Task Planning for Unmanned Vehicle Delivery

ZHU HAOYU
E1632482@u.nus.edu
A0334142R

Introduction

In recent years, unmanned delivery technologies have rapidly penetrated the traditional courier and food-delivery industries. Startups such as RIVR in Switzerland have introduced delivery solutions that integrate autonomous electric vehicles with robot dogs, as shown in Figures 1. Traditional logistics companies, such as SF Express in China, have also begun deploying low-speed autonomous vehicles to handle delivery tasks from warehouses to last-mile stations, as shown in Figure 2.

Fig. 1.

Fig. 2.

In real-world scenarios, however, the number of delivery requests typically far exceeds the available unmanned delivery resources, and a large portion of orders still relies on human couriers. Therefore, our goal is to make use of a limited number of unmanned vehicles (UVs) to complete as many delivery tasks as possible. Doing so not only improves overall delivery efficiency but also reduces the workload of human couriers.

This type of problem can be formulated as a classical Orienteering Problem, in which an agent must traverse a set of target locations—each associated with different rewards—while constrained by a limited travel budget, with the objective of maximizing the total collected reward. However, in large scale problems scenarios, solving such decision-making and optimization problems typically requires balancing solution quality against computational efficiency. In large-scale delivery scenarios, search-based algorithms and conventional solvers often suffer from an exponential increase in computational cost as the number of tasks grows. Meanwhile, delivery demands are highly dynamic, requiring algorithms that can produce solutions rapidly.

Existing studies have shown that learning-based approaches can substantially reduce computation time while maintaining competitive solution quality. Nevertheless, a major challenge faced by learning-based methods is generalization, including generalizing to larger problem sizes and maintaining performance when task structure change.

We use an attention-based as policy network, which naturally addresses the challenge of generalizing the policy to problem instances of varying sizes. We first train the model on a fixed map with randomly generated rewards and observe that the policy learns to make effective decisions under different reward distributions. However, the model exhibits poor generalization when the map layout or task structure change.To further improve generalization, I train the agent on datasets with both randomized maps and randomized rewards. I find that the agent is still able to learn high-quality decision-making strategies, and preliminary experiments indicate that the model possesses a certain degree of generalization capacity across different task sizes, map configurations, and parameter settings. Nonetheless, we have not yet conducted more comprehensive ablation studies to quantitatively evaluate the extent of this generalization capability.

In real-world UV scheduling scenarios, the problem usually involves multiple vehicles collaborating to complete tasks. Our current work considers only the single-vehicle case. Multi-vehicle coordination introduces additional challenges in both task allocation and path planning, which can be addressed using multi-agent reinforcement learning. This represents a promising direction for future research in our project.

MDP Formulation of OP

We make almost no modifications to the original Gym environment, except reducing the UV's maximum travel range from 4 to 3. Details can be found in the proposal and gym_report. A brief summary is provided below:

1. State

The state input to the policy network includes:

- The vehicle's current position,
- Remaining range
- Depots position
- Unvisited points (coordinates, reward values)
- Visited points (coordinates, reward = 0).

2. Action

The action is to select the next point to visit from the set of unvisited points. Invalid points, such as points that are unreachable or would prevent UV from returning to the depot, are masked out.

3. Transfer

Updating the vehicle's position and remaining range, marking the current point as visited, and accumulating rewards.

4. Reward

The reward equals the number of packages delivered at the visited point.

5. End

An episode terminates when the UV returns to the depot.

Attention-based Neural Network

To enable the model to generalize across problem instances of varying sizes, we adopt an attention-based neural network as our policy architecture. In the encoder, we use a Transformer-like structure that treats each task point as a token, producing the embedding of each point and a global graph embedding. Importantly, we do not incorporate positional encoding, as doing so would constrain the model to a fixed number of tokens and prevent it from accepting arbitrary-sized inputs. With this design, our policy network naturally supports tasks containing any number of points.

However, a flexible neural network architecture only provides potential for generalization. In practice, we find that the training strategy plays a much more decisive role in determining whether the model can fully exploit this potential. The way the model is trained ultimately governs its ability to generalize to different task sizes, map configurations, and parameter distributions.

In the decoder, we take the global embedding and the current state of the robot as the input. Two attention layers are applied to compute relevance scores over candidate actions. We mask invalid actions in the first attention layer, which we found significantly improves training performance. We then apply a second mask immediately before the softmax operation to ensure that the agent never selects an illegal action.

1. Encoder

The structure of the encoder is shown in Figure 3, and detailed network specifications can be found in the NN report. The encoder first takes as input the raw task features, including the coordinates and rewards of task points as well as the depot coordinates. These inputs pass through linear layer followed by Attention layers. This process produces a node embedding for each task point. A graph embedding is then obtained by averaging all node embeddings across the graph.

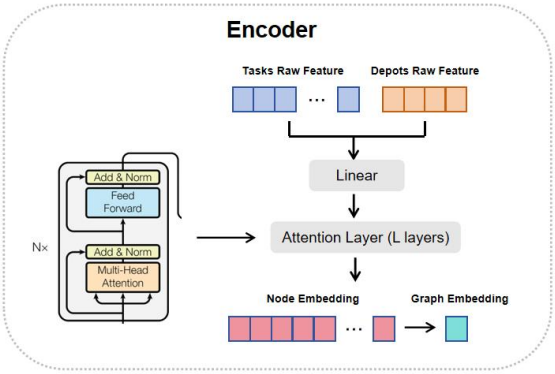

Fig. 3. Encoder

2. Decoder

The structure of the decoder is shown in Figure 4, and detailed architectural specifications are provided in the NN report. The decoder first concatenates the graph embedding generated by the encoder with the current state features, including the robot's current location's node embedding and its remaining travel distance as provided by the environment. This concatenated vector is passed through a linear layer. Subsequently, the output is processed by a Multi-Head Attention layer followed by a Single-Head Attention layer to compute the action logits. After applying a Tanh clipping operation, a softmax function is used to obtain the final distribution of actions probabilities.

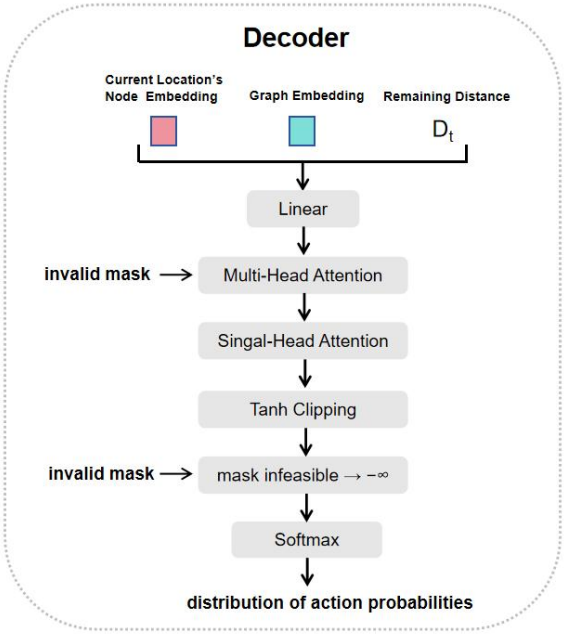

Fig. 4. Decoder

Training

We train our network using the REINFORCE algorithm, and detailed implementation can be found in the train_report.

1. Training Configuration

We run 128 parallel environments every episode, and treat each complete trajectory as a single training sample. To deal with the high-variance issue of REINFORCE, we experiment with two types of baselines: an exponential moving average baseline and a greedy baseline. Every 10 policy updates, one epoch, we evaluate the model on 100 test instances and save the model that achieves the best performance. The network is trained for a total of 2000 epochs.

2. Training Time

The training time per epoch increases as the model improves, because the agent learns to visit more task points, resulting in longer episodes. Our experiments are conducted

on an RTX 4090 GPU. At the beginning of training, each epoch takes approximately 6 seconds. In the later stages, each epoch requires around 20 seconds.Training all 2000 epochs takes roughly 9 hours.However, the model typically converges around 800 epochs, so 3 to 4 hours of training are generally sufficient.

Experiment Result

The training reward curve is shown in Figure 5, where we observe that the model converges at approximately 800 epochs. The greedy baseline achieves better overall performance than the exponential-moving-average baseline, although the latter provides more stable training behavior.

Figure 6 shows the actual performance of the trained policy, demonstrating that the agent has successfully learned meaningful path-planning behavior. Figures 7 and 8 show the loss curves corresponding to the greedy baseline and the exponential-average baseline, respectively.

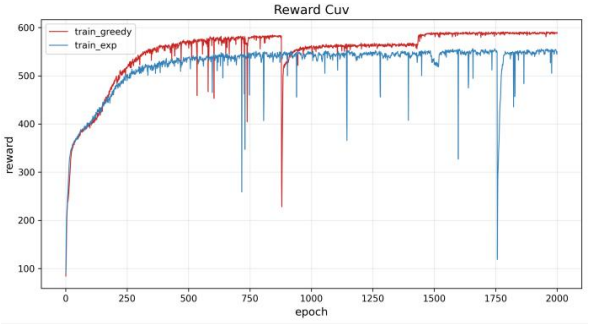

Fig. 5. Training Reward Curve

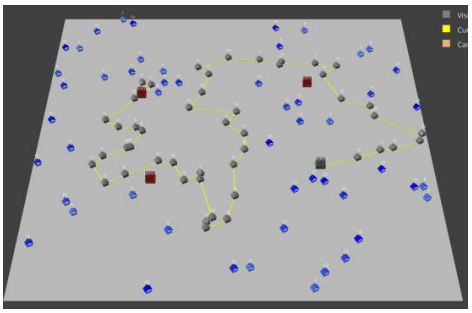

Fig. 6. Performance of the Trained Model

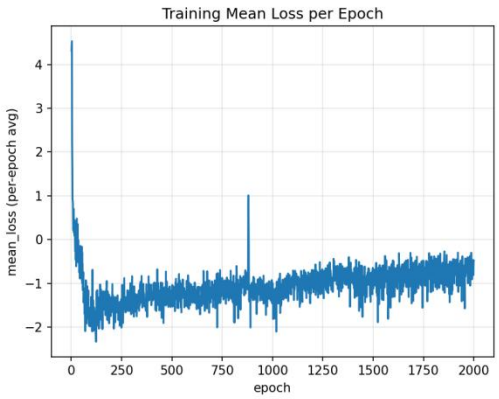

Fig. 7. Loss Curve of EMA Baseline

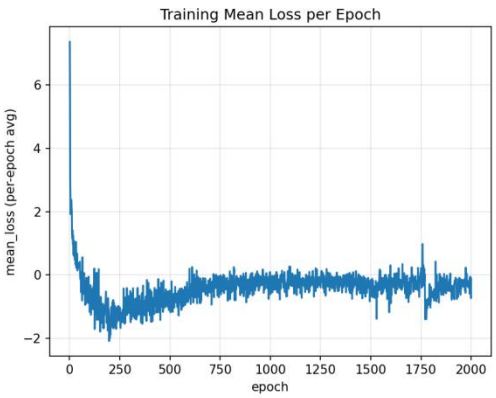

Fig. 8. Loss Curve of Greedy Baseline

1. Compare with OR-tools Benchmarks

Google OR-Tools is an open-source solver capable of addressing various combinatorial optimization problems. In our evaluation, we use OR-Tools as the benchmark for comparison. Due to the high computational complexity of the Orienteering Problem, the solver is unable to obtain an optimal solution even within 20 minutes. To make the comparison feasible, we limit the solver's runtime to 60

seconds per instance.

When applying OR-Tools, the UV's terminal depot must be fixed. Therefore, we run the solver separately for all four depot choices and take the best result. This results in a total computation time of 240 seconds per instance. We evaluate 100 test instances and report the averaged results.

The results show that our RL method achieves a reward only 12.2% lower than OR-Tools while requiring just 0.13 seconds, demonstrating a dramatic improvement in computational efficiency.

Table. 1. Performance Compared with OR-tools

	Reward	Gap	Run time
OR-tools	646.1	-	240 s
RL	567.5	12.2%	0.13 s

2. Discussion

Here I present several interesting observations encountered during model training.

- Batch Size
  A larger batch size leads to a better-performing model. Due to GPU memory limitations on desktop hardware, we initially used a batch size of 48, which is significantly smaller than the batch size of 128 used later on a rented server. We observed that the model trained with batch size 128 achieved approximately 8% higher rewards. This improvement may be attributed to the reduced gradient variance enabled by a larger batch size, resulting in more accurate stochastic gradient estimates.
- Limitation of Generalization Capability
  While RL offers extremely fast inference for operations research problems, its major limitation lies in generalization ability. Our current model learns to plan routes effectively under different reward distributions. However, when testing generalization across new map distributions, different task sizes, or different maximum UV travel ranges, I found that the model generalizes poorly.To address this issue and enhance generalization, I designed an extended set of experiments, which is presented in the next subsection.

Expansion and Discussion

1. Expansion

Building on the original setting with randomized reward distributions, I further train the model on randomly generated maps. I find that the neural network remains effective under this setting, and the reward curve is shown in Figure 9.

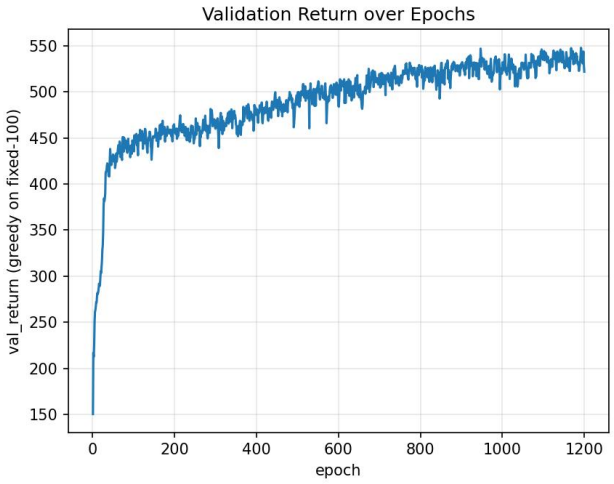

Fig. 9. Training Reward Curve on Randomly Generated Maps Setting

Due to time constraints, I have not yet completed comparisons with OR-Tools under this expanded setting. However, compared with the model trained only on randomized rewards, we observe that the agent trained on randomized maps exhibits substantially improved generalization ability. The agent appears to learn the relationship among the map layout, reward distribution, and its optimization objective.

- Random Maps
  Since the model is trained on randomly generated maps, it naturally acquires the ability to adapt to unseen map distributions. However, its solution quality is slightly worse than that of the previous model trained on a fixed map.
- Different Problem Sizes
  Although the agent is trained on data with a fixed problem size, it is still able to produce reasonably good solutions when tested on much larger instances, even up to ten times the original task size. Nonetheless, solution quality decreases as the problem size grows.
- Different Maximum Travel Ranges
  Even though training is conducted under a fixed maximum travel range, the agent seems to understand that its objective is to collect as much reward as possible within a limited budget. When tested with larger travel ranges, the agent fully utilizes the available distance, whereas the previously trained model only uses the distance scale seen during training.

Overall, the potential of the policy network can only be fully realized through effective training strategies.

2. Discussion

The primary advantage of reinforcement learning over traditional combinatorial optimization algorithms lies in its fast inference speed, which makes it suitable for decision-making problems requiring rapid or dynamic responses. However, generalization remains one of the most significant limitations of RL, and it is often the key factor determining whether an RL method is practically useful.

The foundation of generalization lies in the policy network architecture, but achieving strong generalization further requires appropriate training methodologies that allow the agent to learn as much as possible without collapsing into a “learn nothing” region.

Balancing solution quality and generalization capability remains one of the major challenges for applying reinforcement learning to decision-making and optimization problems. Developing training methods that yield policies with strong generalization ability while preserving high solution quality is crucial for applying reinforcement learning to operations research and combinatorial optimization problems. Meta-reinforcement learning (meta-RL) methods may be a promising direction for future research.

References

[1] Kool, Wouter, Herke Van Hoof, and Max Welling. "Attention, learn to solve routing problems!." arXiv preprint arXiv:1803.08475 (2018).