ITC 1/51
Information Technology and Control
Vol. 51 / No. 1 / 2022
pp. 126-138
DOI 10.5755/j01.itc.51.1.29924

Constraint-aware Policy Optimization to Solve the Vehicle Routing Problem with Time Windows

Received 2021/10/02 | Accepted after revision 2022/01/25

http://dx.doi.org/10.5755/j01.itc.51.1.29924

HOW TO CITE: Zhang, R., Yu, R., Xia, W. (2022). Constraint-aware Policy Optimization to Solve the Vehicle Routing Problem with Time Windows. *Information Technology and Control*, 51(1), 126-138. https://doi.org/10.5755/j01.itc.51.1.29924

Constraint-aware Policy Optimization to Solve the Vehicle Routing Problem with Time Windows

Renchi Zhang
School of Management, Hefei University of Technology, Hefei 230009, China; Key Laboratory of Process Optimization and Intelligent Decision-making, Ministry of Education, Hefei 230009, China;
e-mail: rczhang@mail.hfut.edu.cn

Runsheng Yu
Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, HKG HKSAR;
e-mail: runshengyu@gmail.com

Wei Xia
School of Management, Hefei University of Technology, Hefei 230009, China; Key Laboratory of Process Optimization and Intelligent Decision-making, Ministry of Education, Hefei 230009, China;
e-mail: xiawei@hfut.edu.cn

Corresponding author: xiawei@hfut.edu.cn

The vehicle routing problem with time windows (VRPTW) as one of the most known combinatorial operations (CO) problem is considered to be a tough issue in practice and the main challenge of that is to find the approximate solutions within a reasonable time. In recent years, reinforcement learning (RL) based methods have gained increasing attention in many CO problems, such as the vehicle routing problems (VRP), due to their enormous potential to efficiently generate high-quality solutions. However, neglecting the information

between the constraints and the solutions makes previous approaches performance unideal in some strongly constrained problems, like VRPTW. We present the constraint-aware policy optimization (CPO) for VRPTW that can let the agent learn the constraints as a representation of the whole environment to improve the generalization of RL methods. Extensive experiments on both the Solomon benchmark and the generated datasets demonstrate that our approach significantly outperforms other competition methods.

KEYWORDS: Deep reinforcement Learning, Pointer network, Vehicle routing Problem, Gradient methods, Kullback-Leibler divergence, Constrained Markov Decision Processes.

1. Introduction

The vehicle routing problem (VRP), as an enduring problem of operations research, has been studied for decades due to its wide application in various fields, like logistics, transportation, and manufacturing [6]. Such problem aims to find the optimal routes for available vehicles to travel in order to satisfy the demands of customers under certain constraints [16]. Many exact, approximate, and heuristic methods have been proposed, some of them are state-of-the-art [7, 8, 25, 30]. With the development of deep neural networks, deep reinforcement learning (DRL)-based methods, as a kind of heuristic method, have gained attention due to their enormous potential to efficiently generate high-quality solutions [3, 13, 23].

The DRL-based method is to solve the VRP by training a neural network (NN) model that can map between the state space and the optimal solution. The model consists of many components, the most representative of which is the representation learning and reinforcement learning (RL). Representation learning can exact the efficient feature vectors from the raw data, while RL is to train the whole NN without the optimal solutions as the labels. Recent research mainly focuses on the improvement of representation learning to advance performance. Vinyals et al. [31] presents the pointer network that first treats each option element as a pointer and employs the recurrent neural networks (RNN) to represent the dynamic changes of the features. Bello et al. [3] introduces the attention mechanism to put weights on the different features vectors to improve the probabilities of the pointing mechanism. Nazari et al. [23] and Kool et al. [15] design a method with novel attention mechanism respectively that outperforms the state of the art.

However, the previous work lacks further research on constraints, especially its impact on reinforcement learning. Because reinforcement learning is a way of learning through trial-and-error exploration [3, 13, 15, 23], the quality of trained policy is greatly affected by the exploration strategy under constraints that define the boundary of the solution space [26, 28]. When using reinforcement learning to train neural network models, these methods only mask the solutions that do not satisfy the constraints through 'mask', and do not consider how to make the agent aware of the existence of constraints in a certain state. The neglect of information contained in constraints makes them difficult to find the optimal solutions. Intuitively, as constraints become more complex, the awareness of constraints becomes more necessary.

The VRP with time windows (VRPTW), as a variant of VRP with complex time window constraints, has been widely studied and is ($\mathcal{NP}$)-hard to solve. In such problem, a fleet of identical vehicles serve multiple customers along optimal routes subject to the following constraints: (1) each vehicle can only start from, end at, and acquire items from the depot; (2) each vehicle must visit customers within a specified time interval (time window); (3) total demands of customers served by a single vehicle cannot exceed its capacity; and (4) all demands must be satisfied. The objective is to minimize the sum distance of all tours. The existence of mass complex constraints limits the performance of previous methods, so it is necessary to improve the learning process of RL.

To investigate these problems, we present constraint-aware policy optimization (CPO) to impose on our agent to learn the probability distribution of constraints. A method based on Kullback-Leibler (KL) divergence is constructed to calculate the distance of probability distribution between the constraint and the policy, which can obtain the

gradient to guide learning. Our work comprises four main steps. 1) To formally represent the VRPTW in a sequential decision process for RL, we convert the constrained route planning problem to a deterministic constrained Markov decision process (DCMDP), to which we extend the original policy gradient method. 2) To guide the policy to learn the features of constraints, a constraint-aware training scheme is proposed, which can enhance our method's performance, including a predictive part to predict the current constraints according to the state information, and an inference part to make decisions according to the constraints. 3) To alleviate the sparse reward problem, we use successor representation (SR) as an indicator to guide the choice of actions. 4) An RL training framework for the VRPTW and experiments on the Solomon benchmark and the generated datasets show that our method outperforms other competition algorithms.

The remainder of this paper is organized as follows. Section 2 provides an overview of related works. Section 3 discuss the basic ideas of the deterministic constrained Markov decision process, VRPTW, and successor representation. Three key components of the CPO are presented in Section 4. In order to verify our method, the experiments are shown in Section 5. In Section 6, we present the conclusion.

2. Related Work

Our work is closely related to integral linear programming, heuristic approaches of the VRP, learning-based routing, and constrained RL.

2.1. Integral Linear Programming and Heuristic Approaches

These approaches are commonly used to solve the VRP and its variants, and are formulated as mixed integer linear programming (MILP) problems.

Branch and bound is one of the most famous exact algorithms for the VRPTW. It represents candidate solutions as a tree, and prunes solutions that are beyond the slack upper bound [30]. This approach can find the optimal solutions, but it becomes intractable as the number of decision objects increases. Conversely, heuristic approaches, such as genetic and ant colony algorithms, prefer scalability to optimality, and thus can find approximate solutions of large-scale problems in a relatively short time. However, heuristic algorithms are sensitive to parameters and weak in robustness. All the above methods are designed to solve problems case-by-case. They have to search solutions anew when they face a new instance, which is time-consuming. That is the major obstacle of these methods in practice.

2.2. Learning-Based Routing

Learning-based routing approaches are common due to their efficient solutions and stable performance. Their methods are based on either supervised learning (SL) or RL. Introduced by Hopfield et al. [12], SL-based methods aim to leverage neural networks to find solutions in a supervised manner. Since they require expert guidance, SL methods are intractable in environments where traditional methods cannot work [13, 17]. RL-based methods attempt to find approximate solutions through trial and error [3, 13, 15, 23]. While capable of good performance, they do not work well when the constraints become intricate. Our method is different in that we attempt to enhance the performance through a constraint-awareness learning scheme and an indicator to guide our policy to learn from constraints.

2.3. Constrained Reinforcement Learning

Constrained RL is a trending topic because in real-world agents are strictly restricted to avoid dangerous actions [1, 4, 32]. Widely used methods include expert-guided action intervention [32], Lagrangian relaxation [29], and probability guided exploration [4]. We take motivation from them, but VRP constraints are deterministic to the environment (as discussed in the next section). We leverage these properties to develop a training scheme that makes our method more robust and stable.

3. Background

We discuss the basic ideas of the deterministic constrained Markov decision process (MDP), VRPTW, and successor representation.

3.1. DCMDP

A deterministic constrained Markov decision process can be formulated as a tuple ⟨ S, A, R, p, T, C ⟩, where S is a set of states, A is a set of actions, and $p(s_{t+1}|s_t, a_{t+1})$ is a transition function. Different from the standard MDP method, a set of constraints C is represented in deterministic constrained MDP, which varies by the environment (load constraints in VRP, and load and time- window constraints in VRPTW). Let c_t^i: $S \rightarrow \{0,1\}$ indicate the constraint for action i at step t; $c_t^i = 0$ means that the constraint exists, and $c_t^i = 1$ means that the constraint non-exists. $r : S \times A \times S \rightarrow R$ is the reward function. The goal of RL is to find a policy π: $S \times C \rightarrow A$ that maximizes the expected return $R = \sum_{t=1}^{T} \gamma^{t-1} r_t$ in the trajectory $\tau := (s_1, c_1, a_1 \ldots, s_T, c_T, a_T)$ within T steps. γ is a discount factor.

The DCMDP is a special case of a constrained MDP [2]whose constraints are deterministic, meaning that $p(c_t|s_t) = 1$ ($p(c_t|s_t) = 1$ indicates that the relationship between constraints and the environment is certain; hence, learning the constraints implies learning the environment as well) and is known to the agent before taking action. This property leads to a different solver from Altman et al. [2].

3.2. VRPTW

For VRP, suppose there exist a depot and $|N| - 1$ customers in different locations, for $|N|$ total nodes, where N is the set of nodes. The depot is a special node at which vehicles must start, end, and collect items. For any nodes i, j, we have $d^{i,j} > 0$, $\forall i, j \in N$, $i \neq j$, where $d^{i,j}$ is the distance between nodes i and j. The vehicle sends items to customers and should return to the depot when its residual carrying capacity cannot satisfy any customer demand. Given enough vehicles having the same capability l and starting from the depot, our goal (objective) is to satisfy all the customers' demands with the shortest path. The environment and goal of the VRPTW are similar to those of the VRP except that each customer owns its individual available time wi (or time windows). Moreover, we assume that all vehicles have the same velocity v and must arrive within a customer's time window.

VRP are often represented as MILP. However, MILP does not fit the RL form well (since RL is always represented as a Markov decision process) and to put it in the RL framework requires a transformation. It has been proved that a traveling salesman problem (TSP) can be transformed to an MDP [3]. To strengthen this conclusion, we provide a lemma.

Lemma 1. Standard VRP and VRPTW problems can be converted to DCMDP.

Proof 1. Through giant-tour representation [10], the VRP and VRPTW can be treated as a special form of the TSP (i.e., in some states, some actions are restricted and cannot be chosen). According to Bello et al. [3], the TSP can be transformed to an MDP. Thus, we can conclude that the VRP and VRPTW can be formed as a DCMDP.

Remark. Lemma 1 reveals the interesting phenomena that we can formulate VRP as a DCMDP and represent the load and time windows as constraints.

The trajectory probability of the total process for the DCMDP is

$$p(\tau) = p(s_1, c_1, a_t, c_t \ldots, s_T, c_T, a_T|\theta) = p(s_1) \prod_{t=1}^{T} \pi(a_t|s_t, c_t; \theta) p(s_{t+1}|s_t, a_t) p(c_t|s_t), \quad (1)$$

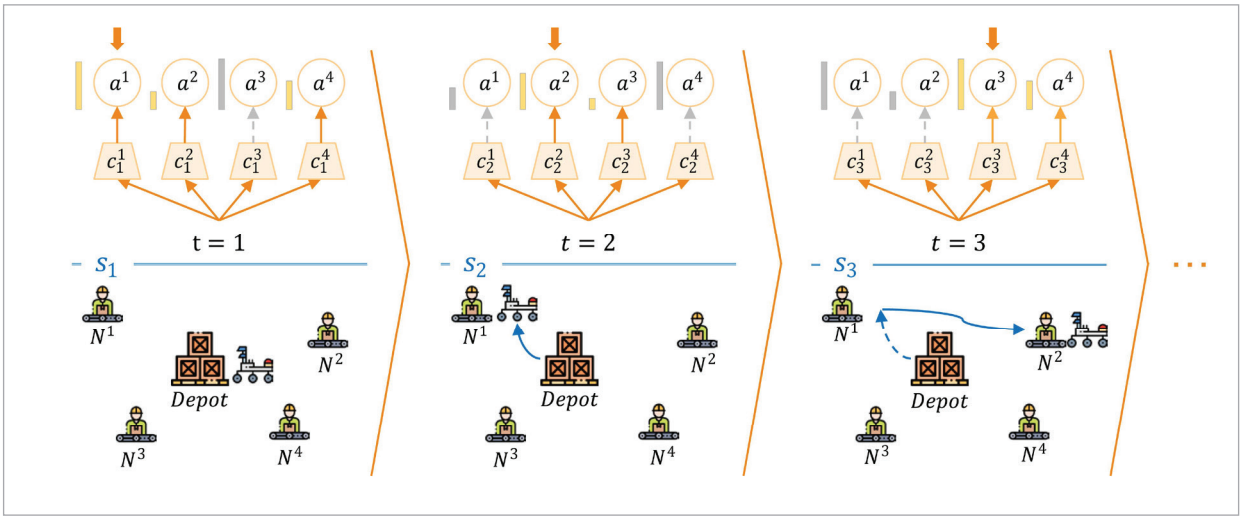
where θ represents the weights of the policy, sT+1 is the terminal state, and $p(\boldsymbol{s_{T+1}}|\boldsymbol{s_T}, \boldsymbol{a_T}) = 1$. This trajectory is based on the probability graph model shown in figure 1. Specifically, for the VRP, $\boldsymbol{c}$ means the load constraints, while for the VRPTW, $\boldsymbol{c}$ means the constraints of both the load and time windows of customers. Our goal is to find an optimal policy $\pi*$ that maximizes the total reward R.

3.3. Successor Representation

Introduced to describe cognitive phenomena in the human brain, successor representation (SR) focuses on the extraction of important states to aid training [20]. We leverage SR as a behavior indicator, regarding states that may lead to a lower total reward as ill-famed, and using SR as a behavior indicator to avoid actions to those states. For example, in the VRP, we might not be willing to let our vehicle return too soon to the depot; the ill-famed state is the situation in which the agent is at the depot without having depleted its load.

Figure 1

The rollout processes. A vehicle starts from the depot and must deliver items to customers (represented as $N^1 \ldots N^4$). For example, in the first step, nodes 1 and 3 are banned for some reason (the arrow from constraint to action is black). Thus, actions can only be chosen in a^1 and a^4. Under the greedy strategy, the vehicle chooses the action with the maximum probability. This continues from step 1 to T, forming the rollout trajectory



4. Methodology

Our methodology has three parts: first, we give the policy gradient for the DCMDP to formally represent a simple solver for VRP; then, a constraint-awareness module is represented to diminish actions against constraints; finally, an SR-based method is designed to address the sparse reward problem.

4.1. Policy Gradient in DCMDP

A policy gradient for the DCMDP must be built to train our policy function. The trajectory probability $p(\tau)$reveals the rollout process. Taking logarithms on both sides, we have

$$\log p(\tau) = \log p(\boldsymbol{s_1}) + \sum_{t=1}^{T} \log \pi(\boldsymbol{a_t}|\boldsymbol{s_t}, \boldsymbol{c_t}; \theta) + \log p(\boldsymbol{s_{t+1}}|\boldsymbol{s_t}, \boldsymbol{a_t}) + \log p(\boldsymbol{c_t}|\boldsymbol{s_t}) \quad (2)$$

To generalize, we take the expectation of trajectories, and with Jensen's inequality [24], the expectation can be formed as

$$\log E_{\tau\sim p(\cdot)}\, p(\tau) \geq E_{\tau\sim p(\cdot)} \left(\log p(s_1) + \sum_{t=1}^{T} \log \pi(\boldsymbol{a_t}|\boldsymbol{s_t}, \boldsymbol{c_t}; \theta) + \log p(s_{t+1}|s_t, a_t) + \log p(c_t|s_t)\right). \quad (3)$$

The right side of the equation is the lower bound of the *log* form of the expectation of trajectories. With reinforcement learning [28], the function $E_{\tau\sim p(\cdot)}(\log p(\tau) R(\tau))$ can be written as

$$E_{\tau\sim p(\cdot)} \left[\left(\log p(s_1) + \sum_{t=1}^{T} \log \pi(a_t|s_t, c_t; \theta) + \log p(s_{t+1}|s_t, a_t) + \log p(c_t|s_t)\right) R(\tau)\right] \quad (4)$$

However, this form of policy gradient may lead to high variance, which restricts the ability to generalize. To reduce the variance, we define the baseline function $b(s_t, c_t) : C \times S \rightarrow R$ which we show in Equation (5) is unbiased:

$$\nabla_{\theta} E_{\pi(a_t|s_t, c_t; \theta)} \left(b(s_t, c_t) \log \pi(a_t|s_t, c_t; \theta)\right) = b(s_t, c_t) \frac{\partial}{\partial \theta} \int \pi(a_t|s_t, c_t; \theta) \mathrm{d}\, a_t = 0 \quad (5)$$

$b(s_t, c_t)$ is similar to the baseline methods in standard variance reduction RL. Differently, we extend the baseline function with constraints as input.

However, $b(s_t, c_t)$ is for the one-step baseline function. To extend it to the trajectory form, we can simply sum them up as $\hat{b}(\kappa) = \sum_{t=1}^{T} b(s_t, c_t)$, or build a neural net-

work to represent it as $\hat{b}$ (κ; λ), where \lambda denotes the weights for the neural network and κ = [s_1, c_1, s_2, c_2... s_T, c_T] is the vector of whole states and constraints for a trajectory. Since $p(s_{t+1}|s_t, a_t)$ and $p(c_t|s_t)$ are irrelevant to θ, based on the analysis above, we have the policy gradient for DCMDP:

$$\nabla_\theta E_{\tau\sim p(\cdot)}\left[\sum_{t+1}^{T} log\,\pi\,(a_t|s_t, c_t; \theta)\left(R - \hat{b}(\kappa)\right)\right] \quad (6)$$

4.2. Constraint-awareness Policy Optimization

Equation (6) provides a way to update the policy function. However, to only use the constraints as input might still not be enough to find information in the constraints (e.g., the relationship between environment and constraints). To further capture this information, we design a method to learn the constraints.

First, we notice that the relationship between constraints, environment, and actions can be represented as $\pi(a_t|s_t) = \sum_{c_t^i} \pi(a_t|s_t, c_t^i)p(c_t^i|s_t)$, where i is the index of the constraint at time t. This indicates that if trained properly, the policy π can implicitly learn the constraints. We assume there exists a strategy $\pi(a_t|s_t)$ that can find the best action under state s_t. Regarding the constraints as the hidden variables, we have:

$$\log \pi\,(a_t|s_t) = \log \int p(a_t, c_t|s_t)dc_t \geq E_{P(c_t|s_t)} \log \frac{p(a_t, c_t|s_t)}{p(c_t|s_t)} \quad (7)$$

$$D_{KL}\big(p(a_t, c_t|s_t) \parallel p(a_t, c_t|s_t)\big) \leq D_{KL}\big(p(a_t|s_t, c_t) \parallel p(c_t|s_t)\big). \quad (8)$$

Equations (7) and (8) are the form of the evidence lower bound. From information theory [33], we have

$$\log \pi\,(a_t|s_t) = E_{c_t\sim P(c_t|s_t)} \log \frac{p(a_t, c_t|s_t)}{p(c_t|s_t)} - D_{KL}\big(p(a_t, c_t|s_t) \parallel p(c_t|s_t)\big) \quad (9)$$

Thus, to maximize $\log p\,(a_t|s_t)$ is equivalent to minimizing $D_{KL}(p(a_t, c_t|s_t) \parallel p(c_t|s_t))$.

Notice also that for VRP, the relationship between constraints and the environment is deterministic, meaning that $p(c_t|s_t) = 1$, as mentioned above. Moreover, $p(c_t|s_t)$ has the same dimension with action, and for each constraint i at time t, $c_t^i = 1$ means that the constraint takes no effect. With those conditions, we have

$$p(a_t, c_t|s_t) = \pi(a_t|c_t, s_t)p(c_t|s_t) \quad \text{s.t.} \quad p(c_t|s_t) = 1; c_t^i \in \{0,1\}, \forall i. \quad (10)$$

Thus, the KL term can be simplified to

$$\min D_{KL}\left(p(a_t, c_t|s_t)|p(c_t|s_t)\right) = \min D_{KL}\left(\pi(a_t|c\}_t, s_t)|p(c_t|s_t)\right). \quad (11)$$

We leverage the max entropy strategy to induce exploration [21]. We combine constraint-awareness and the max entropy strategy and rewrite $p(c_t|s_t)$ in vector form as $\hat{p}(c_t|s_t) = \left[\frac{1(c_t^i=1)}{|C_t^0|}\right]_{i=1}^{N}$ where C_t^1 is a set of constraints that are one (constraints take effect) at time step t and $[\cdot]_{i=1}^{N}$ is a vector with N elements. $\hat{p}(c_t|s_t) = \left[\frac{1(c_t^i=1)}{|C_t^1|}\right]_{i=1}^{N}$ indicates that when node constraint of node i exists: $c_t^i = 0$. In this condition, the probability to choose action to that node is zero. We also average all available actions and minimize the distance between $\hat{p}$ and p through the KL term with a policy to encourage exploration.

Constraint-awareness policy optimization (CPO) can be formulated as follow:

$$\nabla_\theta E_{\tau\sim p(\cdot)}\left[\sum_{t+1}^{T} log\,\pi\,(a_t|s_t, c_t; \theta)\left(R - \hat{b}(\kappa)\right) - \beta \sum_{t=1}^{T} KL\big(\pi(s_t|c_t, s_t; \theta) \parallel \hat{p}(c_t|s_t)\big)\right], \quad (12)$$

where β is a positive parameter to balance the objective function and KL term.

Remark. Because of the deterministic constraints in VRPs, through learning the constraints, agents can implicitly learn the dynamic functions of the environment, and hence can know the results of choosing a certain action, especially worse actions. Thus, our method is also called implicit CPO (ICPO).

4.3. Behavior Indicator

There is still one issue. The rewards are always sparse for VRP [23], that is, agents can only access the final reward at the last step, resulting in a bad credit assignment problem [11]. Hence, to make accurate credit assignments is crucial. A property of VRP is that there exist such actions that the more you choose the lower the expected rewards is. To leverage this property, we choose SR as the tool to indicate these bad behaviors.

Recall the general form of the reward function:

$R = \sum_{t=1}^{T} r(s_t, a_t)$. Due to the nature of VRP, the reward is only available at the end of an episode: $R = r(s_T, a_T)$ To mitigate the sparse reward, we take the ill-famed states into consideration and rewrite the total reward as $\hat{R} = \sum_{t=1}^{T}(s_t \in \hat{S}) + R$, where $\hat{S}$ is a set of ill-famed states and $s_t \in \hat{S}$ means that the state into time t, s_t is the ill-famed state. As mentioned above, an ill-famed state is one that may lead to low total reward. We add a negative term because we want these states to appear as little as possible. Now the ill-famed state term $1(s_t \in \hat{S})$ can be trained by SR as

$$V(s) = -E\left(\sum_{t=1}^{T} \gamma^t 1\left(s_t \in \hat{S}\right) | s_0 = s\right) = -1\left(s_t \in \hat{S}\right) + \gamma E\left(V(s_{t+1})\right). \quad (13)$$

The right-hand side and it has the same recursive format as temporal difference [28].

With the help of SR, agents will take possible bad behaviors into account to make better decisions. ICPO with SR is expressed as follow:

$$\nabla_\theta E_{\tau\sim\hat{p}(\tau)}\left[\sum_{t=1}^{T} log\,\pi\left(a_t|s_t, c_t; \theta\right)\left(\alpha V(s_t) + R - \widehat{(\kappa)}\right) - \beta \sum_{t=1}^{T} \mathrm{KL}\left(\pi(a_t|c_t, s_t; \theta) \parallel p(c_t|s_t)\right)\right], \quad (14)$$

where α is a positive hyperparameter to control the influence of successors.

Remark. 1) We emphasize that SR can be extended in the opposite way. That is, we can maximize consideration of actions that are encouraged. The formulation is the same, but with a positive signal. 2) SR also acts as a regulator to balance human intuition and learning results.

4.4. Training Method

Loss Functions. We design a policy, baseline function, and successor value as neural networks with parameters θ, λ, and η, respectively. The losses of successor value L_v and baseline function L_b are

$$L_v = E_{\tau\sim z}\left[\left(\widehat{R} - \hat{b}(\kappa; \lambda)\right)^2\right] \quad (15)$$

$$L_b = E_{s_{t+1}, s_t\sim z}\left[\left(1\left[s_t \in \hat{S}\right] + \gamma V(s_{t+1}; \eta) - V(s_t; \eta)\right)^2\right], \quad (16)$$

where z is the episode buffer. All the loss functions can be estimated through Monte Carlo sampling. The policy gradient with constraint-aware module and successor representation can be formed as Equation (17).

$$\nabla_\theta E_{\tau\hat{p}(\tau)}\left[\sum_{t=1}^{T} log\,\pi\left(a_t|s_t, c_t; \theta\right)\left(\alpha V(s_t) + R - \hat{\kappa}\right) - \beta \sum_{t=1}^{T} \mathrm{KL}\left(\pi(a_t|c_t, s_t; \theta) \parallel p(c_t|s_t)\right)\right]. \quad (17)$$

The pseudocode can be found in Algorithm 1.

Algorithm 1

Implicit Constraint-awareness policy optimization

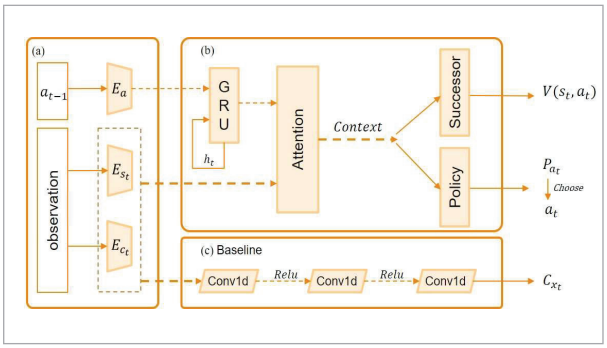
Algorithm

Initializing parameters for actor, successor, and baseline function θ, λ, and η
Generating training dataset
while not convergence **do**
  Sample s_0 from the training dataset
  Initializing the history information s_0
  Initializing the episode buffer z_0
  Rollout Stage
  for 0 to the maximum of nodes **do**
    Select action $a_t \sim \pi(s_t)$ with Boltzmann exploration strategy
    Update the environment with dynamic function
    $z_{t+1} = z_t \cup (s_t, a_t, r_t)$
  end for
  Train Stage
  Update θ, λ, and η through Equations (17), (15) and (16) respectively
end while

Network Structure. The policy network structure has three parts: decision making, encoder, and attention mechanism [3, 23]. The first is used to choose the policy, and the second and third to encode the graph and capture valuable information. A gated recurrent unit is introduced to seize the long-term effects [5]. We leverage the attention mechanism output as the input of the successor function. For the baseline function, since the state and constraints are taken as input, we build the model without sharing variables with the policy networks.

For full understanding of our network, we explain each structure. Our network structure, shown in Figure 2, is mainly built from Nazari et al. and Vinyals et al. [23, 31], and we make some improvements. Unless otherwise mentioned, the activation is ReLU [22].

Figure 2
Structure of policy, baseline, and successor networks. The structure has three parts: (a) the encoder network aims to convert states to embeddings; E_a, E_{s_t}, and E_{c_t} are encoders for action states and constraints, respectively; (b) the policy and successor; h_t is the long-term hidden state for RNN, and context is useful information extracted from the attention model; and (c) baseline function



Objective function. The objective function for VRP is to minimize the tour length, but the objective function in RL is always represented as maximizing the total reward. Therefore, we set the total reward in VRP as the negative tour length

Input. In the VRPTW, we set the current state and constraints as input. The state s_t for the VRP includes the location of the nodes, remaining load, and demand of customers. For VRPTW, since each customer has its own time window, we add two features to provide extra information: 1) the time window of each customer; and 2) the current time. Moreover, the previous action is also added as input to reveal the current location of the vehicle. The input size is $batch \times 2 \times |N|$ for the location of nodes and the time window. The size for other features is $batch \times 1 \times |N|$.

Attention Mechanism. An attention mechanism (AM) captures the internal relationships within a graph, where the embedding taken from GRU is used as the input embedding. The output of AM is a context, which is a 32-dimension vector combining short-term and long-term information. The setting of AM is similar to luong et al. [19].

Decision Making Network. The output of the decision-making network is $|N|$, and the input is the context from the AM. Softmax is used to generate the probability of each action (the activation function in the final layer is softmax).

Successor. A successor is an indicator of possible bad behaviors. We take the context as input, and the output dimension is 1 (the successor value is a scalar).

Baseline Function. The baseline function consists of three 1×1 convolutional neural networks to extract state and constraint information. The output dimension is 1, a scalar.

Action encoder. The action encoder converts the last step action to a vector. The output is of size 16.

5. Experiments

To verify the effectiveness of our approach, we conducted extensive experiments on the VRPTW on two different datasets: a generated dataset and the Solomon benchmark [27]. A well-known dataset for VRPTW studies, the Solomon benchmark, contains multiple instances in three scales: 25, 50, and 100. Like most learning-based methods, our approach requires substantial training data, and its precision advantage is more reflected in statistics, it is necessary to build a generated dataset as the supplement of Solomon benchmark which just has decades instances. Based on the following rules, we randomly generated 100,000 training samples and 1,000 testing samples for each scale of the VRPTW[1] and compared our approach to other baselines on two datasets.

5.1. Generated Dataset

The VRPTW is similar to the VRP, but each customer has its own time window. We generated 10, 20, 50, and 100 nodes with random locations and demands [23]. Each node was randomly located in a two-dimensional discrete coordinate system with range [0,100], and its demand had a uniform distribution $D_i \sim U(1,10)$. The capabilities of vehicles were 20, 30, 40, and 50 for size 10, 20, 50, and 100, respectively.

Assume that at time step t, a vehicle with current load l_t is preparing to send items to customer i, who requires ϵ^i items. When $\epsilon^i \leq l_t$, the vehicle can send the item, and the remaining load becomes $l_{t+1} = \epsilon^i - l_t$. Otherwise, the trade can not be established. Moreover, when no customer can be satisfied, the vehicle is forced to return to the depot.

[1] The source code can be visited in https://gitee.com/MARL_Researcher/vrptw-generator.git

The time window is the key difference between the VRP and VRPTW. Since the VRPTW must consider the time, the time step t no longer reveals the time interval. Here, we assume that each step indicates a transition to a new step, and we denote the time interval as $\hat{t}$. We formally illustrate the update of t and $\hat{t}$ below. We assume the velocity of a vehicle is constant; hence, time is proportional to distance. To avoid the case when a vehicle will never reach some customers, the farthest distance from the depot $\max_i d^{0,i}$ is smaller than its time window w^i.

Dynamic Function and Constraints. After distinguishing the time line and time step. we can define the transition function. The dynamic function and constraints of the VRPTW are based on the VRP. Suppose that at time step t, a vehicle is at location i and decides to go to location j with distance d^{ij}. Recall that we set the velocity as a constant v. Then the change of time is $\hat{t} \leftarrow \hat{t} + \frac{d^{i,j}}{v}$; $t \leftarrow t + 1$. When the current time $\hat{t}$ is not in the range of customer i's time window, $\hat{t} \notin w_i$, the trade cannot be established. Similar to VRP, when no customer can be satisfied, the vehicle is required to return to the depot.

Reward Setting. Common objective functions of the VRPTW include the delivery percentage, total tour length, or both [16]. In our experiments, we chose the total tour length as our objective function, and the total reward was the negative tour length.

5.2. Experiment Setup

We trained our model on a single GeForce RTX 2080, using Adam as the optimizer [14]. For each scale of the model, we performed 10 cycles of training on a training set containing 100,000 instances. The model is considered to have converged at the end of the training. The training time for the VRPTW is 4h, 7h, 11h, and 21h respectively, for 10, 20, 50, and 100 nodes, with a batch size of 256. Boltzmann exploration was used to improve the quality of our method. We used beam search (BS), a widely used optimization method in natural language processing, as a search strategy [13, 15], and γ was 0.95.

Baseline. To reveal the effectiveness of our method, we considered the following baselines: 1. The genetic algorithm (GA) was used as the heuristic algorithm baseline, which performs well in the VRPTW [18]. 2. For the search algorithm, we chose the nearest neighbor (NN) [15]. 3. For the RL baseline method, we took a state-of-the-art RL method as the baseline [23]. 4. We also compared Google OR tools [9], a fast and portable software suite to solve combinatorial optimization problems, including VRP.

Ablations. We conducted substantial ablation studies: 1) ICPO: our complete method. 2) ICPO w/o SR: remove the SR from our framework. 3) ICPO w/o KL: remove constraint-awareness from our framework. 4) Original policy gradient (PG): remove KL, SR, and constraints as input (the original PG method).

5.3. Results

Performance. Table 1 presents the results of several algorithms run on the generated dataset where the capabilities of vehicles were 20, 30, 40, 50 for size 10, 20, 50, 100, respectively and each size contains 1000 instances. Every two columns present the mean of total distance and total CPU time in a kind of environment with different number of nodes. As shown in Table 1, our approaches dramatically outperformed other baselines in most VRPTW environments, and they had the fewest outliers. In particular, the genetic algorithm and OR-tools solved the problem of fast deterioration under the large-scale problem. We think this is because those methods lack sufficient numbers of iterations in a reasonable time.

Moreover, we find that as the size of the VRPTW increases, our method becomes better than RL method baselines. This is because when the size of the VRPTW increases, the constraints more seriously disturb the performance of the solver, and without considering the constraints, the result will be much worse.

Table 2 presents the results for the Solomon benchmark. The best known solution is reported by Solomon Dataset [27]. Since this benchmark had insufficient data to train the neural network models, we pre-trained the model on the generated dataset. In addition, we employed beam search to improve our solutions at rollout. From the results, our approach outperformed the RL method.

Runtime. We compared the runtimes of our method to baselines, as shown in Figure 3. Due to the great disparity of methods (ours only used 14 seconds, while OR-tools took about 2 minutes in the VRPTW 50 to calculate 1000 instances), we took the log_2 of

Table 1
Mean distance and CPU times of compared methods. s means seconds

Methods	VRPTW 10, Cap 20		VRPTW 20, Cap 30		VRPTW 50, Cap 40		VRPTW 100, Cap 50	
	Mean	Time	Mean	Time	Mean	Time	Mean	Time
GA	**5.34**	43s	8.76	237s	19.05	1466s	32.61	25681s
NN	6.71	6.83s	10.57	8.83s	19.31	14.43s	29.97	23.72s
OR-tools	5.34	14.61s	8.54	29.18s	17.47	116.85s	28.82	457.77s
RL	5.79	5.38s	9.53	8.96s	17.44	13.88s	26.96	23.28s
ICPO	5.37	5.61s	**8.07**	9.31s	**16.62**	14.18s	**23.51**	24.95s

Table 2
Results for the Solomon benchmark

Methods	Best Known Solution		RL			ICPO			ICPO_BS(5)		
	Veh.	Dist.	Veh.	Dist.	Gaps	Veh.	Dist.	Gaps	Veh.	Dist.	Gaps
C1(25)	3	191.3	3	226.9	18.6	3	197.8	3.4	3	191.3	0
C2(25)	2	214.5	2	245.1	14.3	2	224.1	4.5	2	214.5	0
R1(25)	6	530.5	8	629.6	18.7	6	542.6	2.3	6	530.5	0
R2(25)	3	391.4	4	428.3	9.4	3	403.5	3.1	3	391.4	0
RC1(25)	4	461.1	5	507.4	10	5	489.4	6.1	4	461.1	0
RC2(25)	3	338	5	388.1	14.8	3	358.6	6.1	3	338	0
C1(50)	5	362.4	6	425.9	17.5	5	376.7	3.9	5	362.4	0
C2(50)	3	359.8	4	426.6	18.6	4	381.7	6.1	3	371.3	3.2
R1(50)	12	1044	14	1209.7	15.9	13	1128.4	8.1	13	1103.4	5.7
R2(50)	6	791.9	9	1011.4	27.37	8	892.6	12.7	7	849.7	7.3
RC1(50)	8	994	11	1130.3	19.7	10	1073.6	13.7	9	1057.4	6.4
RC2(50)	5	684.8	7	847.6	23.8	6	768.9	12.3	6	730	6.6
C1(100)	10	827.4	12	1104.1	33.5	11	927.6	12.1	10	871.3	5.3
C2(100)	3	585.8	5	691.8	18	4	633.9	8.2	3	608.7	3.9
R1(100)	18	1466.6	21	1820.1	24.1	20	1742.6	18.8	19	1583.4	7.9
R2(100)	3	1191.7	5	1665.7	39.8	4	1354.2	13.6	4	1288.2	8.1
RC1(100)	14	1457.4	19	1977.9	35.7	16	1609.1	10.4	15	1583.7	8.7
RC2(100)	9	1261.8	12	1625.5	28.8	10	1436.9	13.9	9	1317.4	4.4

Figure 3
Runtime of five methods

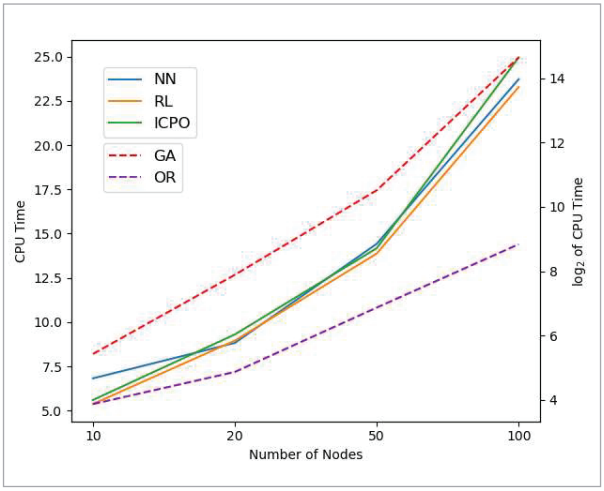
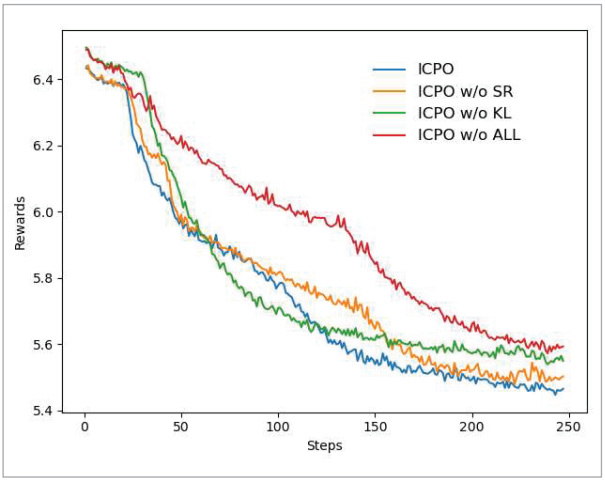


Figure 4
Ablation of ICPO



GA and OR-Tools to make this figure legible. However, the runtimes of these two methods were so long that even in *log* form, the gap was still obvious. Thus, we used two y-axes: the y-axis on the right is for GA and OR-Tools, and that on the left is for the others. We can find that our method is in the middle among all the methods as regards speed (an acceptable running time). Combining Figure 3 and Table 2, we can see that although NN is the fastest, its performance is worst, hence it is hard to use in practice, while ours can maintain a good balance between runtime and solution quality.

Ablations. As shown in Table 1, the complete version of ICPO achieved the highest scores of the three methods. From the ablation presented in Figure 4, we find that ICPO w/o KL is worse than ICPO w/o SR, revealing that constraint awareness plays an important role in getting a good solution, which agrees with our theory. Comparing with original PG, the performance of our method is dramatically better than that of original PG, revealing that ICPO has an advantage in VRPTW.

6. Conclusion

We developed a constraint-awareness RL method to capture the information of constraints to improve performance. Specifically, we changed the VRPTW and the PG method to the DCMDP. To capture the constraints, we designed a constraint-awareness module to reduce the probability of actions against the constraints and enhance performance. For bad behaviors that could decrease the total reward, we leveraged SR as the indicator to diminish the occurrence of those actions. We designed a VRPTW training scheme, and the experiments on the generated datasets and Solomon benchmark revealed that our methods outperform other competition methods.

In the future, we will focus on how to implement the method in practice and consider situations in which agents are competitive.

References

1. Achiam, J., Held, D., Tamar, A., Abbeel, P. Constrained Policy Optimization. Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2017, https://arxiv.org/abs/1705.10528
2. Altman, E. Constrained Markov Decision Processes. CRC Press, 1999, ISBN:978-0-8493-0382-1
3. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S. Neural Combinatorial Optimization with Reinforce-

ment Learning. 5th International Conference on Learning Representations, 2016, https://arxiv.org/abs/1611.09940

4. Berkenkamp, F., Turchetta, M., Schoellig, A., Krause, A. Safe Model-Based Reinforcement Learning with Stability Guarantees. Thirty-first Conference on Neural Information Processing Systems, 2017, https://arxiv.org/abs/1705.08551
5. Chung, J., Gulcehre, C., Cho, K., Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. NIPS 2014 Deep Learning and Representation Learning Workshop, 2014, https://arxiv.org/abs/1412.3555
6. Euchi, J., Zidi, S., Laouamer, L., A Hybrid Approach to Solve the Vehicle Routing Problem with Time Windows and Synchronized Visits In-Home Health Care. Arabian Journal for Science and Engineering, 2020, 45(12), 10637-10652. https://doi.org/10.1007/s13369-020-04828-5
7. Goel, R., Maini, R. A Hybrid of Ant Colony and Firefly Algorithms (HAFA) for Solving Vehicle Routing Problems. Journal of Computational Science, 2018, 2528-37. https://doi.org/10.1016/j.jocs.2017.12.012
8. Gombolay, M. C., Wilcox, R. J., Shah, J. A. Fast Scheduling of Robot Teams Performing Tasks With Temporospatial Constraints. IEEE Transactions on Robotics, 2018, 34(1), 220-239. https://doi.org/10.1109/TRO.2018.2795034 https://doi.org/10.1109/TRO.2018.2795034
9. Google, I., Google's optimization tools (or-tools). 2018, https://github.com/google/or-tools
10. Gromicho, J., Hoorn, J. J. van, Kok, A. L., Schutten, J. M. J. Restricted Dynamic Programming: A Flexible Framework for Solving Realistic VRPs. Computers & Operations Research, 2012, 39(5), 902-909. https://doi.org/10.1016/j.cor.2011.07.002
11. Grzes, M., Kudenko, D. Theoretical and Empirical Analysis of Reward Shaping in Reinforcement Learning. 2009 International Conference on Machine Learning and Applications, 2009. https://doi.org/10.1109/ICMLA.2009.33
12. Hopfield, J. J., Tank, D.W. "Neural" Computation of Decisions in Optimization Problems. Biological Cybernetics, 1985, 52(3), 141-152. https://doi.org/10.1007/BF00339943
13. Khalil, E., Dai, H., Zhang, Y., Dilkina, B., Song, L. Learning Combinatorial Optimization Algorithms Over Graphs. Thirty-first Conference on Neural Information Processing Systems, 2017, https://arxiv.org/abs/1704.01665
14. Kingma, D. P., Ba, J. Adam: A Method for Stochastic Optimization. the 3rd International Conference for Learning Representations, 2015, https://arxiv.org/abs/1412.6980
15. Kool, W., van Hoof, H., Welling, M. Attention, Learn to Solve Routing Problems! Seventh International Conference for Learning Representations, 2019, https://arxiv.org/abs/1803.08475
16. Laporte, G. The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms. European Journal of Operational Research, 1992, 59(3), 345-358. https://doi.org/10.1016/0377-2217(92)90192-C
17. Li, Z., Chen, Q., Koltun, V. Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search. Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, https://arxiv.org/abs/1810.10659
18. Li, J., Han, Y., Duan, P., Han, Y., Niu, B., Li, C., Zheng, Z., Liu, Y. Meta-Heuristic Algorithm for Solving Vehicle Routing Problems with Time Windows and Synchronized Visit Constraints in Prefabricated Systems. Journal of Cleaner Production, 2020, 250119464. https://doi.org/10.1016/j.jclepro.2019.119464
19. Luong, M.-T., Pham, H., Manning, C. D. Effective Approaches to Attention-Based Neural Machine Translation. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, https://doi.org/10.18653/v1/D15-1166
20. Momennejad, I., Russek, E. M., Cheong, J.H., Botvinick, M. M., Daw, N. D., Gershman, S. J. The Successor Representation in Human Reinforcement Learning. Nature Human Behaviour, 2017, 1(9), 680-692. https://doi.org/10.1038/s41562-017-0180-8
21. Nachum, O., Norouzi, M., Xu, K., Schuurmans, D. Bridging the Gap Between Value and Policy Based Reinforcement Learning. Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, https://arxiv.org/abs/1702.08892
22. Nair, V., Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. Proceedings of the 27th International Conference on International Conference on Machine Learning, 2010, https://dl.acm.org/doi/10.5555/3104322.3104425
23. Nazari, M., Oroojlooy, A., Snyder, L., Takác, M. Reinforcement Learning for Solving the Vehicle Routing Problem. Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, https://arxiv.org/abs/1802.04240

24. Petz, D. On the Equality in Jensen's Inequality for Operator Convex Functions. Integral Equations and Operator Theory, 1986, 9(5), 744-747. https://doi.org/10.1007/BF01195811

25. RamachandranPillai, R., Arock, M. An Adaptive Spiking Neural P System for Solving Vehicle Routing Problems. Arabian Journal for Science and Engineering, 2020, 45(4), 2513-2529. https://doi.org/10.1007/s13369-019-04153-6

26. Rousseau, L.-M., Gendreau, M., Pesant, G., Using Constraint-Based Operators to Solve the Vehicle Routing Problem with Time Windows. Journal of Heuristics, 2002, 8(1), 43-58. https://doi.org/10.1023/A:1013661617536

27. Solomon, M. M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research, 1987, 35(2), 254-265. https://doi.org/10.1287/opre.35.2.254

28. Sutton, R. S., Barto, A. G., et al. Reinforcement Learning: An Introduction. MIT press Cambridge, 2018, ISBN:978-0-262-19398-6

29. Tessler, C., Mankowitz, D. J., Mannor, S. Reward Constrained Policy Optimization. Seventh International Conference on Learning Representations, 2018, https://arxiv.org/abs/1805.11074v3

30. Toth, P., Vigo, D. Branch-and-Bound Algorithms for the Capacitated VRP. Society of Industrial and Applied Mathematics, 2002, 29-51. https://doi.org/10.1137/1.9780898718515.ch2

31. Vinyals, O., Fortunato, M., Jaitly, N. Pointer Networks. 29th Conference on Neural Information Processing Systems, 2015, https://arxiv.org/abs/1506.03134

32. Wang, F., Zhou, B., Chen, K., Fan, T., Zhang, X., Li, J., Tian, H., Pan, J. Intervention Aided Reinforcement Learning for Safe and Practical Policy Optimization in Navigation. Conference on Robot Learning, 2018, https://arxiv.org/abs/1811.06187v1

33. Zhang, C., Bütepage, J., Kjellström, H., Mandt, S. Advances in Variational Inference. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019, 41(8), 2008-2026. https://doi.org/10.1109/TPAMI.2018.2889774