A multi-agent reinforcement learning framework for multiple vehicle routing problems with soft time windows

Ke Zhang[a], Meng Li[a*], Zhengchao Zhang[a] , Xi Lin[a], Fang He[b]

[a]*Department of Civil Engineering, Tsinghua University, Beijing 100084, P.R. China*

[b]*Department of Industrial Engineering, Tsinghua University, Beijing 100084, P.R. China*

Keywords: Reinforcement learning; Vehicle routing problem; Attention mechanism; Soft time window; Multi-Agent

1 Introduction

Multi-vehicle routing problem with soft time windows (MVRPSTW) is an indispensable constituent in urban logistics distribution system (Lau et al., 2003). In the last decade, numerous methods for MVRPSTW have sprung up, such as Iterated Local Search (Ibaraki et al., 2008), Genetic Algorithm (Louis et al., 1999; Wang et al., 2008), but most methods are based on heuristic rules which require huge computation time. With the rapid increasing of logistics demand, traditional methods incur the dilemma of computation efficiency. Deep Reinforcement Learning (DRL) shows great power in solving complex operation problems. Significant attention has also been attracted to model the VRP utilizing DRL framework.

Vinyals et al. (2015) propose the Pointer Network to solve the Travel Salesman problem (TSP) which generates a permutation of the input route adopting attention mechanism. Nazari et al. (2018) apply a policy gradient algorithm to solve VRP which consists of a Recurrent Neural Network decoder coupled with attention mechanism. Kool et al. (2018) propose an encoder-decoder framework with multi-head attention layers to solve VRP and show how to train this model using reinforce gradient estimator.

These pioneering researches have gained fruitful results in the field of single vehicle dispatching. However, few studies have attempted at employing DRL to solve

* Corresponding author. Email: mengli@tsinghua.edu.cn.

MVRPSTW because of existing methods only applicable for single agent. To solve this problem, we propose a novel reinforcement learning architecture named Multi-Agent Attention Model (MAAM).

2 Methodology

The MAAM model is essentially an attention-based encoder-decoder model which takes road network into account by a masking procedure. The problem definition and details of proposed model are described as follows.

2.1 Problem Definition

The road network can be regarded as a fully connected graph with randomly generated depot and customers in this Euclidean plane. Given vehicle capacity Q, number of vehicles M and a fully connected graph $G(\mathfrak{V})$ where $\mathfrak{V} = \{v_0, v_1, \dots, v_N\}$, v_0 is the depot with coordinate $\boldsymbol{x_0}$, $v_i (i \neq 0)$ is a customer with coordinate $\boldsymbol{x_i}$, demand d_i, time windows (e_i, l_i), early and late penalty coefficients $p_{0,i}$, $p_{1,i}$. The MVRPSTW is to find a set of vertex-disjoint routes $\boldsymbol{r}[m]$ $(m = 1,2,\dots,M)$ for each vehicle starting and ending at depot v_0. Under this circumstance, each customer v_i is served only once by one of the vehicles within its time windows. The problem is to find a solution $\boldsymbol{r}[1,M] = (\boldsymbol{r}[1], \boldsymbol{r}[2], \dots, \boldsymbol{r}[M])$ with minimal total cost, which is defined as:

$$Cost(\boldsymbol{r}[1,M]) = d_{sum}(\boldsymbol{r}[1,M]) + p_{sum}(\boldsymbol{r}[1,M]), \tag{1}$$

where $d_{sum}(\boldsymbol{r}[1,M]) = \sum_{m=1}^{M} \sum_{b=1}^{|\boldsymbol{r}[m]|-1} ||\boldsymbol{x}_{r[m][b]}, \boldsymbol{x}_{r[m][b+1]}||_2$ is the total traveling cost of all vehicles, $p_{sum}(\boldsymbol{r}[1,M]) = \sum_{m=1}^{M} \sum_{i=1}^{N} \left[I_{(e_i > \tilde{t}_i)} * p_{0,i} * (e_i - \tilde{t}_i) + I_{(\tilde{t}_i > l_i)} * p_{1,i} * (\tilde{t}_i - l_i) \right]$ denotes the total penalty for time window constraints, $\tilde{t}_i$ represents the total travel time when a vehicle arriving at customer i.

2.2 Encoder Framework

In MAAM model, the encoder firstly computes initial customer embeddings h_i through a learned linear projection with parameters W^1 and b^1:

$$h_i = W^1 \left[\boldsymbol{x_i}, d_i, \hat{t}_{i,0}, \hat{t}_{i,1} \right] + b^1. \tag{2}$$

Then the embeddings are updated using multiple attention layers. Each attention layer carries out a multi-head attention and a feed-forward operation. The attention

mechanism can be interpreted as a weighted message passing algorithm between customers in a graph. We compute the attention value Z times with different parameters and denote the results by h'_{iz} for $z \in \{1,2,\dots,Z\}$.The final multi-head attention value for customer i is a function of $h_1,\dots,h_n$:

$$\mathcal{F}_i(h_1,\dots,h_n) = \sum_{z=1}^{Z} W_Z^O h'_{iz}. \tag{3}$$

The remainder of attention layer is a feed-forward operation F with skip-connection:

$$\hat{h}_i = h_i \ + \mathcal{F}_i(h_1,\dots,h_n), \tag{4}$$

$$h_i^{(1)} = \hat{h}_i + F(\hat{h}_i), \tag{5}$$

where the operation F is defined as:

$$F(\hat{h}_i) = W_1^f \,\mathrm{ReLu}\left(W_2^f \mathcal{F}_i(h_1,\dots,h_n) + b_0^f\right) + b_1^f. \tag{6}$$

We compute equations (9-10) λ times and acquire $\{h_i^{(\lambda)}, i = 1,\dots,n\}$. Finally, the encoder computes an aggregated embedding of the input customers as the mean of the final output layer:

$$\bar{h}^{(N)} = \frac{1}{n}\sum_{i=1}^{N} h_i^{(\lambda)}. \tag{7}$$

2.3 Decoder Framework

In decoder part, the agents perceive the state of the environment and each other. Then they decide a sequential action set based on the knowledge obtained through this perception.

(1) State

The global state can be divided into environment state and agent state. Environment state contains the final customers embedding $\bar{h}^{(N)}$ and customers which have been already visited. The agent state consists of the current vehicle location and remaining capacity.

At each decoding timestep, the vehicle chooses the customers to visit in the next step. After visiting customer i, the remaining capacity $\hat{d}_{m,t}$ of vehicle m is updated.

In order to utilize information of state, we define multiple vehicles context embedding $h_{(c),t}^{(N)}$ for the decoder at timestep t which comes from the encoder and the vehicle output up to timestep t:

$$h_{(c),t}^{(N)} = \left[\bar{h}^{(N)}; h_{r_{t-1},1}^{(N)}; \hat{d}_{1,t}; h_{r_{t-1},2}^{(N)}; \hat{d}_{2,t}; \ldots; h_{r_{t-1},M}^{(N)}; \hat{d}_{M,t}\right]. \quad (8)$$

(2) Action

Action for each vehicle represents the choice of next customer to be visited at timestep t. Firstly, we compute a new multiple vehicles context embedding $h_{(c),t}^{(N)\prime}$ using the multi-head attention mechanism:

$$h_{(c),t}^{(N)\prime} = \mathcal{F}\left(h_{(c),t}^{(N)}\right). \quad (9)$$

Then compute the compatibility of the query $q_{(c)}$ with all customers:

$$q_{(c)} = W^Q h_{(c),t}^{(N)\prime}, \quad (10)$$

$$k_i = W^K h_i^{(N)}, \quad (11)$$

$$u_{i,m,t} = \tanh\left(\frac{q_{(c)}^T k_i}{\sqrt{d_k}}\right). \quad (12)$$

We mask (set $u_{i,m,t} = -\infty$) customers which has been visited before timestep t, and the customers whose demand exceed vehicle remaining capacity.

Finally, we compute the probability of choosing customer i at timestep t for vehicle m through the softmax function:

$$p_{i,m,t} = \text{softmax}(u_{i,m,t}) = \frac{e^{u_{i,m,t}}}{\sum_j e^{u_{j,m,t}}}. \quad (13)$$

(3) Reward

The reward is defined as the total tour cost:

$$R(\boldsymbol{r}[\boldsymbol{1},\boldsymbol{M}]) = Cost(\boldsymbol{r}[\boldsymbol{1},\boldsymbol{M}]) \quad (14)$$

2.4 Training Method

We optimize the parameter by the reinforce gradient estimator:

$$\nabla_\theta L(\theta|s) = E_{r\sim p_\theta(\cdot|s)}[(R(\boldsymbol{r}[\boldsymbol{1},\boldsymbol{M}]) - R(\boldsymbol{r^{BL}}[\boldsymbol{1},\boldsymbol{M}]))\nabla_\theta log p_\theta(\boldsymbol{r}[\boldsymbol{1},\boldsymbol{M}]|s)]. \quad (15)$$

$R(\boldsymbol{r}[\boldsymbol{1},\boldsymbol{M}])$ is the cost of a solution from a deterministic sample decoding of the

model while $R(\boldsymbol{r^{BL}}[\mathbf{1}, \boldsymbol{M}])$ is from deterministic greedy decoding. We use the Adam optimizer to train parameter by minimizing $\nabla_\theta L(\theta|s)$.

3 Results

3.1 Dataset Description and Parameter Setting

The customers locations, demands and time windows are randomly generated from uniform distribution. Specifically, the depot location and customers are randomly generated in the square $[0,10] \times [0,10]$. Vehicle capacity is set as 300. Time window is randomly generated from $[0,60]$. $p_{0,i}$ and $p_{1,i}$ are randomly generated from $[0,0.2]$ and $[0,1]$ separately. Each customer demand is randomly generated from $[0,10]$ (two vehicles), $[0,15]$ (three vehicles), $[0,20]$ (four vehicles) and $[0,25]$ (five vehicles). We evaluate our model on 1000 instances. The proposed MAAM is compared with Genetic Algorithm (GA) and Iterated Local search algorithm (ILS).

3.2 Result Comparison

Table 1 shows the total costs of each model under each testing scenario. Our proposed model achieves the best performance compared with other baselines both in solution quality and computation efficiency. Unlike most classical heuristic methods, our model is robust to problem changes, e.g., when a customer changes its demand value or relocates to a different position, it can automatically adapt the solution.

Table 1. Performance comparison for MVRPSTW

	Vehicle	Cost	Time	Vehicle	Cost	Time
GA		278.7	6.2(h)		267.5	5.4(h)
ILS	2	181.6	1.1(h)	3	161.6	56(m)
MAAM		**131.5**	**5(s)**		**132.3**	**5(s)**
GA		263.5	4.5(h)		281.4	4.1(h)
ILS	4	177.8	35(m)	5	187.9	34(m)
MAAM		**139.4**	**4(s)**		**146.5**	**4(s)**

4 Conclusion

In this paper, we propose a novel DRL model for solving MVRPSTW problem. Validated by synthetic experiments, MAAM consistently outperforms traditional

methods with limited computation time. The result further proves that DRL is promising for solving MVRPSTW problem in real urban logistics applications. In the future research, it will be an important topic to utilize reinforcement learning to solve online real-time vehicle routing problem of practical importance.

References

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A., 2017. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine. 34(6), 26-38.

Kool, W., van Hoof, H., & Welling, M., 2018. Attention solves your TSP, approximately. Avaiable from: arXiv: 1803.08475.

Lau, H. C., Sim, M., & Teo, K. M., 2003. Vehicle routing problem with time windows and a limited number of vehicles. European journal of operational research. 148(3), 559-569.

Ibaraki, T., Imahori, S., Nonobe, K., Sobue, K., Uno, T., & Yagiura, M., 2008. An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. Discrete Applied Mathematics. 156(11), 2050-2069.

Louis S J, Yin X, Yuan Z Y., 1999. Multiple vehicle routing with time windows using genetic algorithms. In: Proceedings of the 1999 Congress on Evolutionary Computation- CEC99. IEEE, 3, 1804-1808.

MohammadReza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takac.,2018. Reinforcement learning for solving the vehicle routing problem. In Advances in Neural Information Processing Systems, pp. 9860–9870.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly., 2015. Pointer networks. In: Advances in Neural Information Processing Systems, pp. 2692–2700.

Wang X, Xu C, Hu X., 2008. Genetic algorithm for vehicle routing problem with time windows and a limited number of vehicles. In: 2008 International Conference on Management Science and Engineering 15th Annual Conference Proceedings, IEEE, pp. 128-133.