the 1st/2nd/3rd place
2. Jury prize: 250 €
3. Young talent prize: 250 €
4. Leaderboard prizes: 1300 € total

The top three prizes were scored according to the leaderboard criteria described in Section 3.2 on the hidden dataset in the finals. To further encourage innovative ideas, even if they did not lead to top performance, the 'jury prize' was awarded at the organizers' discretion to a method providing novel methodological ideas. For this award, we sought an approach that (1) offered a novel solution approach, (2) used ML, (3) achieved good performance within the time limit, and (4) did not win first place.

The 'young talent prize' was awarded to the best-performing team consisting only of students and/or PhD candidates. Finally, the leaderboard prizes were 50/30/20 € awarded to the number 1/2/3 on the leaderboard on a weekly basis, to encourage early competition.

4. Summary of team contributions

Each of the finalist teams has described their approach in a short paper and presented it during the final workshop (see Section 6). Both the paper and the presentation slides have been published on the competition website[5]. We provide an overview of the finalist's contributions here.

Kleopatra Jungel et al. (2022) improved HGS for the static variant by reducing some of the randomization steps and implementing a caching mechanism for unchanged routes across local search iterations. For the dynamic problem, they implemented a parametrizable prize-collecting variant of HGS (PC-HGS) that jointly optimizes the selection of requests to dispatch and the routes, based on a set of prizes that indicates the preferences of requests to dispatch in the current epoch. The optimal prize is unknown a priori, but a machine learning model is trained to predict prizes, using a loss function that encourages the PC-HGS results to be close to the anticipative oracle solution (see baselines).

3. https://codalab.lisn.upsaclay.fr/
4. https://github.com/TCatshoek/codalab-dockers/blob/master/legacy-gpu/Dockerfile
5. https://euro-neurips-vrp-2022.challenges.ortec.com/#results

OptiML van Doorn et al. (2022) simplified the baseline HGS implementation by removing the ordered crossover, initial constructions, and growing neighborhood/population components introduced in Kool et al. (2022a). They added a diversity criterion to the parent selection step, made code optimizations to the local search, and extensively tuned the parameters. For the dynamic problem, their approach simulates multiple scenarios of future requests, for which they optimized the routes using their solver as oracle (see baselines). May-go requests that are frequently combined with must-go requests in the simulations are marked as 'must-go' as well, and this process is repeated several times to define the final set of requests to dispatch. This set is then optimized using the static solver, and all resulting routes with at least one (original) must-go request are dispatched.

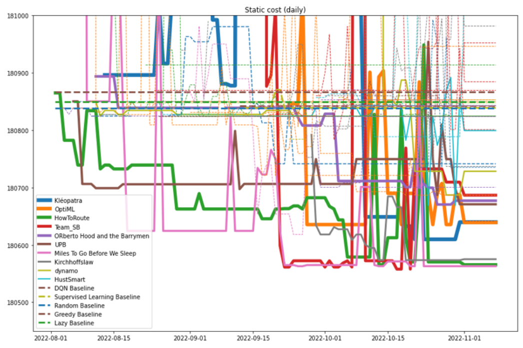
Team_SB Kwon et al. (2022) tuned the HGS parameters and generalized the binary tournament to a k-way tournament, where k is adapted automatically such that higher-fitness individuals are selected more frequently if a new best solution was found recently. Furthermore, to encourage diversity, individuals were removed from the population after generating a certain number of offspring and a warm-restart mechanism. The dynamic variant used two neural networks: one to predict priorities for requests and one to select requests that must be dispatched. These were used in the HGS algorithm with a modified objective that can select requests, weighing the priorities against the additional distance. Both networks were trained using reinforcement learning.

HustSmart Li et al. (2022) added the EAX crossover to HGS and added the SISR algorithm to refine the solution every time a new best solution is found. For the dynamic problem, they computed a heuristically defined penalty score for each request and used OR-Tools to solve the VRPTW while allowing to drop visits at the cost of the penalty. This resulted in a set of requests, for which routes were then further optimized using the HGS algorithm.

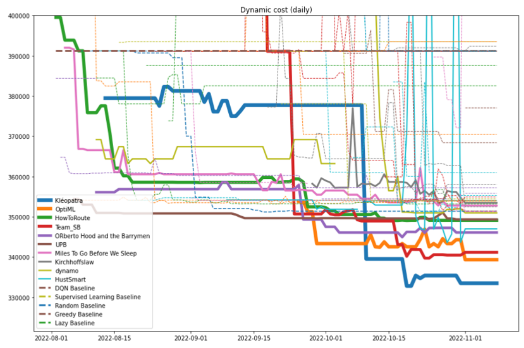
Miles To Go Before We Sleep Reese et al. (2022) made several performance improvements to the HGS code for the static problem and tuned the parameters on a diverse subset of the training instances. For the dynamic variant of the problem, they implemented two heuristics for selecting requests that would dispatch may-go orders depending on how (spatially) close they were to must-go orders. The routes given the selection were optimized using HGS.

ORBerto Hood Hildebrandt et al. (2022) tuned the HGS parameters for the static variant of the problem and optimized the set of requests to dispatch for the dynamic variant over a set of sampled future scenarios. For this, they defined a single integrated problem that takes into account all scenarios, which they solved using a column-generation procedure. The routes for the resulting set of requests were optimized using HGS.

HowToRoute Mesa et al. (2022) reduced the frequency of the intensification procedure and implemented an adjusted crossover in the HGS algorithm. For the dynamic variant, they dispatched may-go requests that are close to must-go requests and found a solution using HGS. Then, they applied a method to remove may-go requests from routes (to be postponed) if this reduced the average cost of requests in the route. The final selection was further optimized using HGS.



(*a*) Static cost



(*b*) Dynamic cost

Figure 1: Static and dynamic results during the qualification phase of the competition.

Kirchhoffslaw Ao and Zong (2022) tuned the HGS parameters for the static variant of the problem and trained a neural network using reinforcement learning to select requests to dispatch for the dynamic variant. Actions correspond to whether or not an order should be selected and the final reward was computed by comparing the final cost after running HGS on the selected requests for each epoch to the final costs of the greedy baseline method.

UPB Dieter (2022) focused on the dynamic variant of the problem and implemented a regret policy that uses the available information about the request distribution to analytically compute a regret value for each may-go request: the possible improvement that would be missed when the request would be dispatched immediately. Requests with a regret value below a threshold were dispatched immediately and optimized using HGS.

dynamo Ghannam and Gleixner (2022) added an additional HGreX crossover to HGS for the static problem. For the dynamic variant, they adapted HGS to allow including a subset of the requests in the optimization, where the average cost (total driving duration divided by number of requests) over the orders was used as an objective. A penalty ensured must-go orders were dispatched immediately, and an additional lateness term favored including requests for which time windows ended sooner.

5. Results

Figure 1 shows the results on the public leaderboard for the static and dynamic problem variants for different teams during the qualification phase. As results could slightly vary between different evaluations, the charts are not strictly decreasing, but for both problem variants, we see a clearly improving trend in the results as the competition progresses. The average of a team's rank on the static and dynamic problem determined the overall rank on the leaderboard, visualized in Figure 2. The number of participants on the leaderboard steadily increased during the competition, and some of the finalists only made it to the top 10 in the public leaderboard towards the end of the qualification phase.

The top 10 of the public leaderboard defined the set of finalists, which were thoroughly evaluated during the final phase of the competition. This was done using a hidden test set,

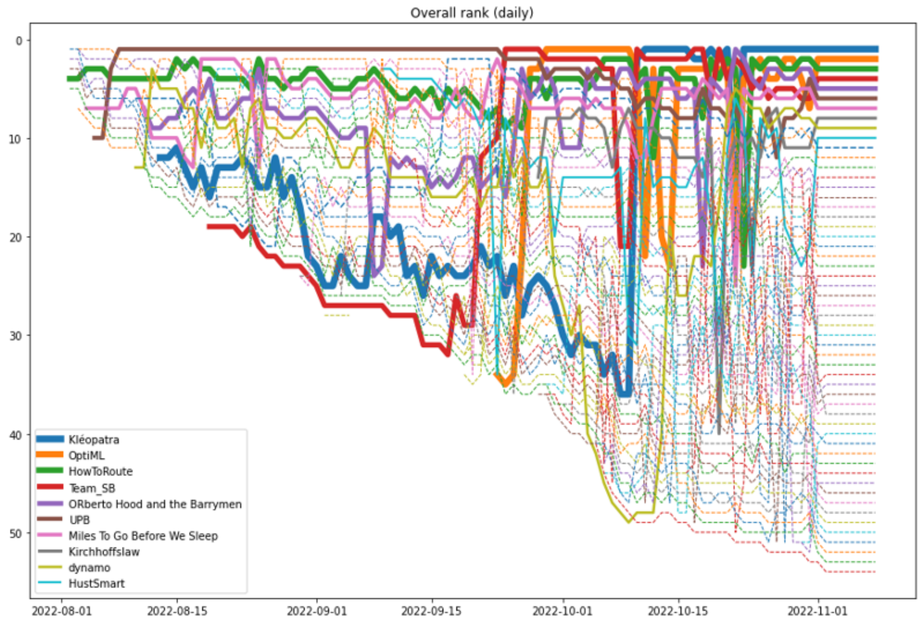


Figure 2: Leaderboard rank during the qualification phase of the competition.

which consisted of 100 instances, which were solved for both the static and dynamic variants, with longer runtimes (see Section 3). Table 1 presents the results of the finalists, where teams were ranked by the averages of their rank on the static and the dynamic variant of the problem. Team Kleopatra (Jungel et al., 2022) obtained the first place, ranking first for the dynamic problem and second for the static problem, winning 2022 euros. Team OptiML (van Doorn et al., 2022) ranked second overall, by ranking first on the static problem and third on the dynamic problem and won 750 euros. Team_SB (Kwon et al., 2022) won the third place and 500 euros, by ranking third on the static problem and second on the dynamic problem. In general, we observed a strong correlation between each team's performance on the static and the dynamic variant of the problem, suggesting that teams were able to divide their efforts well between both problems.

The jury prize was awarded to Team_SB as they got excellent results using an innovative machine learning-based approach.[6] The young talent prize was awarded to team Kleopatra. Finally, leaderboard prizes for being in the top 3 of the public leaderboard during the qualification phase were awarded to teams UPB, Kleopatra, Team_SB, 4pd_vrp (not qualified), and OptiML.

All of the real-world data that was used during the competition, including the hidden dataset used for final evaluation, has been made available under the CC BY-NC 4.0 license in the GitHub repository of the competition. We hope this dataset will be continued to be used as a real-world benchmark. For verifiability and reproducibility purposes, all of the settings used for the final evaluations and the individual solutions found by the finalist

6. Team Kleopatra got even better results using a machine learning-based approach but was not eligible for the jury prize as they won the 1st place overall.

Table 1: Final results of the EURO Meets NeurIPS 2022 Vehicle Routing Competition.

Rank	**Team**	**Static (rank)**	**Dynamic (rank)**	**Avg. (avg. rank)**
1	Kleopatra (Jungel et al., 2022)	157200.61 (2)	348831.56 (1)	253016.1 (1.5)
2	OptiML (van Doorn et al., 2022)	157188.67 (1)	359270.09 (3)	258229.4 (2)
3	Team_SB (Kwon et al., 2022)	157214.33 (3)	358161.36 (2)	257687.8 (2.5)
4	HustSmart (Li et al., 2022)	157227.24 (5)	361803.57 (4)	259515.4 (4.5)
5	Miles To Go Before We Sleep (Reese et al., 2022)	157224.13 (4)	369098.13 (7)	263161.1 (5.5)
6	ORberto Hood and the Barrymen (Hildebrandt et al., 2022)	157301.21 (9)	362481.13 (5)	259891.2 (7)
7	HowToRoute (Mesa et al., 2022)	157251.06 (7)	369797.03 (8)	263524.0 (7.5)
8	Kirchhoffslaw (Ao and Zong, 2022)	157249.31 (6)	370670.53 (9)	263959.9 (7.5)
9	UPB (Dieter, 2022)	157322.36 (10)	367007.49 (6)	262164.9 (8)
10	dynamo (Ghannam and Gleixner, 2022)	157287.45 (8)	90341072.98 (10)	45249180.0 (9)

solvers have been published as well, together with a notebook that reproduces the final results table. Most of the finalist teams have published their code, which is linked on the competition website.

6. Retrospective

In the end, close to 150 teams registered via the website, of which 50 teams made a final submission. During the qualification phase, a total of over 800 submissions have been made. 180 people joined the Slack workspace, where around 4500 messages were exchanged, so this was actively used to communicate with and amongst participants. Among the participants were researchers from both the operations research and machine learning communities, with many teams from academia as well as some industry participants. The participants came from all over the world.

The concluding workshop at NeurIPS took place as a virtual meeting on December 7th, 2022. It was attended by around 30 people, including the competition organizers and members of the ten finalist teams. The recording is available on the competition website.

In hindsight, the (real world) dataset for the static problem contained many instances that were too small to permit improvements over the HGS baseline, since optimal or near-optimal solutions would always be found in a short CPU time. Nonetheless, this part of the challenge still led to useful methodological and practical developments. In particular, the software developed by OptiML, the team ranking first on the static variant, has been released as PyVRP, a state-of-the-art open source VRPTW solver that can be easily used from Python. This will help ML researchers to build on and contribute to state-of-the-art OR solvers for vehicle routing problems in the future.

Compared to the static variant, very significant progress beyond the baseline strategies has been made for the dynamic variant of the problem. All the best approaches used machine learning as a key component of their winning policies. This shows that machine learning is not only promising for the future, but can already compete with alternative approaches on dynamic optimization problems.

The motivation for having two problem variants, but scoring participants by their combined performance, was to reward versatile approaches that would work well for both variants, as well as ensure that effort would go into both improving static performance and handling the dynamic aspect of the problem. In practice, most teams implemented specific solution strategies for both variants, and in a few cases, only tuned the baseline algorithm parameters for the static variant.

From a technical perspective, the Codalab platform enabled us to evaluate code submissions, but we needed many adjustments to make it work for our use case. As our evaluation required GPUs, we had to attach our own compute workers, and while Codalab provides a Docker image to support this use case, it assumes running dedicated workers 24/7, which is wasteful of (GPU) compute resources as most of the time the submission queue would be empty. In order to evaluate submissions as jobs on a shared computer cluster, we had to 1) switch to Apptainer/Singularity (as Docker required root privileges which were not available), 2) adapt the worker script to terminate when the queue is empty, and 3) set up a separate external monitoring script to launch jobs when new submissions were in the queue.

Apart from evaluation, there were some additional challenges: the Codalab leaderboard did not compute the ranks correctly, so we published a correct leaderboard that would automatically update on our own competition website. In the future, given the combination of challenges, we would investigate other platforms such as Kaggle or AICrowd. Nonetheless, the submission platform with a real-time leaderboard played an essential role in the competitions, with teams telling us they refreshed the page multiple times a day. The public results encouraged teams to do better and actively participate early on during the competition, where the leaderboard prize was an additional incentive.

7. Conclusion

As the competition organizers, we were pleased to see that this competition successfully brought AI and OR researchers together, leading to new approaches that advance the state-of-the-art ML and OR techniques for solving both static and dynamic versions of VRPTW. In addition, we have made available the instances in the competition as a real-world benchmark that researchers can use for testing new approaches. All the developed baseline methods and the implementations of many participants are also publicly available. In the future, the organization team will continue the effort to bring the communities of OR and AI together to solve relevant, real-world optimization problems.

Acknowledgments

We would like to thank ORTEC for sponsoring the majority of the prize pool, as well as both the EURO VeRoLog and DSO working groups which provided additional support to enable the jury and young talent prizes. The compute infrastructure used during the competition was generously sponsored by the HPC lab of the Eindhoven University of Technology. Finally, we would like to thank both the EURO and NeurIPS conferences for their support while organizing this competition.

References

Wenxuan Ao and Zefang Zong. Hybrid genetic search and learn-to-select for the EURO meets NeurIPS 2022 vehicle routing competition. *The EURO Meets NeurIPS 2022 Vehicle Routing Competition*, 2022. URL https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/kirchhoffslaw.pdf.

Claudia Archetti, Nicholas Kullman, Catherine McGeoch, Jorge Mendoza, Panos Pardalos, Mauricio Resende, Eduardo Uchoa, and Thibaut Vidal. The 12th DIMACS implementation challenge, 2021. URL http://dimacs.rutgers.edu/programs/challenge/vrp/.

Ruibin Bai, Xinan Chen, Zhi-Long Chen, Tianxiang Cui, Shuhui Gong, Wentao He, Xiaoping Jiang, Huan Jin, Jiahuan Jin, Graham Kendall, et al. Analytics and machine learning in vehicle routing research. *arXiv preprint arXiv:2102.10012*, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

Jinho Choo, Yeong-Dae Kwon, Jihoon Kim, Jeongwoo Jae, André Hottung, Kevin Tierney, and Youngjune Gwon. Simulation-guided beam search for neural combinatorial optimization. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=tYAS1Rpys5.

Sung Hoon Chung, Bhawesh Sah, and Jinkun Lee. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research*, 123:105004, 2020.

Luciano Costa, Claudio Contardo, and Guy Desaulniers. Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985, 2019.

Paulo R d O Costa, Jason Rhuggenaath, Yingqian Zhang, and Alp Akcay. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In *Asian conference on machine learning*, pages 465–480. PMLR, 2020.

Paulo da Costa, Jason Rhuggenaath, Yingqian Zhang, Alp Akcay, and Uzay Kaymak. Learning 2-opt heuristics for routing problems via deep reinforcement learning. *SN Computer Science*, 2(5):1–16, 2021.

Peter Dieter. A regret policy for the dynamic vehicle routing problem with time windows. *The EURO Meets NeurIPS 2022 Vehicle Routing Competition*, 2022. URL https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/UPB.pdf.

Christian Fikar and Patrick Hirsch. Home health care routing and scheduling: A review. *Computers & Operations Research*, 77:86–95, 2017.

Mohammed Ghannam and Ambros Gleixner. Hybrid genetic search for the dynamic vehicle routing problem. *The EURO Meets NeurIPS 2022 Vehicle Routing Competition*,

2022. URL https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/dynamo.pdf.

Joaquim Gromicho, Pim van't Hof, and Daniele Vigo. The VeRoLog solver challenge 2019. *Journal on Vehicle Routing Algorithms*, 2(1):109–111, 2019.

Florentin D. Hildebrandt, Roberto Roberti, Barrett W. Thomas, and Marlin W. Ulmer. A two-stage set partioning approach for the EURO meets NeurIPS 2022 vehicle routing competition. *The EURO Meets NeurIPS 2022 Vehicle Routing Competition*, 2022. URL https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/ORberto_Hood_and_the_Barrymen.pdf.

André Hottung, Bhanu Bhandari, and Kevin Tierney. Learning a latent search space for routing problems using variational autoencoders. In *International Conference on Learning Representations*, 2020.

Kai Jungel, Léo Baty, and Patrick Klein. Structured learning for the EURO meets NeurIPS 2022 vehicle routing competition. *The EURO Meets NeurIPS 2022 Vehicle Routing Competition*, 2022. URL https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/Kleopatra.pdf.

Brian Kallehauge, Jesper Larsen, Oli BG Madsen, and Marius M Solomon. *Vehicle routing problem with time windows*. Springer, 2005.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2018.

Wouter Kool, Joep Olde Juninck, Ernst Roos, Kamiel Cornelissen, Pim Agterberg, Jelke van Hoorn, and Thomas Visser. Hybrid genetic search for the vehicle routing problem with time windows: a high-performance implementation. *12th DIMACS Implementation Challenge Workshop*, 2022a.

Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep policy dynamic programming for vehicle routing problems. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, 2022b.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

Yeong-Dae Kwon, Jinho Choo, Daseul Bae, Jihoon Kim, and André Hottung. Cost shaping via reinforcement learning for vehicle routing problems. *The EURO Meets NeurIPS 2022 Vehicle Routing Competition*, 2022. URL https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/Team_SB.pdf.

Gilbert Laporte, Stefan Ropke, and Thibaut Vidal. Heuristics for the vehicle routing problem. In Paolo Toth and Daniele Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, chapter 4, pages 87–116. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014. doi: 10.1137/1.9781611973594.ch4.

Yunhao Li, Junjie Zhang, Yuxuan Wang, Wenhan Shao, Zhouxing Su, and Zhipeng Lü. Customer penalty-based heuristic algorithm for the EURO meets NeurIPS 2022 vehicle routing competition. *The EURO Meets NeurIPS 2022 Vehicle Routing Competition*, 2022. URL https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/HustSmart.pdf.

Michalis Mavrovouniotis, Charalambos Menelaou, Stelios Timotheou, Christos G. Panayiotou, Georgios Ellinas, and Marios M. Polycarpou. Benchmark set for the IEEE WCCI-2020 competition on evolutionary computation for the electric vehicle routing problem, 2020.

Daniel Merchán, Jatin Arora, Julian Pachon, Karthik Konduri, Matthias Winkenbach, Steven Parks, and Joseph Noszek. 2021 Amazon last mile routing research challenge: Data set. *Transportation Science*, 2022.

Juan P. Mesa, Alejandro Montoya, Alejandro Uribe, and Camilo Álvarez. A modified HGS for static VRPTW and a neighbors addition-postponement strategy for dynamic VRPTW. *The EURO Meets NeurIPS 2022 Vehicle Routing Competition*, 2022. URL https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/HowToRoute.pdf.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Yuichi Nagata, Olli Bräysy, and Wout Dullaert. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 37(4):724–737, 2010.

Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.

Dimitris C Paraskevopoulos, Gilbert Laporte, Panagiotis P Repoussis, and Christos D Tarantilis. Resource constrained routing and scheduling: Review and research prospects. *European Journal of Operational Research*, 263(3):737–754, 2017.

Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1): 1–11, 2013.

Marcus Poggi and Eduardo Uchoa. New Exact Algorithms for the Capacitated Vehicle Routing Problem. In Paolo Toth and Daniele Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, chapter 3, pages 59–86. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

Brandon M. Reese, Steve Harenberg, Laszlo Ladanyi, Rob Pratt, and Yan Xu. Hybrid genetic search improvements for the EURO meets NeurIPS 2022 vehicle routing competition. *The EURO Meets NeurIPS 2022 Vehicle Routing Competition*,

2022. URL https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/MilesToGoBeforeWeSleep.pdf.

Jasper van Doorn, Leon Lan, Luuk Pentinga, and Niels A. Wouda. Solving a static and dynamic VRP with time windows using hybrid genetic search and simulation. *The EURO Meets NeurIPS 2022 Vehicle Routing Competition*, 2022. URL https://github.com/ortec/euro-neurips-vrp-2022-quickstart/raw/main/papers/OptiML.pdf.

Thibaut Vidal. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers and Operations Research*, 140:105643, 2022.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489, 2013.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673, 2014.

Thibaut Vidal, Gilbert Laporte, and Piotr Matl. A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, 286:401–416, 2020.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of 35th AAAI Conference on Artificial Intelligence*, pages 12042–12049, 2021a.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. NeuroLKH: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem. *Advances in Neural Information Processing Systems*, 34, 2021b.

Yingqian Zhang, Laurens Bliek, Paulo da Costa, Reza Refaei Afshar, Robbert Reijnen, Tom Catshoek, Daniël Vos, Sicco Verwer, Fynn Schmitt-Ulms, André Hottung, et al. The first ai4tsp competition: Learning to solve stochastic routing problems. *Artificial Intelligence*, 319:103918, 2023.