# Hierarchical Reinforcement Learning for Vehicle Routing Problems with Time Windows

Yunli Wang[†,*], Sun Sun[†], Wei Li[◇]
[†] National Research Council Canada
[†] National Research Council Canada
[◇] University of Ottawa

**Abstract**

Vehicle routing problem with time windows (VRPTW) is a practical and complex vehicle routing problem (VRP) which is faced by thousands of companies in logistics and transportation. Usually, VRP is solved by traditional heuristic algorithms. Recently, deep learning models under the reinforcement learning (RL) framework have been proposed to solve variants of VRP. In our study, we propose to use the hierarchical RL to find an optimal policy for generating optimal routes in VRPTW. The hierarchical RL structure includes a low level which generates feasible solutions and a high level which further searches for an optimal solution. Experimental results on two distinct datasets show that the proposed hierarchical RL model outperforms the non-hierarchical RL model and the heuristic algorithms Google OR-Tools. The proposed model also shows generalization capability in different scenarios: varied time window constraints, from small-scale to large-scale problems, and across different datasets. The flexible framework of hierarchical RL can also be applied to solve other complex VRPs with multiple objectives.

**Keywords:** Vehicle routing problem with time windows; hierarchical reinforcement learning; generalization

## 1. Introduction

Standard vehicle routing problem (VRP) aims to find an optimal routing from the depot to multiple customers so as to minimize the cost, and can be formulated as a combinatorial optimization problem. Practically, VRP has found many important applications in logistics and transportation. Depending on different applications there are several variants of VRP such as capacitated vehicle routing problem (CVRP), split delivery VRP (SDVRP), VRP with time windows (VRPTW), and VRP pick and delivery (VRPPD). VRP generalizes Traveling Salesman Problem (TSP) and is NP-hard. To solve VRP traditional methods can be categorized into exact solutions, classical heuristic algorithms, and meta-heuristic algorithms [1, 2]. Exact solutions only exist in small-scale problems. Heuristic and meta-heuristic algorithms are usually designed for one particular variant of VRP, and thus cannot be easily adjusted for other VRP problems.

Recently deep learning approaches have been proposed to solve large-scale combinatorial optimization problems with less inference time [3–5]. The initial effort based on an attention equipped encoder-decoder model used supervised learning to solve TSP [6]. Reinforcement learning (RL) was introduced to replace the supervised learning for solving TSP [7]. Other deep learning models with different encoder architectures were also proposed to solve CVRP, stochastic VRP [3], or variants of TSP [5]. Unlike heuristic algorithms, these deep learning based models are able to deal with multiple variants of VRP [3–5], and can be generalized from small-scale to large-scale problems [4]. On the other hand, although deep learning based models have been used to solve TSP, CVRP, and TSP with time windows (TSPTW), etc, very few studies have focused on more practical and complex VRPs such as VRPTW

VRPTW has found many applications in distribution management such as beverage and food delivery, bank and postal delivery, and waste collection [1]. In this paper, we focus on the study of VRPTW based on deep learning approaches. Exact algorithms and heuristics for VRPTW have been reported in operation research community [1]. Recently a deep learning approach has been proposed to solve VRPTW by using the attention on sequence

[*] Yunli.Wang@nrc-cnrc.gc.ca

construction of multiple routes [8]. Differently, in our study we instead use a hierarchical RL model to solve VRPTW. A hierarchical RL algorithm could, in principle, efficiently find solutions to complex problems and reuse representations between related tasks [9]. In classical RL tasks, hierarchical RL can significantly reduce the action space and the number of the training iterations, and can also improve generalization [10].

We also empirically investigate the generalization of the hierarchical RL model for VRPTW, which is ignored by most previous work. Specifically, most deep learning models for VRP are trained and tested with data from the same distribution. However, in practice, it is highly likely that the test samples are from a different distribution (e.g., when the number of customers changes slightly). Additionally, since complex VRPs are extensions of VRP with multiple constraints the test distribution inevitably changes if these constraints change (e.g., by relaxing or restricting the constraints).

Experimentally, we compare the hierarchical RL model with non-hierarchical RL models and traditional heuristic algorithms for VRPTW. Regarding generalization, we test our algorithm in three scenarios: generalization to a relaxed/restricted time window, generalization to large-scale problems, and generalization across different datasets. Experimental results generally show the superior performance of our proposed algorithm.

## 2. **Related work**

Traditional methods for solving VRP include exact algorithms, heuristic, and meta-heuristic algorithms, and are usually designed for specific problems. Exact algorithms such as branch-and-bound and branch-and-cut are usually slow and can only be applied to small instances. Examples of classical heuristic algorithms are constructive methods, sequential insertion heuristics, and two-phase methods for generating solutions. These heuristic algorithms are able to produce solutions with high quality within modest computation time, and can be extended to problems with different constraints. Meta-heuristic algorithms such as Tabu search and evolutionary algorithms have reached promising results on relatively large-scale VRP [1], and some have been applied to solve a combination of multiple VRPs. For instance, with historical data a three layer search algorithm 3L-SDVRP was proposed to solve a combinatorial optimization problem of VRP, splitting delivery routing (SDR), and 3D bin packing (3DBP) [11], and a heuristic algorithm was proposed for 3D constraints SDVRP [12].

Recently deep learning models have been proposed to solve combinatorial optimization problems [3–5, 7]. With TSP represented as a sequence of nodes a pointer network was proposed to employ the attention on recurrent neural networks (RNNs) to learn the probability distribution of nodes, and finally to output the permutation over the input [6]. Later, reinforcement learning (RL) was introduced to solve TSP [7]. Along this line of work, the following deep learning models use very similar RL architecture but different neural network structures for a variety of VRPs. For example, an encoder-decoder architecture used for CVRP was proposed based on pointer networks, and an actor-critic policy gradient method was used for training the model [3]. The combination of a graph embedding method (structure2vec) and Q-learning was used to solve TSP [4]. A transformer-based encoder and REINFORCE algorithm were used to solve VRP and its variants CVRP, SDVRP, OP, and PCTSP [5]. Additionally, some deep learning models were suggested to improve existing methods [13, 14]. To solve VRPTW, multiple routes were suggested to be generated in parallel based on multiple encoders including node encoder, tour encoder and vehicle encoder [8].

Although several deep learning approaches have been proposed to solve VRPs, only a few studies focus on complex VRPs. One big challenge for solving complex VRPs is to find feasible solutions. For simple VRPs such as CVRP and SDVRP, *masking scheme* is used to assign infinity to infeasible solutions by masking nodes that cannot be visited [3]. However,

complex VRPs such as VRPTW may include multiple constraints and thus *masking scheme* may not be sufficient to guarantee feasibility. To solve VRPTW, in our study we use hierarchical RL by first generating feasible solutions at the low level and then searching an optimal solution at the high level.

## 3. Methods

VRPTW can be formulated as a complete undirected graph $G = (V, E, A)$. The set $V = \{v_0, ..., v_n\}$ is a vertex set. The vertex $v_0$ represents the depot, and the vertex $v_i, i \neq 0$, represents a customer with the demand $q_i$. Each edge $e \in E = (i, j), i, j \in V$, is associated with the travel cost $c_{ij}$. All vertexes have the node attribute $A = [X, T, D]$, where $X$ represents the location, $T$ represents the time window, and $D$ represents the demand of customers (except the depot). In particular, we have $X_i \subset R^2$ denoting the coordinates of the location and $T_i \subset R^2$ including the time window requested by the customer. We assume that there are $M$ vehicles each with the capability $Q$ available at the depot for delivery. The goal of VRPTW is to minimize the total cost such that the demand of each customer should be met within the time window. Customers can be visited only once for their demand. Also, we assume a constant speed of vehicle, so the travel time is proportional to the travel distance and the travel cost.

### 3.1. Graph pointer network

Graph pointer network (GPN) has been proposed to solve TSP with time windows (TSPTW) [15]. In our study, GPN is adopted to create routes for VRPTW. Compared to TSPTW, VRPTW additionally includes the vehicle capacity constraint. The input to GPN is a graph with the unordered vertex set $V$, and the output includes multiple routes. Each route consists of a sequence of ordered $v_k$ indicating the visiting order. As shown in Figure 1, GPN uses the encoder-decoder architecture. The encoder has two components: the context vector which represents the global representation of VRPTW, and the pre-context vector which captures the local context of the current route. The context vector is generated from a graph convolutional network (GCN) [16]. Similar to other sequence-to-sequence models, the sequence of previous nodes $(v_1, v_2, ..., v_{k-1})$ in the current route is used to generate the pre-context embedding. In the decoder, the context vector and the pre-context vector are combined using the attention mechanism to predict the probability distribution of the next customer $v_k$ on the route.

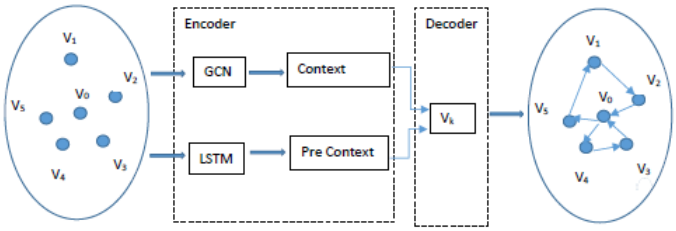


*Figure 1.* The encoder-decoder architecture in graph pointer network. The input to the encoder is the vertex set $V$ with nodes $v_0, v_1, v_2, ...$ and the output of the decoder is one solution of two routes with ordered sequence of nodes $v_0, v_4, v_3, v_0$ and $v_0, v_5, v_1, v_2, v_0$.

The context vector is generated using the graph embedding method GCN. Given a network, a graph embedding method is used to optimize the mapping of nodes to low-dimensional vectors where geometric relationships in the learned embedding space reflect

the structure of the original groups. The input of a graph embedding algorithm is an undirected graph $G = (V, E, A)$. The graph embedding approach maps each node $v_i$ to a low dimensional vector $Z_{v_i} \in R^d$. Most of these methods use an encoder-decoder network, in which the encoder maps nodes to vector embeddings, and the decoder decodes a graph proximity measure from the vector embeddings.

To learn the node embeddings, GCN uses convolutional neural networks to aggregate information from neighbors. In the training process, the initial layer (Layer 0) of the neural network is assigned as the node features $A$. The node embeddings $Z_{v_i}$ are aggregated from the direct neighbors in Layer 1 and the second order neighbors in Layer 0 using aggregation function $G$. GCN concatenates the node embedding of a node $v \in V$ with neighbor embeddings using $G$. The combined vector from the node embeddings of all nodes learned in GCN serves as the context vector in GPN.

The attention mechanism used at the decoder combines the context vector and the pre-context vector to generate the next node in the route. In particular, at the decoder, the node with the highest probability is selected as the next node in the route (during training we instead sample the next node from the distribution generated by the encoder). One pass of GPN creates one solution for VRPTW. The optimization of these solutions is conducted by hierarchical RL as explained below.

### 3.2. **Hierarchical reinforcement learning (RL)**

VRPTW can be formulated as an optimization problem with multiple constraints. Based on hierarchical RL, we transform VRPTW into multi-objective optimization problems with two layers. To address the time windows, we focus on the soft penalty and penalize the time window violations in the objective function. Most traditional heuristics for VRPTW use a hierarchical objective which minimizes the number of vehicles and then the travel cost [1]. In our study, minimizing the number of vehicles is used as an auxiliary task in the objective function for VRPTW. We decompose the optimization for VRPTW into two problems by using a two-level hierarchical RL architecture. The low level and the high level are both built by GPN but with different training objectives. Below, we first formulate VRPTW as a RL problem, then introduce the training objectives in the low and the high level models.

3.2.1. *RL for VRPTW.* Denote one solution for VRPTW as $\sigma$ with $\sigma_k = v_i, k \in 1, 2, ..., N$. This consists of multiple routes and each route starts from the depot and ends at the depot using one vehicle. With GPN many solutions can be generated for VRPTW, and RL is used to search for an optimal solution.

We first formulate VRPTW as a Markov Decision Process (MDP). Under MDP an agent aims to minimize the cost by interacting with the environment. For VRPTW, the environment $s_t$ includes the vehicle capacity, the locations of all nodes, as well as the demands and the time windows of all customers. Each state is defined as the set including all previously visited customers $s_t = \{\sigma_k\}_{k=1}^{t}$. The action $a_t$ is to add a specific customer in the route (i.e., $a_t = \sigma_{k+1}$) or to return to the depot (i.e., $a_t = \sigma_1$). The action $a_t$ is thus discrete and the action space will increase with the customer size. If the demand of a customer is satisfied, the customer will be masked so it will not be selected in the future. Depending on the objective, the cost $c(s_t, a_t)$ can include the travel cost $c_{k,k+1}$, the time penalty $t_{k,k+1}$, and the number of the vehicles. Given the existing customers, the agent aims to find a policy $\pi$ which is a distribution over the next visiting customer. The MDP problem for VRPTW is thus to find an optimal policy to minimize the expected cost: $\pi^* = \arg\min_{\pi} \mathbb{E}[\sum_{k=1}^{N} c(s_t, a_t)|\pi]$.

3.2.2. *Hierarchical RL for VRPTW.* Hierarchical RL framework consists of a low level RL model and a high level RL model. With the hierarchical structure we solve VRPTW in two steps. Specifically, we first find feasible solutions in the low level and then search for an optimal solution in the high level. VRPTW is formulated as a multi-objective optimization

problem, which is a linear combination of the travel cost, the time penalty and the number of vehicles. $\alpha$, $\beta$, and $\gamma$ are the coefficients associated with the time cost $C_\sigma$, the time penalty $T_\sigma$, and the number of vehicles $M_\sigma$, respectively. In particular, the objective of the low level model is to minimize the time penalty and the number of vehicles:

$$\min_{\sigma} L_l(\sigma) = \sum_{k=1}^{N} \beta T_\sigma + \gamma M_\sigma, \tag{3.1}$$

where the total time penalty for violating the time windows is defined as $T_\sigma = \sum_{k=1}^{N} t_{v_k}$ with $t_{v_k}$ being the accumulated time violations.

The model trained at the low level is used as a hidden variable $H, h_t \in H$, at the high level. The input to the high level consists of a graph $G = (V, E, A)$ and the hidden variable $H$. In each state, the action $a_t$ is sampled from a policy $\pi(a_t|s_t, h_t)$ in the decoder of GPN at the high level. The high level RL model is trained to minimize the travel cost, the time penalty, and number of vehicles:

$$\min_{\sigma} L_h(\sigma) = \sum_{k=1}^{N} \alpha C_\sigma + \beta T_\sigma + \gamma M_\sigma, \tag{3.2}$$

where the total travel cost is defined as $C_\sigma = \sum_{k=1}^{N} c_{(k,k+1)}$, where $c_{k,k+1}$ is the travel cost from $v_k$ to $v_{k+1}$. Note that we use the number of vehicles $M_\sigma$ in the objective of both low level and high level to reduce the number of vehicle used.

In the above hierarchical RL framework, any RL algorithm can be used as a base algorithm and we use REINFORCE [17] in our study. To compare with the proposed hierarchical RL model, we also create a non-hierarchical RL model, which uses GPN and optimizes all costs in one layer.

$$\min_{\sigma} L_{non}(\sigma) = \sum_{k=1}^{N} \alpha C_\sigma + \beta T_\sigma + \gamma M_\sigma. \tag{3.3}$$

## 4. **Experiments**

To test the performance of our proposed algorithm, we compare it with non-hierarchical RL algorithm and Google OR-Tools. Google OR-Tools implement many heuristic algorithms for a variety of VRPs including VRPTW, and is often served as a baseline to represent the performance of traditional heuristic algorithms. Most deep learning models are applied to solve simple VRPs such as CVRP. As far as we know, only one previous work focuses on VRPTW using deep learning method [8], and the linear combination of the time cost and the accumulated time violation is used as the single performance metric in [8]. Since the experimental setup are different, For comparison, we perform experiments on the same Solomon dataset used in [8] with a more strict vehicle capacity.

We consider two types of TW constraints in VRPTW: 1) the hard constraint in which solutions that violate the time windows are deemed infeasible, and 2) the soft constraint in which the accumulated time violation $T_\sigma$ is added to the objective. In the experiments, hierarchical and non-hierarchical RL models search for the optimal number of vehicles for meeting requirements of the demand within the time windows. Therefore, we test both hierarchical and non-hierarchical RL models with the soft constraint, while we test OR-Tools with both the hard and the soft constraints.

Since in practice finding feasible solutions can be crucial for VRPTW, to capture both the feasibility and the efficiency for VRPTW, we adopt three metrics for performance evaluation: the accuracy, the time cost, and the number of vehicles. All evaluation is performed on 1000 test instances. Concretely, the accuracy denotes the percentage of problems with feasible

solutions, the time cost denotes the average time cost for all solutions, and the number of vehicles is the average number of vehicles used in these solutions.

### 4.1. **Datasets**

We test all algorithms on one dataset that is commonly used in literature and also generate a new datatset for VRPTW.

**Solomon dataset**: The Solomon dataset is generated based on statistics of R201 instance of the benchmark VRPTW problem Solomon [18]. Different from the Solomon data used in [8], we use a more strict vehicle capacity constraint: capacity 60 for 20 consumers and capacity 300 for 50 consumers. With a fixed number of vehicles, feasible solutions cannot be guaranteed on Solomon dataset. Solomon dataset is a relatively easy since the time window of each node can be large. We thus generate a new dataset using a local search algorithm 2-opt.

**2-opt dataset**: 2-opt dataset is generated using the heuristic algorithm 2-opt to ensure the existence of feasible solutions [19]. Given the number of vehicles $M$, the time window relaxation factor $tw$, and the capacity relaxation factor $cr$, one data sample is generated. Larger values of $tw$ and $cr$ indicate an easier VRPTW. The generation steps are described below.

(1) Randomly generate the coordinates $(X_0, X_1, ..., X_n)$ of $n+1$ nodes in $V$ in a square area of [0,1], where $v_0$ is the depot.
(2) Using K-Means on $v_1, ..., v_n$ generate $P$ clusters $Cl_1, Cl_2, ..., Cl_P$.
(3) For each cluster $Cl_j \cup V_0$, run 2-opt local search to find a TSP solution $\sigma_i$, which is a sequence of nodes in the cluster.
(4) Assign the demands to the nodes in each cluster:
    - Sample the demand $q_i$ uniformly in $\sigma_i$ in the range of [25, 75]
    - Normalize the demand in the group to $cr * \sum q_i$.
(5) Assign the time windows to the nodes in each cluster:
    - Run 2-opt on the centroids of clusters to determine the visiting sequence $Cl'_1, Cl'_2, ..., Cl'_P$.
    - Initialize the time counter $ct$ to 0.
    - From $Cl'_1$ to $Cl'_P$ do:
        - Reinitialize time counter $ct$ to 0 with only one vehicle.
        - Visit nodes in $\sigma_i$, and set the lower bound of the time window as $\max(0, ct - tw * \epsilon)$, and the upper bound of the time window as $ct + 1 + tw * \epsilon$, where $\epsilon \in [0.4, 1]$ is a random number.

In step 3, it is a simple and naive intuition that assigning a route to each cluster is better than creating links between different clusters, since nodes within a cluster are geometrically closer. After assigning nodes within a cluster to a route, building the route is then a TSP. The goal of 2-opt is to both create a feasible solution and to provide a baseline performance. Another difference in training RL models on two datasets is the hyper-parameters used. In training the hierarchical and non hierarchical RL models, $\alpha$ is set as 1, and $\beta$ and $\gamma$ are set as 10 at the low and high levels in 2-opt datasets. In Solomon datasets, $\alpha$ is used as 1, $\beta$ is set as 1750, and $\gamma$ is fixed as 10.

### 4.2. **Hierarchical RL**

4.2.1. *Problem size 20.* In Table 1, we compare the performance of hierarchical RL, non-hierarchical RL, the traditional heuristic algorithm 2-opt, and Google OR-Tools on VRPTW with 20 customers. With the number of vehicles $M$ fixed to be five, 2-opt and OR-Tools can find feasible solutions for all test cases, and OR-Tools performs better than 2-opt in terms of the time cost. For the hierarchical RL, we consider three different objectives in the high level: the time cost, the linear combination of the time cost and the time penalty,

| Method | Objectives | Accuracy | Time Cost | # Vehicles |
|---|---|---|---|---|
| 2-opt | time cost | **1.000** | 7.44 | **5.00** |
| OR-Tools | time cost | **1.000** | **7.03** | **5.00** |
| Non-hier RL | time cost + time penalty + #vehicles | 0.986 | 12.42 | 6.25 |
| Hier RL | time cost (high) | 0.830 | 7.19 | 5.03 |
| Hier RL | time cost + time penalty (high) | 0.970 | 7.27 | **5.00** |
| Hier RL (suggested) | time cost + time penalty + #vehicles (high) | **0.995** | **6.78** | **5.00** |

*Table 1.* Performance of hierarchical RL, non-hierarchical RL, and baselines on VRPTW with problem size 20.

| Setting | Method | Accuracy | Time Cost | # Vehicles | Time (s) |
|---|---|---|---|---|---|
| Easy problem | OR-Tools_hard | 1.000 | 7.091 | 5.000 | 22 |
| $tw = 2$ | OR-Tools_soft | **1.000** | 9.117 | 5.000 | 33 |
| | Non_hier RL | 0.997 | 7.371 | 5.031 | 19 |
| | Hier RL | 0.959 | **5.663** | **4.166** | 14 |
| Hard problem | OR-Tool_hard | 0.910 | 7.163 | 5.000 | 2240 |
| $tw = 1$ | OR-Tools_soft | 0.910 | 9.064 | 5.000 | 36 |
| | Non_hier RL | 0.992 | 8.397 | 5.799 | 18 |
| | Hier RL | **0.993** | **6.570** | **4.854** | 15 |

*Table 2.* Performance of hierarchical RL on easy and difficult problem settings with problem size 20.

and the suggested objective in Section 3.2.2. We found that neither of the hierarchical and the non-hierarchical RL can reach 100% feasibility on the test set. The hierarchical RL models perform much better than the non-hierarchical RL model in terms of the time cost and the number of vehicles. For the hierarchical models, the proposed one leads to a higher accuracy. Overall, the hierarchical RL model reaches comparable performance with traditional methods on problem size 20.

Furthermore, we test the hierarchical RL on easy and difficult problems by changing the time window relax factor $tw$ and the capacity relax factor $cr$. The comparison among hierarchical RL, non-hierarchical RL, and Google OR-Tools on problem size 20 is shown in Table 2. Google OR-Tools is tested on VRPTW with the hard (OR-Tool_hard) and the soft (OR-Tools_soft) constraint using a fixed number of vehicles as 5. Therefore, RL models are comparable to OR-Tools_soft.

Overall, RL models show better performance than OR-Tools on both the easy and the difficult VRPTW problems. The accuracy of OR-Tools obviously deceases when the problems become difficult, while the accuracy of RL models does not change much. The time cost of RL models increases with the difficulty, and the hierarchical RL uses less vehicles on both the easy and the difficult problems. The running time of OR-Tools on the difficult setting dramatically increases. This indicates that OR-Tools is not able to find feasible solutions for more difficult cases. Also, the hierarchical RL with comparable accuracy, a lower time cost, and a less number of vehicles performs better than the non-hierarchical RL. On the difficult setting with more strict time windows, the RL models show more resilience and outperform Google OR-Tools.

We also test the hierarchical RL and the OR-Tools on the easy and the difficult problems on Solomon dataset by varying the capacity of vehicles. In the easy problem setting we fix the vehicle capacity to 120, and in the difficult problem setting we reduce the vehicle capacity to 60. We then train two hierarchical RL models: Hier RL(120) and Hier RL(60). The number of vehicles used in OR-Tools is also adjusted from three to six in the easy setting, and from six to seven in the difficult setting to provide more flexibility. The results of RL models and OR-Tools are shown in Table 3.

| Problem | Method | Accuracy | Time Cost | #Vehicles | Time(s) |
|---|---|---|---|---|---|
| Easy | OR-Tools | 0.787 | **2859** | **3.000** | 8113 |
| 120 capacity | OR-Tools | 0.944 | 2891 | 4.000 | 2316 |
| | OR-Tools | 0.992 | 2912 | 5.000 | 635 |
| | OR-Tools | **0.999** | 2917 | 6.000 | 248 |
| | Hier RL(120) | 0.914 | 2935 | 3.357 | 11 |
| Difficult | OR-Tools | 0.719 | **4309** | **6.000** | 2544 |
| 60 capacity | OR-Tools | **0.958** | 4500 | 7.000 | 918 |
| | Hier RL(60) | 0.951 | 4474 | 6.256 | 13 |

*Table 3.* Problem size 20: Performance of hierarchical RL models with the vehicle capability 120 (easy) and 60 (difficult) on Solomon dataset.

| Method | Setting | Accuracy | Cost | # Vehicle |
|---|---|---|---|---|
| OR-Tools | 5 vehicles | 0.040 | 3689 | 5.00 |
| OR-Tools | 6 vehicles | 0.443 | 3996 | 6.00 |
| OR-Tools | 7 vehicles | 0.618 | 4058 | 7.00 |
| OR-Tools | 8 vehicles | 0.870 | 4178 | 8.00 |
| OR-Tools | Variable vehicles | **1.000** | **4371** | 7.46 |
| Hier RL(slow) | Variable vehicles | 0.733 | 5973 | 6.90 |
| Hier RL(fast) | Variable vehicles | 0.739 | 5805 | **6.27** |

*Table 4.* Performance of hierarchical RL models on Solomon dataset with problem size 50.

On Solomon dataset, hierarchical RL performs better than OR-Tools in the difficult problem setting. For easy problems, OR-Tools leads to a higher accuracy with more vehicles and less running time. With a similar number of vehicles on easy problems, Hier RL (120) achieves a higher accuracy and comparable time cost. Both Hier RL (60) and Hier RL (120) outperform OR-Tools with a higher accuracy and a less number of vehicles on difficult problems. Also, the running time of hierarchical RL is much less than OR-Tools.

On both datasets 2-opt and Solomon, we observe that the hierarchical RL model outperforms OR-Tools on difficult VRPTW problems, and performs comparable on easy problems. In general, 2-opt is a more challenging dataset than Solomon since it takes much longer time for OR-Tools to search for feasible solutions (see Tables 2 and 3).

4.2.2. *Problem size 50.* We further test the hierarchical RL model and OR-Tools on Solomon datasets with the problem size 50 (i.e., 50 customers). Two settings of OR-Tools are used: one with a fixed number of vehicles and the other with a variable number of vehicles. To stable the training of multi-objectives in our model, we use the warm-up technique during training by increasing the time windows penalty by 10 every 100 batches. We consider two hierarchical RL models: Hier RL (slow) which warms up from 100 to 2500, and Hier RL (fast) which warms up from 300 to 1800. The results of Hier RL modles and OR-Tools are shown in Table 4.

The hierarchical RL performs better than OR-Tools with a fixed number of vehicles, but worse with a variable number of vehicles on problem size 50. Similar to the experimental results on problem size 20, both the accuracy and the time cost of OR-Tools increase with the number of vehicles. With a variable number of vehicles OR-Tools reaches 100% accuracy and 7.46 vehicles on average. The hierarchical RL model outperforms OR-Tools using fixed seven vehicles. However, The hierarchical RL model has a lower accuracy and a higher time cost than OR-Tools with a variable number of vehicles. Therefore, if using a less number of vehicles has a higher priority than feasibility, Hierarchical RL model is still a good choice.

On problem size 20 and 50, the hierarchical RL model shows superior performance to the traditional approach OR-Tools on difficult problems with a less number of vehicles and less running time. Also, large-scale problems can be solved by our method in a timely fashion in practice.

| Model | Test dataset | Accuracy | Time Cost | # Vehicles |
|---|---|---|---|---|
| Hier RL | $tw = 4$ | 1.00 | 11.76 | 10.41 |
| $(tw = 4)$ | $tw = 1$ | 0.62 | 20.21 | 14.16 |
| Hier RL | $tw = 4$ | 0.93 | 15.41 | 6.17 |
| $(tw = 1)$ | $tw = 1$ | 0.99 | 24.04 | 14.64 |

*Table 5.* Generalization of the hierarchical RL model on 2-opt datasets with problem size 50.

| Model | Test data | Accuracy | Time Cost | # Vehicles |
|---|---|---|---|---|
| Hier RL | $tw = 4$ | 1.000 | 7.230 | 6.6 |
| $(tw = 3)$ | $tw = 3$ | 1.000 | 7.300 | 6.6 |
| Easy | $tw = 2$ | 0.992 | 7.490 | 6.6 |
| | $tw = 1$ | 0.888 | 8.212 | 6.6 |
| Hier RL | $tw = 4$ | 1.000 | 8.084 | 6.1 |
| $(tw = 1)$ | $tw = 3$ | 1.000 | 7.722 | 6.2 |
| Difficult | $tw = 2$ | 0.998 | 7.595 | 6.4 |
| | $tw = 1$ | 0.998 | 8.052 | 6.6 |
| Hier RL | $tw = 4$ | 1.000 | 7.138 | 6.4 |
| $(tw = 1 \sim 3)$ | $tw = 3$ | 1.000 | 7.169 | 6.4 |
| Mix-up | $tw = 2$ | 0.999 | 7.325 | 6.4 |
| | $tw = 1$ | 0.997 | 7.878 | 6.5 |

*Table 6.* Performance of hierarchical RL models trained on mix-up dataset ($tw = 1 \sim 3$) on 2-opt dataset with problem size 20.

### 4.3. **Generalization**

Generalization is important for real applications of VRP. Very often in practice the test distribution is different from the training distribution due to, e.g., the change of customers and constraints. We test the generalization of the hierarchical RL model for different scenarios and describe below.

4.3.1. *Generalization to different data distributions.* First we evaluate whether the hierarchical RL model trained on easy and difficult sets respectively can be used to generate feasible solutions on a mixed dataset with both easy and difficult data samples (see Table 5). The RL model is either trained on datasets with the easy time window constraints and tested on datasets with difficult time window constraints or vice versa. It shows that the RL model trained on the easy setting ($tw = 4$) fails to generate feasible solutions for around 40% difficult cases. In contrast, the RL model obtained from difficult cases ($tw = 1$) can reach a high level of feasibility with a much less number of vehicles on easy problems.

We call a mix-up strategy by mixing data samples from different distributions in training. We show the performance of generalization under mix-up in Table 6. Note that VRPTW becomes easier with a higher value of the time window factor $tw$. During the training of the hierarchical RL models, the time windows of all nodes are generated either using the same factor $tw = 1$ (difficult)/$tw = 3$ (easy) or a varied factor $tw = 1 \sim 3$. We found that the RL model trained under mix-up can reach a better performance than that trained with only one type of dataset distribution. In particular, the performance of the hierarchical RL model trained with mix-up is much better than that trained on easy problems regarding the three metrics considered, and the model also reaches a lower time cost than the hierarchical RL model trained in the difficult setting.

4.3.2. *Generalization across datasets.* In real applications, the number of customers may vary. In a few previous studies[3, 4], the generalization of deep learning models from small-scale to large-scale problems is tested and the deep learning models generally show a variety of generalization based on the network architecture. We evaluate the generalization of the hierarchical RL model trained on VRPTW problem size 20 to problem size 50 (see Table 7).

| Hier RL | Test dataset | Accuracy | Time Cost | # Vehicle |
|---|---|---|---|---|
| vrptw20 | vrptw20 $tw = 4$ | 1.000 | 7.303 | 6.578 |
| ($tw = 4$) | $tw = 3$ | 1.000 | 7.297 | 6.512 |
| | $tw = 2$ | 0.999 | 7.560 | 6.551 |
| | $tw = 1$ | 0.909 | 8.606 | 6.976 |
| vrptw20 | vrptw50 $tw = 4$ | 0.997 | 11.161 | 7.362 |
| ($tw = 4$) | $tw = 3$ | 0.967 | 11.509 | 7.532 |
| | $tw = 2$ | 0.833 | 12.467 | 7.928 |
| | $tw = 1$ | 0.259 | 17.331 | 10.280 |

*Table 7.* Generalization of the hierarchical RL model on 2-opt dataset with problem size 20.

The results show that the hierarchical RL model performs well on easy VRPTW problem size 50, but deteriorates dramatically on difficult cases. On easy VRPTW of problem size 50, the hierarchical RL model is able to find feasible solutions with only a moderately more number of vehicles. However, the accuracy of the hierarchical RL drops to 26% on difficult cases with $tw = 1$.

## 5. Discussion

Previous work that applies RL on VRP usually adopts a flat RL structure [3–5] and designs problem-specific network architectures [8]. For these methods, to solve more complex VRP, either the RL algorithm or the encoder-decoder network needs to be redesigned. For example, REINFORCE was used for VRP, but is not able to deal with the long trajectories in Stochastic VRP, so REINFORCE was replaced with A3C for Stochastic VRP [3]. Also, to solve VRPTW, multiple encoders of node, tour, and vehicle were designed to replace the standard one-node encoder in deep learning models [8].

Our study shows that the hierarchical RL is able to achieve superior performance for VRPTW. The hierarchical RL we considered is a general framework which is suitable for complex combinatorial optimization problems with multiple conflicting objectives. Feasible solutions are first generated in the lower level, and then an optimal solution is produced in the high level. In contrast, in the non-hierarchical framework, since the action space is much larger it generally takes a much longer training time locating the feasible regions.

## 6. Conclusions

In this study, we proposed a new hierarchical RL framework for solving VRPTW. The goal of VRPTW is to find an optimal policy which generates optimal routes for visiting customers within the time windows. We adopted graph pointer networks (GPNs) in the hierarchical RL framework. The GPN is based on an encoder-decoder architecture with the attention mechanism. The encoder includes the context vector and the pre-context vector. With the attention mechanism used on the context and the pre-context vectors, the decoder produces the next customer to be visited in the routes.

Experimentally, we considered three performance metrics for evaluation: the accuracy, the time cost, and the number of vehicles. We showed that the hierarchical RL model generally outperforms the non-hierarchical RL model and the traditional heuristic algorithm Google OR-Tools on two datasets Solomon and 2-opt. In addition, we showed that the hierarchical RL model reaches a better performance than Google OR-Tools especially on difficult VRPTW with more strict time window constraints. We improved the generalization of the hierarchical RL models by mixing up data from different time window constraints. The hierarchical RL model we proposed also shows generalization from small-scale to large-scale datasets, which has not been investigated in previous work.

In this work we have focused on VRPTW. We are also interested in using the hierarchical RL to solve other complex VRPs such as VRPPD with more challenging constraints.

Currently, only a few classical RL algorithms such as REINFORCE, A3C, and Q-learning have been used for combinatorial optimization problems. We would like to adopt more advanced RL algorithms such as soft A3C in the hierarchical RL model to further boost the performance.

## Acknowledgements

The authors would like to thank Artificial Intelligence for Logistics program at the National Research Council Canada for support.

## References

[1] J.-F. Cordeau et al. "Vehicle routing". In: *Handbooks in operations research and management science* 14 (2007), pp. 367–428.

[2] G. Laporte. "Fifty years of vehicle routing". In: *Transportation science* 43.4 (2009), pp. 408–416.

[3] M. Nazari et al. "Reinforcement learning for solving the vehicle routing problem". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9839–9849.

[4] E. Khalil et al. "Learning combinatorial optimization algorithms over graphs". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6348–6358.

[5] W. Kool, H. Hoof, and M. Welling. "Attention solves your TSP, approximately". In: *Statistics* 1050 (2018), p. 22.

[6] O. Vinyals, M. Fortunato, and N. Jaitly. "Pointer networks". In: *Advances in neural information processing systems*. 2015, pp. 2692–2700.

[7] I. Bello et al. "Neural combinatorial optimization with reinforcement learning". In: *The International Conference on Learning Representations*. 2017.

[8] J. K. Falkner and L. Schmidt-Thieme. "Learning to Solve Vehicle Routing Problems with Time Windows through Joint Attention". In: *arXiv preprint arXiv:2006.09100* (2020).

[9] T. Haarnoja et al. "Latent space policies for hierarchical reinforcement learning". In: *arXiv preprint arXiv:1804.02808* (2018).

[10] N. Mazyavkina et al. "Reinforcement learning for combinatorial optimization: A survey". In: *arXiv preprint arXiv:2003.03600* (2020).

[11] X. Li et al. "A data-driven three-layer algorithm for split delivery vehicle routing problem with 3D container loading constraint". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 528–536.

[12] A. Bortfeldt and J. Yi. "The Split Delivery Vehicle Routing Problem with three-dimensional loading constraints". In: *European Journal of Operational Research* 282.2 (2020), pp. 545–558.

[13] X. Chen and Y. Tian. "Learning to Perform Local Rewriting for Combinatorial Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 6281–6292.

[14] H. Lu, X. Zhang, and S. Yang. "A learning-based iterative method for solving vehicle routing problems". In: *International Conference on Learning Representations*. 2019.

[15] Q. Ma et al. "Combinatorial Optimization by Graph Pointer Networks and Hierarchical Reinforcement Learning". In: *arXiv preprint arXiv:1911.04936* (2019).

[16] T. N. Kipf and M. Welling. "Semi-supervised classification with graph convolutional networks". In: *International Conference on Learning Representations*. 2017.

[17] R. J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[18] M. M. Solomon. "Algorithms for the vehicle routing and scheduling problems with time window constraints". In: *Operations research* 35.2 (1987), pp. 254–265.

[19] J.-Y. Potvin and J.-M. Rousseau. "An exchange heuristic for routeing problems with time windows". In: *Journal of the Operational Research Society* 46.12 (1995), pp. 1433–1446.