Reinforcement Learning for Vehicle Routing Problem

Rotem Bar, Uria Levkovitz

ABSTRACT Combinatorial optimization problems(COP) are problems that involve finding the "best" solution from a finite but potentially large set of candidate solutions. Traveling salesman problem (TSP) and vehicle routing problem (VRP) are some of the real-world examples of such problems. As it turns out, these classes of problems form the core of the supply chain industry and can mean the difference between market leadership and obscurity for some of the world's fastest-growing brands. This study will examine the use of reinforcement learning methods for the VRP as presented in several recent studies. Additionally, we will conduct an experiment that will examine the results of using gated recurrent units (GRU) instead of recurrent neural networks (RNN) in the model architecture.

I. INTRODUCTION

VRP is an essential NP-hard Combinatorial Optimization Problem [1] with extensive industrial applications in various domains. Exact methods have the exponential worst-case computational complexity, rendering them impractical for solving real large-scale problems, even for highly optimized solvers such as Concorde. In contrast, although lacking optimality guarantees and non-trivial theoretical analysis, heuristic solvers search [2] for near-optimal solutions with much lower complexity. They are usually desirable for real-life applications where statistically better performance is the goal. Traditional heuristic methods [3][4][5] are manually designed based on expert knowledge, which is generally human-interpretable. However, supported by the recent development of deep learning technology, modern methods [6][7][8] train robust deep neural networks to learn the complex patterns from the VRP instances generated from specific distributions. These works constantly improve the performances of deep learning models for solving VRP. Unfortunately, they are far worse than the strong traditional heuristic solver and are generally limited to small problem sizes.

II. VRP

The VRP [9] concerns the service of a delivery company, how things are delivered from one or more depots with a given set of home vehicles and operated by a set of drivers who can move on a given road network to a set of customers. It asks for a determination of a set of routes, S (one route for each vehicle that must start and finish at its depot), such that all customers' requirements and operational constraints are satisfied, and the global transportation cost is minimized. This cost may be monetary, distance, or otherwise. The road network can be described using a graph where the arcs are roads and vertices are junctions between them. The arcs may be directed or undirected due to the possible presence of one-way streets or different costs in each direction. Each arc has an associated cost, which is generally its length or travel time, which may depend on vehicle type. To know the global cost of each route, the travel cost, and the travel time between each customer and the depot must be known. To do this, our original graph is transformed into one where the vertices are the customers and depot, and the arcs are the roads between them. The cost of each arc is the lowest cost between the two points on the original road network. This is easy to do as shortest path problems are relatively easy to solve. Sometimes it is impossible to satisfy all of a customer's demands, and in such cases, solvers may reduce some customers' demands or leave some customers unserved. To deal with these situations, a priority variable for each customer can be introduced or associated penalties for the partial or lack of service for each customer given.

The objective function of a VRP can be very different depending on the particular application of the result. Still, a few of the more common objectives are:

- Minimize the global transportation cost based on the global distance traveled as well as the fixed costs associated with the used vehicles and drivers

- Minimize the number of vehicles needed to serve all customers
- Least variation in travel time and vehicle load
- Minimize penalties for low-quality service
- Maximize a collected profit/score.

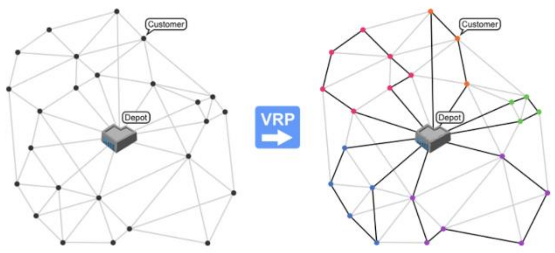
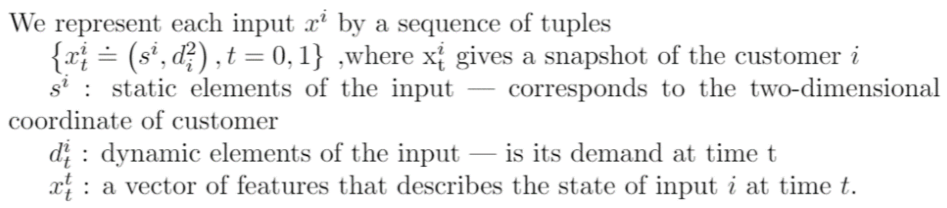


FIGURE 1. VRP example

III. REINFORCEMENT LEARNING FOR VRP

This section will describe how we will use reinforcement learning to solve the VRP[7]. To do so, we will first define the following definitions needed for the reinforcement learning process.



We represent each input x^i by a sequence of tuples

$\{x_t^i \doteq (s^i, d_i^2), t = 0, 1\}$,where x_t^i gives a snapshot of the customer i

s^i : static elements of the input — corresponds to the two-dimensional coordinate of customer

d_t^i : dynamic elements of the input — is its demand at time t

x_t^t : a vector of features that describes the state of input i at time t.

FIGURE 2. VRP problem definition

The method also uses Y sub t to denote the decoded sequence up to time t, i.e., Y sub t = {y sub 0, · · ·, y sub t}. It tries to find a stochastic policy π that generates the sequence Y in a way that minimizes a loss objective while satisfying the problem constraints. The optimal policy π∗ will generate the ideal solution with probability 1. The model aims to make π as close to π∗ as possible. It uses the probability chain rule to decompose the probability of generating sequence Y, i.e., P(Y |X sub 0), as follows:

$$P(Y|X_0) = \prod_{t=0}^{T} P(y_{t+1}|Y_t, X_t), \quad (1)$$

and

$$X_{t+1} = f(y_{t+1}, X_t) \quad (2)$$

Is a recursive update of the problem representation with the state transition function f. The attention mechanism calculates each component on the right-hand side of (1).

$$P(y_{t+1}|Y_t, X_t) = \text{softmax}(g(h_t, X_t)), \quad (3)$$

Where: g: an affine function that outputs an input-sized vector h sub t: the state of the RNN decoder that summarizes the information of previously decoded steps y sub 0, · · ·, y sub t.

The proposed attention mechanism employed in this model is illustrated in the figure below.

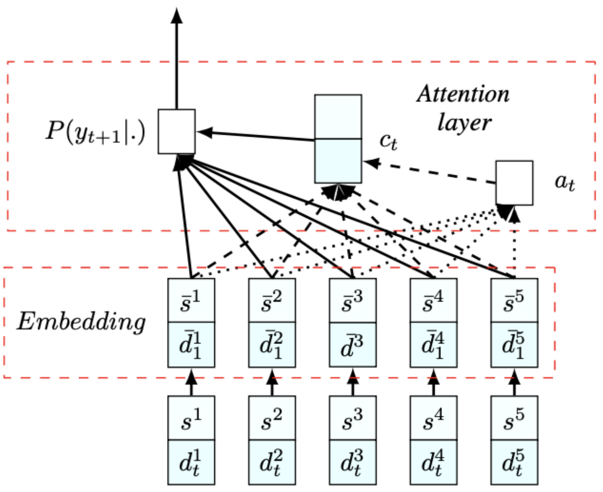


FIGURE 3. Model architecture

Figure 3: The embedding layer maps the inputs to a high-dimensional vector space. On the right, an RNN decoder stores the information of the decoded sequence. Then, the RNN hidden state and embedded input produce a probability distribution over the next input using the attention mechanism. As seen in figure 3. This model is composed of two main components. The first is a set of embeddings that maps the inputs into a D-dimensional vector space. Multiple embeddings may exist corresponding to different elements of the input, but they are shared among the inputs. The second component of this model is a decoder that points to input at every decoding step. The framework uses RNN to model the decoder network. Notice that both the static and dynamic elements are fed as the inputs to the decoder network.
An attention mechanism is based on directing the model to pay greater attention to certain factors when processing the data. Figure 3 illustrates the attention mechanism employed in this framework. At decoder step (i), a context-based attention mechanism is used to extract the relevant information from the inputs using a variable-length alignment vector a sub t. In other words, a sub t specifies how much every input data point might be relevant in the next decoding step t.
The alignment vector a sub t is then computed as:

$$a_t = a_t(\bar{x}^i_t, h_t) = \text{softmax}\left(u_t\right), \quad \text{where } u^i_t = v^T_a \tanh\left(W_a[\bar{x}^i_t; h_t]\right). \tag{4}$$

$$c_t = \sum_{i=1}^{M} a^i_t \bar{x}^i_t, \tag{5}$$

$$P(y_{t+1}|Y_t, X_t) = \text{softmax}(\tilde{u}^i_t), \quad \text{where } \tilde{u}^i_t = v^T_c \tanh\left(W_c[\bar{x}^i_t; c_t]\right). \tag{6}$$

The embeddings and the attention mechanism are invariant to the input order. Hence, the order in which the inputs are fed into the network does not matter. The proposed model applies to tasks with no obvious input sequence, such as sorting.

IV.RNN and GRU

In our study, we will examine the use of the architecture described in the previous section using the GRU model instead of the RNN model. In the following section, we will briefly describe the two models.

A. RNN

RNN [10] connects the outputs of all neurons to the inputs of all neurons. This is the most general neural network topology because all other topologies can be represented by setting some connection weights to zero to simulate the lack of connections between those neurons. The illustration to the right may be misleading to many because practical neural network topologies are frequently organized in "layers," and the drawing gives that appearance. However, what appear to be layers are, in fact, different steps in time of the same fully recurrent neural network. The left-most item in the illustration shows the recurrent connections as the arc labeled 'v.' It is "unfolded" in time to produce the appearance of layers. Sine in the study we covered[7], the Another decided not to disconnect the RNN modules and not pass the hidden states between them. The outcome of these RNN modules is shown in figure 4. The results are modules of the received input. A neural network with tan-hyperbolic activation function transforms the inputs into high dimensional vectors.

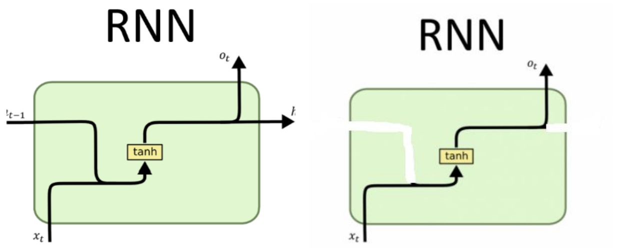


FIGURE 4. left: RNN module; Right: RNN module without passing hidden state

B. GRU

The critical distinction between vanilla RNNs and GRUs [11] is that the latter supports the gating of the hidden state. This means that we have dedicated mechanisms for when a hidden state should be updated and also when it should be reset. These mechanisms are learned, and they address the concerns listed above. For instance, if the first token is of great importance, we will learn not to update the hidden state after the first observation. Similar to the previously described RNN section, in our study, we will not pass the hidden state between the GRU modules. The outcomes are presented in figure 5, where the input passes through two neural networks, one with a tan-hyperbolic activation function and the second with a sigmoid activation function.

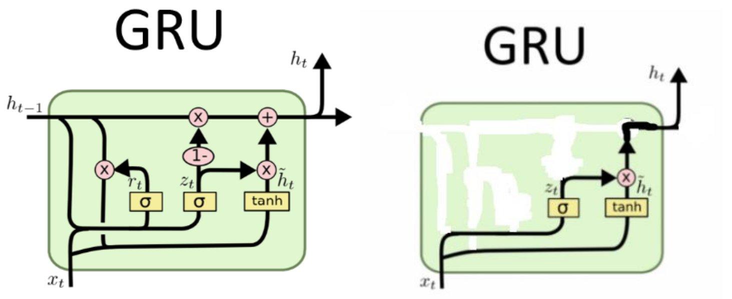


FIGURE 5. left: GRU module; Right: GRU module without passing hidden state

V. EXPERIMENTAL RESULTS

To evaluate the model performance, we will conduct the model architecture described in figure 3, presented in [7]. Our study implemented the algorithm with one change, replacing the RNN module with a GRU muddle. The experiment can be divided into two phases. The first phase is training the model with 3500 iterations of the forward pass and backpropagation, the same as [7]. We will conduct the trained model over 1000 data samples in the next testing phase. The evaluation metric is the mean length of all the predicted paths. Due to running time and computer processing limitations, we will perform the experiments on the two following seniors:

- 10 cities + load 20
- 20 cities + load 30

Addionlty we will compare the experimental results with the results received at [7], which were performed with RNN modules.

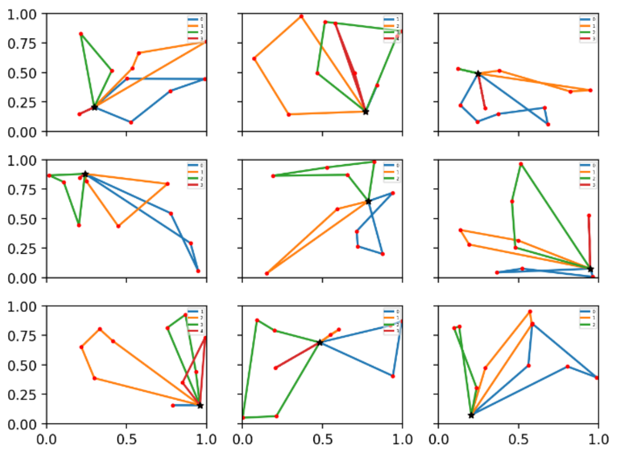


FIGURE 4. VRP with 10 cities + load 20

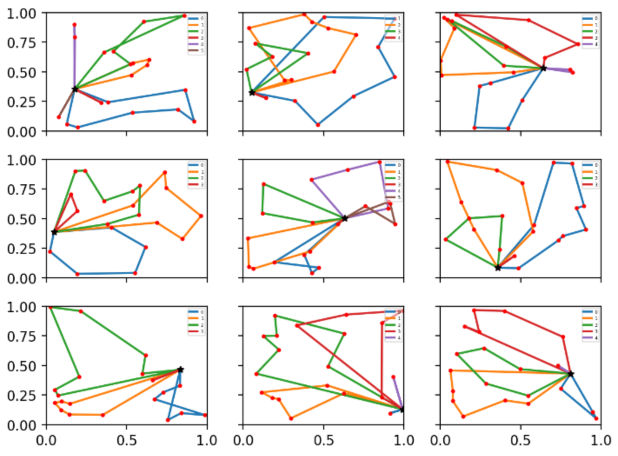


FIGURE 5. VRP with 20 cities + load 30

	RNN	**GRU**
VRP with 10 cities + load 20	3.97	4.01
VRP with 20 cities + load 30	6.59	6.91

TABLE 1. Mean length results (less is better)

Figures 4 and 5 present some of the model predictions of the two scenarios, demonstrating how the model performs on the data samples. We can see that the results consist of both systems for the 10 and 20 cities. To evaluate by a metric, we will need to examine the results in table 1. To evaluate the performance, we will look at table 1, which shows better outcomes for the model with the RNN modules, as presented in 7.

VI. CONCLUSIONS

This study examined the VRP, a well-known task studied for many years. For the VRP, several algorithms address this task. In our work, we examined the use of reinforcement learning for the VRP. For that purpose, we implemented the algorithm that [7] suggested. We conduct an experiment to evaluate the model performance by examining different cities of 10 and 20 cities. The experiment shows the RNN capability over the GRU modules in the model architecture with better results in finding the best solutions. The purpose of this study was to demonstrate one of the solutions of the VRP conducted with reinforcement learning and in addition to that, to explore a different direction by changing one main component in the model architecture.

REFERENCES

[1] Toth, P., 2000. Optimization engineering techniques for the exact solution of NP-hard combinatorial optimization problems. European journal of operational research, 125(2), pp.222-238.

[2] Groër, C., Golden, B. and Wasil, E., 2011. A parallel algorithm for the vehicle routing problem. INFORMS Journal on Computing, 23(2), pp.315-330.

[3] Taillard, É.D., 1999. A heuristic column generation method for the heterogeneous fleet VRP. RAIRO-Operations Research, 33(1), pp.1-14.

[4] Laporte, G. and Semet, F., 2002. Classical heuristics for the capacitated VRP. In The vehicle routing problem (pp. 109-128). Society for Industrial and Applied Mathematics.

[5] Prins, C., 2002. Efficient heuristics for the heterogeneous fleet multitrip VRP with application to a large-scale real case. Journal of Mathematical Modelling and Algorithms, 1(2), pp.135-150.

[6] Li, J., Xin, L., Cao, Z., Lim, A., Song, W. and Zhang, J., 2021. Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, 23(3), pp.2306-2315.

[7] Nazari, M., Oroojlooy, A., Snyder, L. and Takác, M., 2018. Reinforcement learning for solving the vehicle routing problem. Advances in neural information processing systems, 31.

[8] Kool, W., van Hoof, H., Gromicho, J. and Welling, M., 2022. Deep policy dynamic programming for vehicle routing problems. In International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (pp. 190-213). Springer, Cham.

[9] Braekers, K., Ramaekers, K. and Van Nieuwenhuyse, I., 2016. The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering, 99, pp.300-313.

[10] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[11] Dey, R. and Salem, F.M., 2017, August. Gate-variants of gated recurrent unit (GRU) neural networks. In 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS) (pp. 1597-1600). IEEE.