Learning to solve a stochastic orienteering problem with time windows

A. Hottung[a,*], K. Tierney[a]

[a] Bielefeld University, Bielefeld, Germany
andre.hottung@uni-bielefeld.de kevin.tierney@uni-bielefeld.de
[*] Corresponding author

Extended abstract submitted for presentation at the 11th Triennial Symposium on Transportation Analysis conference (TRISTAN XI)
June 19-25, 2022, Mauritius Island

January 14, 2022

Keywords: stochastic optimization, learning to optimize, orienteering problem

1 INTRODUCTION

Recently, there has been increasing attention on learning based approaches for routing problems. Most of these approaches use deep reinforcement learning (DRL) to learn heuristics for standard routing problems (e.g., the traveling salesperson problem (TSP) and the capacitated vehicle routing problem (CVRP)). However, real-world routing problems are usually more heavily constrained and significantly more complex than these standard problems. Furthermore, existing DRL approaches ignore the inherent stochastic nature of (real-world) routing problems and only focus on solving the deterministic routing problems. We propose an DRL approach for a heavily constrained, stochastic routing problem. More precisely, we consider the time-dependent orienteering problem with stochastic weights and time windows (TD-OPSWTW) and demonstrate that our DRL approach is well suited to solve this problem.

Numerous machine learning based approaches for routing problems have been proposed since Vinyals *et al.* (2015) used their newly introduced pointer network architecture to solve the TSP via supervised learning. Most approaches use reinforcement learning for model training because it does not require a training set that includes optimal or high-quality solutions. These approaches construct solutions autoregressively, meaning for each new decision to make, the network accepts its previous decision as input, and usually do not allow for extensive search (Nazari *et al.* (2018), Kool *et al.* (2019), Kwon *et al.* (2020)). Recently, there has also been an increasing focus on methods that combine learned heuristics with problem-independent, high-level metaheuristics that provide search guidance (Hottung & Tierney (2020), Hottung *et al.* (2021b). These methods are able to find solutions of higher quality but are also much slower. In contrast to deterministic routing problems, stochastic routing problems have seen only little attention in the ML literature. However, a few approaches do exist to try to solve routing problems with dynamic customer requests with DRL (Bono (2020), Sultana *et al.* (2021), Basso *et al.* (2022)).

We propose a new DRL approach for the TD-OPSWTW that learns a policy that can be used to construct tours dynamically. At each decision point, the policy takes into account the current time and location of the vehicle, which depends on past, realized travel times, to decide which customer should be visited next. This allows the policy to dynamically react towards the current situation. Our approach was developed as part of the *IJCAI AI for TSP competition*[1] where it won first place in the reinforcement learning track.

[1]https://github.com/paulorocosta/ai-for-tsp-competition

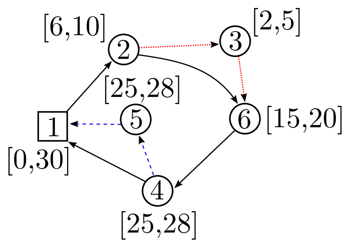

Figure 1 – *A TD-OPSWTW instance with solution (solid, black line). Time windows are shown in square brackets. Alternative solutions are shown by the red, dotted line and blue, dashed line.*

Our approach consists of three components. First, we use the POMO reinforcement learning approach of Kwon *et al.* (2020) to learn one policy per problem size. Next, we use efficient active search (Hottung *et al.*, 2021a) to fine-tune the learned policies for each instance being solved, thus creating an individualized policy for each instance of the test set. Finally, we use Monte-Carlo rollouts to construct the final solutions.

2 TD-OPSWTW

The TD-OPSWTW considers a given a graph, $G = (V, E)$, with nodes V representing customers and a single depot, and edges E between all nodes. Each node i is assigned a location (x_i, y_i) in a Euclidean plane, a time window $(\underline{w}_i, \bar{w}_i)$, and a prize p_i. The goal is to construct a tour starting and ending at the depot that maximizes the collected prizes by visiting nodes. If a node is visited, it must be visited during its time window. If a node is visited too early, the node is not considered visited until the beginning of the time window. If a node is visited too late, a penalty e is incurred. The travel time between nodes, $\hat{t}_{ij}$, is stochastic, but the visit duration at each node is instantaneous. The maximum travel time is given by T, and the penalty p is incurred if it is exceeded. Note that in this version of the orienteering problem, no costs are incurred traveling between nodes.

The AI for TSP competition proposes two versions of the TD-OPSWTW. The first, which does not allow recourse, requires the route to be fixed in advance of realizing the travel times. This is called the *supervised track*. The second problem allows recourse, thus the route may be adjusted at each node according to the realized travel times seen so far. This is called the *reinforcement learning track*. We focus only on the second version of the problem, although we note that our approach actually would have also won the supervised track had it been entered there.

Figure 1 shows an instance and a corresponding solution for the TD-OPSWTW. The time windows for each node are shown in square brackets. Assuming $p_i = 1$ for all nodes, a possible solution is given by the black, solid line. The blue, dashed line shows an alternative end of the tour that would allow earning an extra reward by visiting node 5. If the vehicle arrives at node 4 early enough, visiting node 5 may still be viable within its time window. This decision would be made on the fly as the tour is carried out.

3 SOLUTION APPROACH

Our approach to tackle the TD-OPSWTW consists of three components. We use the POMO reinforcement learning approach proposed by Kwon *et al.* (2020) to learn an initial policy per problem size. We then use efficient active search (Hottung *et al.*, 2021a) to fine-tune the policy to a single instance. Finally, we use Monte-Carlo rollouts for the final solution construction.

POMO We use the POMO approach to learn a policy for each problem instance size. The POMO approach is an end-to-end DRL approach that exploits symmetries in combinatorial optimization problems to encourage exploration during learning. The network architecture of the employed model is based on the transformer architecture and consists of an encoder and a decoder neural network. The encoder is used to construct embeddings that represent the problem instance. The decoder is used to construct a solution based on these embeddings. It is important to note that the embedding generation takes significantly more time than the solution construction from the decoder. However, once the embeddings for one instance have been generated, multiple corresponding solutions can be generated by the decoder.

We slightly adjust the POMO implementation to support the TD-OPSWTW. Each node i of the network is given to POMO as a vector $(x_i, y_i, p_i, \underline{w}_i, \bar{w}_i)$. Additionally, we change the decoder context to include the current time t, the embedding of the current node, and the embedding of the depot. Finally, we forbid actions that correspond to traveling to a node i where $\underline{w}_i > T$ or where $\bar{w}_i < t$ as well as previously visited nodes.

Efficient active search Efficient active search (EAS) (Hottung *et al.*, 2021a) is a method that uses reinforcement learning to fine tune a learned policy for solving a combinatorial optimization problem to a single test instance. EAS iteratively adjusts carefully selected model parameters at test time based on the solutions generated by the learned model. In our approach, the parameters we adjust are the ones in the instance embeddings generated by the decoder as part of the policy. More precisely, we use the trained POMO encoder to generate embeddings for all considered instances. Next, we use reinforcement learning to modify these instance embeddings with the objective to increase solution quality when sampling solutions with the POMO decoder. Note that the travel times between nodes are not fixed during the training process. Instead, the travel times are sampled for each solution construction process. This allows EAS to learn a robust policy that can create high-quality solutions for a wide range of scenarios.

Monte-Carlo rollouts The final solution for an instance is generated using Monte-Carlo (MC) rollouts. At each decision step, we first use the decoder and the fine-tuned embeddings to generate a probability distribution over all possible actions (i.e., all possible nodes that can be visited next). For the five with the highest associated probability values, we then perform Monte-Carlo rollouts each. Each Monte-Carlo rollout starts with the corresponding actions selected in the previous step, and then the solution is completed by sampling the following actions according to the decoder. Once all Monte-Carlo rollouts are complete, the action with the highest average reward is selected. After the action has been chosen, the actual travel times between the nodes is revealed and the process continues with the next decision.

4 RESULTS

We evaluate our approach on the test set of the *IJCAI AI for TSP competition* competition. The test set consists of 1,000 instances partitioned into equal subsets of 20, 50, 100, and 200 customers. First, we train one separate policy model using POMO for each of the four problem sizes. This takes up to several days for full convergence. Note that we use instances generated on the fly for the training. Subsequently, we use EAS to fine tune the learned policies for each test instance separately, which takes up to 30 minutes per instance on an Nvidia V100 GPU. Finally, we use Monte-Carlo rollouts for the final solution construction, performing 600 rollouts for each possible action.

We compare the performance of our final approach (consisting of POMO & EAS & MC) to the performance of only POMO and POMO and & EAS. The results are shown in Table 1. For POMO and POMO & EAS we construct solutions greedily by always choosing the action that was assigned the highest probability value by the network. EAS is able to significantly improve

Table 1 – *Performance on the test set.*

Method	**Average reward**
POMO	10.43
POMO & EAS	10.67
POMO & EAS & MC	**10.77**

the performance of POMO by 2.3%. Monte-Carlo rollouts can further improve the performance by an additional 0.9%. These results show that fine tuning a policy via EAS can improve the performance in this stochastic problem setting.

Our proposed approach won the first place of the reinforcement learning track of the *IJCAI AI for TSP competition*. Table 2 shows the performance of the best three approaches of the competition. Our approach (team *RISE up*) outperforms the second-best approach by 1.8%. We note that our approach without Monte-Carlo rollouts would have also won the competition. A further interesting note, is that our approach would have also tied for first place in the supervised learning track had we submitted it there, despite not using any supervised learning.

Table 2 – *Final leaderboard of the competition (top 3)*

Team	**Average Reward**
RISE up (ours)	10.77341
Ratel	10.58859
ML for TSP	10.39341

References

Basso, Rafael, Kulcsár, Balázs, Sanchez-Diaz, Ivan, & Qu, Xiaobo. 2022. Dynamic stochastic electric vehicle routing with safe reinforcement learning. *Transportation Research Part E: Logistics and Transportation Review*, **157**, 102496.

Bono, Guillaume. 2020. *Deep multi-agent reinforcement learning for dynamic and stochastic vehicle routing problems.* Ph.D. thesis, Université de Lyon.

Hottung, André, & Tierney, Kevin. 2020. Neural Large Neighborhood Search for the Capacitated Vehicle Routing Problem. *European Conference on Artificial Intelligence*, 443–450.

Hottung, André, Kwon, Yeong-Dae, & Tierney, Kevin. 2021a. Efficient Active Search for Combinatorial Optimization Problems. *arXiv preprint arXiv:2106.05126.*

Hottung, André, Bhandari, Bhanu, & Tierney, Kevin. 2021b. Learning a Latent Search Space for Routing Problems using Variational Autoencoders. *International Conference on Learning Representations.*

Kool, Wouter, van Hoof, Herke, & Welling, Max. 2019. Attention, Learn to Solve Routing Problems! *In: International Conference on Learning Representations.*

Kwon, Yeong-Dae, Choo, Jinho, Kim, Byoungjip, Yoon, Iljoo, Gwon, Youngjune, & Min, Seungjai. 2020. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning. *Pages 21188–21198 of:* Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., & Lin, H. (eds), *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc.

Nazari, Mohammadreza, Oroojlooy, Afshin, Snyder, Lawrence, & Takác, Martin. 2018. Reinforcement learning for solving the vehicle routing problem. *Pages 9839–9849 of: Advances in Neural Information Processing Systems.*

Sultana, Nazneen N, Baniwal, Vinita, Basumatary, Ansuma, Mittal, Piyush, Ghosh, Supratim, & Khadilkar, Harshad. 2021. Fast Approximate Solutions using Reinforcement Learning for Dynamic Capacitated Vehicle Routing with Time Windows. *arXiv preprint arXiv:2102.12088.*

Vinyals, Oriol, Fortunato, Meire, & Jaitly, Navdeep. 2015. Pointer Networks. *Pages 2692–2700 of:* Cortes, C, Lawrence, N D, Lee, D D, Sugiyama, M, & Garnett, R (eds), *Advances in Neural Information Processing Systems 28.* Curran Associates, Inc.