Deep Reinforcement Learning Approach to Solve Dynamic Vehicle Routing Problem with Stochastic Customers

Waldy Joe, Hoong Chuin Lau
School of Information Systems
Singapore Management University
waldy.joe.2018@phdcs.smu.edu.sg, hclau@smu.edu.sg

Abstract

In real-world urban logistics operations, changes to the routes and tasks occur in response to dynamic events. To ensure customers' demands are met, planners need to make these changes quickly (sometimes instantaneously). This paper proposes the formulation of a dynamic vehicle routing problem with time windows and both known and stochastic customers as a route-based Markov Decision Process. We propose a solution approach that combines Deep Reinforcement Learning (specifically neural networks-based Temporal-Difference learning with experience replay) to approximate the value function and a routing heuristic based on Simulated Annealing, called DRLSA. Our approach enables optimized re-routing decision to be generated almost instantaneously. Furthermore, to exploit the structure of this problem, we propose a state representation based on the total cost of the remaining routes of the vehicles. We show that the cost of the remaining routes of vehicles can serve as proxy to the sequence of the routes and time window requirements. DRLSA is evaluated against the commonly used Approximate Value Iteration (AVI) and Multiple Scenario Approach (MSA). Our experiment results show that DRLSA can achieve on average, 10% improvement over myopic, outperforming AVI and MSA even with small training episodes on problems with degree of dynamism above 0.5.

Introduction

We consider a real-time planning problem in urban logistics in which changes to the routes and tasks are required in response to dynamic events. For example, customers may add or cancel requests throughout the day; travel or service times may change drastically due to traffic congestion; order amounts or demands may need to be modified. To ensure customers' demands are met, logistics service providers need to dynamically respond to these changes quickly.

More precisely, we consider a Dynamic Vehicle Routing Problem (DVRP) with time windows and both known (i.e. fixed) and stochastic customers. We model this problem as a route-based Markov Decision Process (MDP), and to solve the problem efficiently, we propose an approach that combines Deep Reinforcement Learning (RL) (specifically neural networks-based Temporal-Difference (TD) learning with experience replay) to approximate the value function and a routing heuristic based on Simulated Annealing (SA), called DRLSA. Furthermore, to exploit the structure of this problem, we propose a state representation based on the total cost of the remaining routes of the vehicles and the current time at the point of decision-making. We show that the cost of the remaining routes of vehicles can serve as proxy to the sequence of the routes and time window requirements. This state representation captures both spatial and temporal features which impact decision-making.

Given our approach, an optimized re-routing decision can be produced almost instantaneously for moderately large problem instances based on small training episodes in the magnitude of thousands. This is contrasted with sampling-based online approaches which reportedly take the order of 100 seconds per decision (Voccia, Campbell, and Thomas 2017) and most MDP-based approaches that adopted Value Function Approximation (VFA) method only managed to solve single vehicle problems without any additional constraints like time windows or capacity and required training episodes in the magnitude of millions (Ulmer, Mattfeld, and Köster 2017; Ulmer, Thomas, and Mattfeld 2018). Our approach is also oblivious to probability distributions of demand uncertainty unlike stochastic optimization methods.

We compare experimentally the performance of DRLSA with Approximate Value Iteration (AVI) (Ulmer, Thomas, and Mattfeld 2018) and Multiple Scenario Approach (MSA) (Bent and Van Hentenryck 2004b), and show that DRLSA outperforms both methods in problem settings where dynamic requests take up more than 50% of the total requests.

This paper makes the following contributions:

- We formulate our problem as a route-based MDP and propose a state representation based on the total cost of the remaining routes of the vehicles and the current time at the point of decision-making.
- We propose a solution approach that combines neural networks-based TD learning with experience replay to approximate value function and a routing heuristic based on SA (called DRLSA) to solve the problem.
- We show experimentally that our approach outperforms existing methods when dynamic requests are prevalent.

Related Works

Research on DVRPs witnessed a surge in the recent decade (see survey by Psaraftis, Wen, and Kontovas (2016)). This trend is likely to continue given increasing needs toward services like same-day delivery.

DVRP can be broadly categorized into dynamic-deterministic and dynamic-stochastic (Pillac et al. 2013). Even within dynamic-stochastic VRP (DSVRP), stochasticity arises in different aspects namely travel times, demands, customers or combinations of these stochastic aspects (Ritzinger, Puchinger, and Hartl 2016). Research focus in the recent decade has been on DSVRP as it models more closely the real-world environment. This paper focuses on solving DVRP with both known and stochastic customers.

DVRP can be modelled as MDP because in DVRP, decisions need to be made in view of uncertainty and are done sequentially. Conventional MDP identifies optimal decision such as next customer to visit at every decision point. However, this does not address our need, as we need to determine not only who the next customer is (which may be sufficient for ride-sharing problems), but what the updated route plan is (as goods must be loaded onto vehicles prior to delivery, which typically should not be undone arising from future requests). Ulmer et al. (2017) proposed route-based MDP as a common modeling framework for DVRP. In route-based MDP, the state and action space include updates to the routes. This paper adopts the route-based MDP as the model.

There are 3 broad categories of approaches for solving DSVRP:

1. **Offline or Pre-processed decision support.** Policies or values for decision-making are computed prior to execution of plan. Here, the solution approaches are mainly MDP-based. Unfortunately, MDP-based approaches fall into the curse of dimensionality and hence are not suitable for most real-world problems (Pillac et al. 2013). Approximate Dynamic Programming (ADP) approaches are commonly used to tackle the scalability issue, and one such ADP method for DVRP is Approximate Value Iteration (AVI). Ulmer, Thomas, and Mattfeld (2018) and Agussurja, Cheng, and Lau (2019) proposed AVI to solve DVRP as an MDP. Both papers proposed state aggregation and representation to further overcome the challenge of large state space. Another main challenge of ADP is to generate enough scenarios during the training phase so as to accurately assign a value to a state. AVI approximates the value function as a lookup table and may fail if certain state is not encountered during the training phase. Most works used AVI on single vehicle setting without time window constraints which may not extend well for more complex problems like those with multiple vehicles and additional constraints such as time windows or capacity. VFA via non-parametric models such as neural networks is a popular choice for more complex problems. This paper proposes a neural-network based VFA.

   We also like to point out there has been increasing research on machine learning methods (such as sequence-to-sequence models and deep RL) for solving classical combinatorial optimization problems (Vinyals, Fortunato, and Jaitly 2015; Bello et al. 2016; Nazari et al. 2018). The main idea behind is to let algorithms learn their own heuristics rather than depending on hand-crafted heuristics. These methods had been applied to pure DVRP where initial routes are not existent and routes are constructed from scratch. These methods may not work on partially dynamic problems like ours where initial routes are already available and re-routings are done dynamically with minimal disruptions to the existing routes for practical reasons.

2. **Online decision.** Unlike the pre-processed decision support approaches, online decisions do not compute optimal global policy; rather computations are performed during the execution of plan. Common approaches in this category are usually termed as lookahead strategy or rolling horizon procedures (Powell 2011) such as rollout algorithms (Bertsekas, Tsitsiklis, and Wu 1997; Secomandi 2001; Goodson, Thomas, and Ohlmann 2017) and Multiple Scenario Approach (MSA) (Bent and Van Hentenryck 2004b; Voccia, Campbell, and Thomas 2017). They are mainly sampling-based approaches. The common feature of these methods is that they focus on the current state and instance. They do not consider the values of all possible states but only the relevant states at the decision point. At the decision point, the methods do 'roll-out' or 'lookahead' to simulate what will happen in the future and use this information to guide decision-making. Bent and Van Hentenryck (2004a) introduced the consensus algorithm into MSA to solve online VRPs with stochastic customers. Consensus algorithm performs offline optimization on the available and sampled requests once per scenario and returns the decision with the largest score or lowest cost. Online decisions approaches are suitable where there is no strict time constraint imposed when decision-making is required and work well with increasing degree of dynamism (DoD). Thus, the choice between online or offline methods will be very much dependent on how fast decisions need to be made during execution time.

3. **Hybrid approaches.** There have been attempts to combine both approaches to leverage the strengths of both approaches. Ulmer et al. (2018) proposes offline-online approximate dynamic programming which embeds the offline VFA into the online roll-out algorithm. Ulmer et al. (2018) only managed to run at most 16 'lookahead' samples for every decision instance due to resource constraint. The gain achieved through this hybrid approach may not be compelling enough given the vast compromise in terms of decision-making time. De Filippo, Lombardi, and Milano (2018) proposed an integration of offline and online optimization by considering multi-stage optimization problems where the first phase requires offline decision and the subsequent phases require online decision. The proposed approach works on DVRP with stochastic travel time where customers are assigned offline but routes are optimized online. This, however, does not work for DVRP with stochastic customers as customer requests are not known beforehand.

Problem Description and Model

Problem Description

We are given a fleet of M identical vehicles initially located at the depot at the start of the day. Each vehicle has an initial route $\beta_m(0)$ consisting of a sequence of customer orders, C_0 to fulfill for that particular day. Every route starts and end at the depot. Every order has a delivery time window $[e_n, l_n]$. In addition, there is a lunch hour when no delivery can be made. A waiting time is incurred if the vehicle either arrives early or during the lunch hour. Throughout the day, new orders arrive from C_r. An action/decision x_k is selected to modify a route assigned with the new order(s). Delivery later than the time window upper bound incurs a penalty cost per unit time violated. The objective is to minimize the sum of total travel and waiting times of all vehicles and penalty cost for time window violations. In this paper, we focus on insertion of new orders rather than cancellation since both warrant similar approach and insertion is more challenging and interesting than cancellation.

Model Formulation

We model this dynamic multi-vehicle same-day delivery routing problem with time windows and stochastic customers as a route-based MDP. Table 1 provides the set of notations and the corresponding descriptions used in the model.

Why Route-based MDP? Ulmer et al. (2017) first proposed route-based MDP as a unifying modelling framework for VRP where the action is not merely the next customer to visit but the remaining route to be assigned to a vehicle. Unlike the passenger ride-sharing problem where single-action (myopic) MDP makes more sense with somewhat simple consideration of passengers' pickup and drop off locations, logistics suffers much more restrictions and complications.

Firstly, the planning horizon is much longer (e.g. entire day) and the delivery time windows and meal breaks must be considered, so the decision of which vehicle and sequence to serve a given request must anticipate future dynamic requests. This implies that the value of a state needs to take into account the projected route of the vehicle in anticipation of future new requests. The calculations of rewards take into account the penalty and waiting time that may occur into the future and these calculations can only be derived if a route is available.

Secondly, the problem setting we address is a real-world Business-to-Business (B2B) delivery operation. The visibility of entire routes is important to drivers from the point of view of execution planning, cargo loading, tracking and communication on the ground. Each delivery job is tied to its respective cargo that needs to be loaded to the assigned vehicle, and swapping cargo between vehicles is not viable operationally. Note however that in this paper, in order to simplify discussion, we assume that a new request can be served without considering its pickup. The reader may deduce that incorporating pickup is fairly straightforward, since we are making route changes within a single vehicle.

Table 1: Set of Notations used in the MDP model.

Notation	Description
M	Set of identical vehicles, $M \in \{1, ..., M\}$
	Depot is set as location 0
$\delta_m(k)$	A route or sequence of remaining locations to visit by vehicle m at decision epoch k
$\delta^x_m(k)$	A route or sequence of remaining locations to visit by vehicle m after executing decision x at decision epoch k
$\beta_m(k)$	A route or sequence of locations to visit and visited by vehicle m updated as of decision epoch k
$\beta^x_m(k)$	A route or sequence of locations to visit and visited by vehicle m after executing decision x at decision epoch k
$vloc_m(k)$	Location of vehicle m at decision epoch k
$t(k)$	Time at decision epoch k
C_k	Set of realized orders updated as of decision epoch k
C_r	Set of stochastic orders
$CS_n(k)$	Order status of customer order n at decision epoch k
$[e_n, l_n]$	Delivery time window at customer location for order n
$\tau(\delta_m(k))$	Total travel time of vehicle m when following route $\delta_m(k)$
$\tau(\beta_m(k))$	Total travel time of vehicle m when following route $\beta_m(k)$
$wait(\delta_m(k))$	Total waiting time of vehicle m when following route $\delta_m(k)$
$wait(\beta_m(k))$	Total waiting time of vehicle m when following route $\beta_m(k)$
$pen(\delta_m(k))$	Total penalty cost for time windows violation of vehicle m when following route $\delta_m(k)$
$pen(\beta_m(k))$	Total penalty cost for time windows violation of vehicle m when following route $\beta_m(k)$

Decision Epoch. A decision epoch or decision point k occurs at every time step $t(k)$. This means that dynamic event can take place at any time throughout the time horizon. Thus, by this definition, k is equal to $t(k)$. This definition of decision epoch is chosen to simulate real-world environment where dynamic events can take place at any time and also to facilitate modelling the arrival rate of these events.

State. A state of the MDP consists of two parts, pre-decision state S_k and post-decision state S^x_k. S_k captures the necessary information required such as the current time, locations of all the vehicles, the remaining routes of all the vehicles and the statuses of all the realized orders. S_k is represented as the following tuple:

$S_k = \langle t(k), vloc(k), \delta(k), CS(k)\rangle$

where $vloc(k) = (vloc_m(k))_{m \in M}, \delta(k) = (\delta_m(k))_{m \in M}$ and $CS(k) = (CS_n(k))_{n \in C_k}$. The post-decision state S^x_k captures the changes to the state upon executing a decision. $t(k)$ remains the same while the other three components are updated depending on the decision taken.

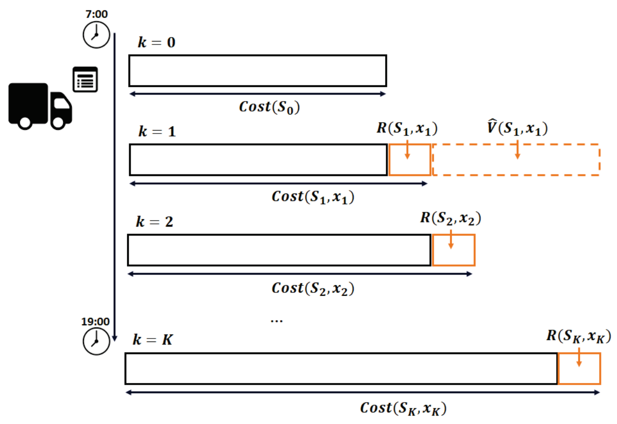

Figure 1: Illustration on how the reward function is derived.

Action/Decision. x_k at decision epoch k is the action of updating the remaining route of the vehicle which is assigned to serve the new order. The route of the chosen vehicle m, $\delta_m(k)$ is revised to $\delta^x_m(k)$ after executing $x_k \in X(S_k)$ where all orders should be delivered within $[e_n, l_n]$. Note that the time window is a soft constraint while lunch hour period is a hard constraint. We also assume that there is no swapping of customer orders among vehicles as the load picked up for a particular customer order must be delivered to the respective customer.

Transition. There are two main transitions in the model namely, from pre-decision state, S_k to post-decision S^x_k and from S^x_k to the next pre-decision state, S_{k+1}. The transition from S_k to S^x_k has been mentioned in earlier. Meanwhile, during transition from S^x_k to S_{k+1}, a realization of new order, ω takes place, and C_{k+1} is updated by adding the new order to C_k and $S_{k+1} = (S^x_k, \omega)$ where $\omega \in C_r$.

Reward Function. The "reward" function, $R(S_k, x_k)$ is defined as the incremental increase in total *cost* of being in state S_k and choosing decision x. $Cost(S_k, x)$ is defined as the total cost (i.e. travel plus wait plus time window violation) when choosing decision x at state S_k.

$$Cost(S_k, x) =$$

$$\sum_{m=1}^{M} \tau(\beta^x_m(k)) + wait(\beta^x_m(k)) + pen(\beta^x_m(k))$$

$$R(S_k, x_k) = Cost(S_k, x_k) - Cost(S_{k-1}, x_{k-1})$$

Figure 1 illustrates how this reward function is derived.

Value Function. The value function equation, $V(S_k)$ can be approximated to the following Bellman Equation:

$$\hat{V}(S_k) = min_{x \in X(S_k)}\{R(S_k, x) + \gamma \hat{V}(S^x_k)\}$$

The goal is to minimize the expected future cost over the planning horizon. In other words, given a state S_k, select decision x that returns the minimum $\hat{V}(S_k)$. As shown in Figure 1, the approximated value function (for e.g. $\hat{V}(S_1, x_1)$) would take into consideration expected the future rewards from yet-to-realized dynamic events.

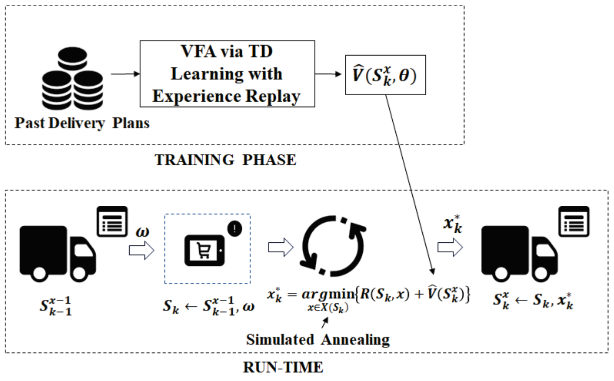
Past Delivery Plans
VFA via TD Learning with Experience Replay
$\hat{V}(S^x_k, \theta)$
TRAINING PHASE
ω
x^*_k
S^x_{k-1}
$S_k \leftarrow S^{x-1}_{k-1}, \omega$
$x^*_k = \underset{x \in X(S_k)}{argmin}\{R(S_k, x) + \hat{V}(S^x_k)\}$
$S^x_k \leftarrow S_k, x^*_k$
Simulated Annealing
RUN-TIME

Figure 2: Framework of the proposed approach, DRLSA.

Solution Approach

Our solution approach is adapted from Ulmer, Thomas, and Mattfeld (2018) but with major modification given the differences in problem settings. Figure 2 shows that our proposed approach, DRLSA, comprises two phases namely training phase and run-time. We propose neural networks-based TD learning with experience replay to approximate the value function which is subsequently utilized by a Simulated Annealing (SA) algorithm to generate the revised routes during run-time. Due to large state space, a state representation based on the cost of the remaining routes of vehicles and current time is used to capture the spatial and temporal attributes of the state.

VFA via TD Learning with Experience Replay

During the training phase, we use a set of historical delivery plans to simulate the initial delivery plans and, depending on the value of DoD, a percentage of the orders in the plan is randomly removed and these orders are added subsequently as realizations of new orders. This is done to simulate dynamic events. This offline simulation outputs the approximated value function for each post-decision state, $\hat{V}(S^x_k, \theta)$ in a form of a non-parametric function or neural networks. The bulk of the computational time of this approach is during this training phase; and during execution, the run time is spent on SA to generate the changes to the routes.

Algorithm 1 describes how our proposed neural networks-based TD learning with experience replay algorithm approximates the value function. This algorithm is adapted from the vanilla version of Deep Q-Network (DQN) with experience replay (Mnih et al. 2013) with $V(S^x, \theta)$ replacing $Q(S, x, \theta)$ at lines 11, 16, 20 and 21 since we are approximating the value function instead of the state-action value or Q-function. Similar to DQN, two value function networks are used to deal with non-stationarity of the target network and experience replay to ensure that randomly selected samples are independent. However, another difference is that this proposed algorithm is on-policy while DQN is off-policy (see difference in line 16). Mini-batch gradient descent is used to update the parameter of target network as shown in line 20. In addition, line 11 includes SA operator to choose the optimal re-routing action instead of computing the 'true'

Algorithm 1: VFA via TD Learning with Experience Replay

```
Input  : No. Simulation Runs $N$, Replay Memory $D$, Initial Value Function $V$ with random
         weights $\theta$, Initial Target Value Function $\hat{V}$ with random weights $\theta^{-}$
Output: $\theta$
$i = 1$
while $i \leq N$ do
    Initialise a typical day scenario of delivery plan with stochastic orders
    Initialise $S_0$
    $k = 1$
    while $S_k \neq S_K$ do
        if new order = True then
            $S_k \leftarrow (S^x_{k-1}, \omega)$
            With probability $\epsilon$ select a random decision $x_k$
            Otherwise
            // use SA to find the optimal feasible decision $x_k$
            $x_k \leftarrow \arg\min_{x \in X(S_k)}\{R(S_k, x) + \gamma V(S^x_k, \theta)\}$
            // proceed with the updated route
            $S^x_k \leftarrow (S_k, x_k)$
            Store transition $(S_k, S^x_k, R(S_k, x_k))$ in $D$
            Sample random minibatch of transitions $(S_j, S^x_j, R(S_j, x_j))$ from $D$
            if $S^x_k \neq S_K$ then
                $y_j \leftarrow R(S_j, x_j) + \gamma \hat{V}(S^{x_j}_j, \theta^{-})$
            else
                $y_j \leftarrow R(S_j, x_j)$
            end
            Perform a gradient descent on $(y_j - V(S^{x_{j-1}}_{j-1}, \theta))^2$ with respect to parameter $\theta$
            Reset $\hat{V} = V$ for every $C$ steps
        else
            // proceed with the existing route
            $S^x_k \leftarrow S_k$
        end
        $k \leftarrow k + 1$
    end
    $i \leftarrow i + 1$
end
return $\theta$
```

$\arg\min$. SA explores the search space via 2-opt local search probabilistically. Routing heuristic is used here since computing the 'true' $\arg\min$ involves brute force enumerating of all possible re-routing sequences.

Why Deep RL? We propose an RL-based approach because it is model-free and labelled data are not available. Furthermore, deep RL-based method allows the neural networks to approximate the value function and tune the corresponding parameters based on the observed rewards over many training episodes. There is no need to make assumption on the underlying relationships between the features in the states. The approximated value function also takes into account the underlying probability distribution of the occurrences of dynamic event without the need to explicitly specify the probability distribution of the new orders unlike stochastic optimizations methods. Compared to lookup-based ones like AVI, neural networks-based VFA is also able to handle larger state space.

Why Approximate Value Function? We chose an algorithm that approximates the value function instead of Q-function. This is deliberately chosen to fit the context of our problem. In DSVRP, a decision involves a re-routing of the remaining routes which does NOT consist of an explicit action per se. As shown in Figure 3, the decision in response to new customer 6 is not merely choosing the position for insertion but it can also include swapping of existing customers' positions. This means that the action space is exponentially large with respect to number of customers, since every scenario consists of different possible actions due to different possible lengths of routes and different possible customer locations. Zhang and Dietterich (2000) also concluded that Q-function is not suitable in problem settings where the actions are largely dependent on the current states like job scheduling problems. Specifying the Q-value of each action is not possible since action space changes depending on the current state. This also means that off-policy TD learning method like Q-learning may not be applicable directly. Thus, approximating the value of a state using on-policy TD learning is more suitable for this problem setting.

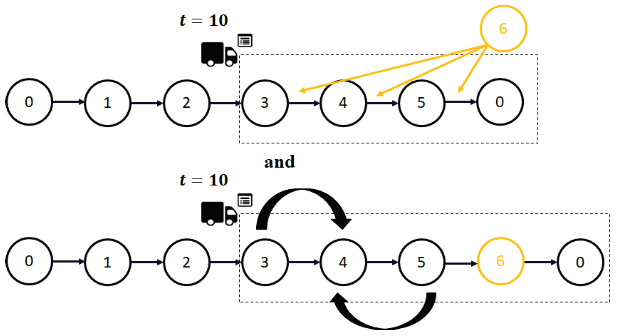

Figure 3: A re-routing decision is more than insertion of new customer and may include swapping of existing customers.

Routing Optimization via SA

During the run-time phase, a set of scenarios of daily delivery plans and dynamic realizations of stochastic new orders are presented. Our approach utilizes the approximate value function from the training phase to compute value of each decision explored during the SA search. As shown in Figure 2, x^*_k is modified using SA taking into account both immediate cost plus expected future reward computed by the approximate value function, $\hat{V}(S^x_k)$. We note that many routing heuristics can be used to optimize the routes. In this paper, we picked SA due to its efficiency and effectiveness to provide fast quality solution for vehicle routing problems (Chiang and Russell 1996; Osman 1993). The focus of the paper is not to find the best heuristics for routing, but a flexible solution framework which enables different heuristics to be used in optimizing the routes.

State Representation

Due to the large state space, the post-decision states are usually aggregated or represented based on certain handcrafted features. It is important to find features that exploit the structure of the problem and are able to sufficiently differentiate between two distinct states which may require two different policies. Thus, we propose a state representation based on the current time and the cost of the remaining routes of the vehicles at decision epoch k which includes the penalty cost for time window violations for each vehicle. The proposed state representation is shown below:
$S'^x_k = \langle t(k), Cost_{remain}(S_k, x), Cost_{penalty}(S_k, x)\rangle$ where $Cost_{remain}(S_k, x) = (Cost_{remain,m}(S_x, k))_{m \in M}$, $Cost_{(remain,m)}(S_k, x) = \tau(\delta^x_m(k)) + wait(\delta^x_m(k))$, $Cost_{penalty}(S_k, x) = (Cost_{penalty,m}(S_k, x))_{m \in M}$, $Cost_{(penalty,m)}(S_k, x) = pen(\delta^x_m(k))$

Algorithm 2: Approximate Value Iteration

```
   Input  : Initial Values $\hat{V}_0$, No. Simulation Runs $N$, Step Size $\alpha$
   Output: Values $\hat{V}_N$
1  $i = 1$
2  while $i \leq N$ do
3      $k = 1$
4      while $S_k \neq S_K$ do
5          if new order = True then
6              $S_k \leftarrow (S^x_{k-1}, \omega)$
               // use SA to find the optimal feasible decision $x_k$
7              $x_k \leftarrow \arg\min_{x \in X(S_k)} \{R(S_k, x) + \gamma \hat{V}_{i-1}(S^x_k)\}$
               // proceed with the updated route
8          else
               // proceed with the existing route
9              $S^x_k \leftarrow S_k$
10         end
11         $S^x \leftarrow S^x \cup \{S^x_k\}$
12         $R_k \leftarrow R_{k-1} + R(S_k, x)$
13         $k \leftarrow k + 1$
14     end
       // Update the Value Lookup Table
15     forall $S^x_k \in S^*$ do
16         $\hat{V}_i(S^x_k) \leftarrow (1-\alpha)\hat{V}_{i-1}(S^x_k) + \alpha(R_K - R_k)$
17     end
18     $i \leftarrow i + 1$
19 end
20 return $\hat{V}_N$
```

Time. $t(k)$ is an important temporal feature in the problem because the earlier the decision point during run-time, the more likely there will be new stochastic requests in the future and thus, the anticipatory value for future rewards is higher. The same principle applies when $t(k)$ happens later during run-time.

Cost of Remaining Routes. The re-routing decision to the existing route does not depend on the previous visited locations prior to decision point k. This is because the immediate and future rewards only depend on the remaining routes.

Specifying the exact remaining routes of the vehicles, $\delta(k)$ may result in a very large state space. The proposed state representation uses the cost of remaining routes, $Cost_{remain}(S_k, x)$ as a proxy to for the routes. For example, the cost of route $[3, 4, 5, 0]$ is different from $[3, 5, 4, 0]$ and this sufficiently differentiate the routes. There may be scenarios where the cost of two routes may be the same even though the exact sequences of the routes are different. However, this is not a concern since re-routing decision is dependent on the immediate reward and the post-decision state which are different in both scenarios. For example, inserting Customer 6 at the second position (i.e. $[3, \mathbf{6}, 4, 5, 0]$ and $[3, \mathbf{6}, 5, 4, 0]$ respectively) will result in different immediate rewards and post-decision states in both scenarios.

Penalty Cost. State representation also needs to capture the time window requirements. Assuming two different new orders with the same location but different time windows requirements, inserting these two orders at the same exact position in a route may result in different reward and post-decision state. This is because the penalty costs for the time window violations are different. Thus, the total penalty cost can serve as a proxy for the time window requirement.

Note that the state representation does not need to explicitly specify the exact route as routing optimization is handled by SA during run-time.

Experiments

The objective of the experiment is to evaluate the performance of DRLSA against other existing algorithms, both offline and online. We evaluate these approaches based on how much improvement they achieve over the pure re-optimization method i.e. myopic approach.

Benchmark Algorithms

Approximate Value Iteration. Algorithm 2 shows the AVI algorithm adapted to the problem setting of this paper. Cumulative observed rewards are used to approximate the future expected rewards of a state (lines 12 and 16). We use SA as routing heuristics to compute the $\arg\min$ in line 7 and same state representation for a fairer comparison.

Multiple Scenario Approach. MSA with consensus algorithm used in this experiment is adapted from Bent and Van Hentenryck (2004a; 2004b). MSA does not have training phase and is directly applied during the run-time. The main idea of this algorithm is to find the optimal route at every decision point by sampling lookahead scenarios and compute the decision that returns the lowest average total cost across the samples.

Algorithm 3 details how this algorithm is applied during the run-time. At every decision point k, J samples of future new requests are collected (line 8). For every sample, an optimal route is calculated assuming that all the future new requests until $k + H$ are known at k (line 9). We also use SA as the routing heuristic to compute the optimal route. The resulting optimal route from each sample is stored with the sampled future new orders removed (line 10). This will represent one possible re-routing decision. Across many samples, unique re-routing decisions are stored and the average total cost is calculated for every unique decision or route (line 11). Route with the lowest average total cost will be selected as the best decision (line 14).

Myopic Approach. This approach is simply choosing a decision that gives the minimal immediate total rewards.

Experiment Design

We use 2-month's worth of historical delivery data from a local logistics service provider containing 48 customer locations. The data contains the daily delivery plans generated by an optimization algorithm that minimizes travel time, wait time, make span (the amount of time the vehicle is out during the delivery) and penalty cost due to time window violations. The delivery data is split into training (34 days) and test sets (10 days). 2 vehicles are used for this experiment with an average of 22 daily orders.

For every test scenario, a random daily delivery plan from the test set is picked as the initial routes of the vehicles. Depending on DoD, a percentage of the orders in the initial plan is randomly removed and are added subsequently during the simulation; following a Poisson process. The spatial distribution of customers requesting these new orders are based on probability distribution derived from the historical data. We need to specify this distribution in running MSA

Algorithm 3: Multiple Scenario Approach with Consensus Algorithm

```
Input  : Initial State $S_0$, Horizon $H$, Sample Size $J$
Output: Total Cost $Cost(S_K)$, Final State $S_K$
$k = 1$;
while $S_k \neq S_K$ do
    if new order = True then
        $S_k \leftarrow (S^x_{k-1}, \omega)$
        $j = 1$
        // initialize an empty list to store sampled optimal
           routes
        $X^* = [\ ]$
        // find the optimal feasible decision $x^*$ through
           sampling 'lookahead' scenarios
        while $j \leq J$ do
            // sample a set of future new orders (until time
               $k+H$) based on current state, $S_k$
            $\Omega_j \leftarrow SAMPLE(S_k, H)$
            // use SA to optimize the route assuming all sampled
               new orders are known beforehand
            $x^*_j \leftarrow OPTIMIZE(S_k, \Omega_j)$
            // extract the optimal route without the sampled new
               orders
            $x^*_j \leftarrow x^*_j \setminus \Omega_j$
            // add the sampled optimal routes to a list and keep
               account of the counts and average cost of each
               unique route
            $X^* \cup x^*_j$
            $j = j + 1$
        end
        // choose the optimal route which returns the lowest
           average total cost
        $x^* \leftarrow \arg\min_{x^*_j \in X^*} Cost(S_k, x^*_j)$
        // proceed with the updated route
        $S^x_k \leftarrow (S_k, x^*)$
    else
        // proceed with the existing route
        $S^x_k \leftarrow S_k$
    end
    $k \leftarrow k + 1$
end
return $Cost(S_K), S_K$
```

and not for DRLSA or AVI as these two methods will learn the underlying distribution during training.

We use the percentage improvement in terms of the final total cost that the approaches (DRLSA or AVI or MSA) can achieve over myopic as the performance measure.
$\%improve_{DRLSA} = \frac{Cost_{DRLSA}(S_K) - Cost_{myopic}(S_K)}{Cost_{myopic}(S_K)} \times 100\%$

By this definition, the tested approaches must achieve a negative % improvement if they are to result in lower cost. For simplicity's sake and consistency with other works in the literature, we report reduction in cost as positive value and increase in cost as negative value.

Experiment Setup. The experiment consists of 3 phases. The 3 phases of experiment are as follow:

- **DRLSA vs. AVI.** We use 3 different values of DoD, $0.3, 0.5$ and 0.7. DoD refers to the percentage of total orders that are dynamic. For each DoD, we ran 20 experiment runs. Each experiment run consists of 50 daily test scenarios.
- **DRLSA vs. MSA.** We use only $DoD = 0.7$ with 100 test scenarios and sample size, $J = 50$ and 100.
- **Further Experimentations on DRLSA.** We test DRLSA with larger DoDs and larger problem scale.

Model and Training Setup. DRLSA is a fully connected neural network with 2 hidden layers with 64 nodes and 32 nodes respectively. We experimented on various ranges of parameters such as number of hidden layers, nodes, batch size, update and learn frequencies and learning rate. We empirically evaluated and found that the following parameters resulted in the best performance in terms of average total cost: batch size $= 20$, learn and update frequencies of once in every 5 steps, learning rate $= 0.001$ and $\gamma = 0.99$. The value of ϵ in the ϵ-greedy step is set to be decreasing as the number of training episode increases. Stochastic Gradient Descent is used as the optimizer. For AVI, we use a constant value of $\alpha = 0.1$. For MSA, we set H as large number to simulate that all future requests are known. SA is the default routing heuristic with the following dynamic cooling function, $T = T_{max} \times e^{-f(\frac{step_{current}}{step_{max}})}$ where $step_{max} = 2500$ and $T_{max} = 25000$.

VFA via non-parametric function has an advantage over the table-based ones because it may not need to observe all possible states and approximate the function based on the observed ones. We observe that for DRLSA, the average total cost of past 100 episodes begins to drop and stabilizes after 2500 training episodes. Thus, we set the number of training episodes for DRLSA to be 3500 and 35000 for AVI. We show that even with less training episodes, DRLSA is able to outperform AVI.

Assumptions. There are several assumptions used in modelling the DVRP environment. There is no capacity constraint, no new order after certain designated time and no swapping of loads among vehicles once assigned. New orders can be loaded from another customer locations so no depot returns are required. Vehicle which is on the way to the next customer order needs to fulfill the order first. Travel and service times are assumed to be static.

Table 2: Average % of improvement of DRLSA vs. AVI over different DoDs.

	Method	**0.3**	**0.5**	**0.7**
Average% of improvement over 20 experiment runs	DRLSA	-1.95%	-0.98%	11.90%
	AVI	0.24%	-1.25%	-1.51%

Experiment Results

DRLSA vs. AVI. Table 2 provides the summary of the phase one results. The proposed approach outperforms AVI for $DoD = 0.7$, achieving on average 12% improvement over myopic. Figure 4 shows the average performances of DRLSA and AVI over myopic over 20 experiment runs. Although the training time of DRLSA is 4 times longer than AVI (DRLSA takes about 36 secs/scenario compared to AVI which takes about 8 secs), the computation time during runtime is almost instantaneous (< 10 secs/decision) for both.

We observe that AVI performs quite poorly for every DoD. This can be due to the fact that the training episodes may not be sufficient for the algorithm to approximate most or all possible state representations. AVI approximates value of states that have never been encountered during training as 0 and this is equivalent to myopic approach. It is no surprise that AVI results in near 0% improvement. DRLSA does not

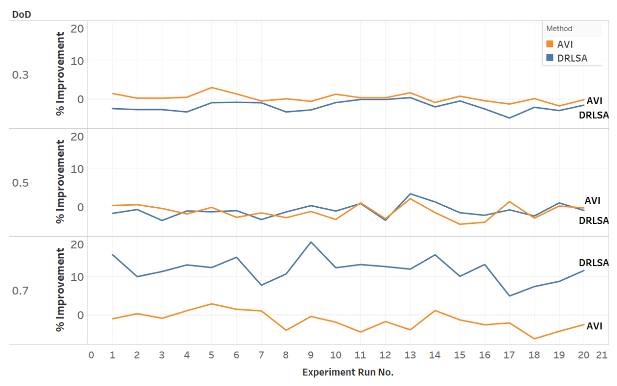

Figure 4: Average % of improvement of DRLSA vs. AVI over different experiment runs.

Table 3: Average % of improvement and computation time per scenario for DRLSA vs. MSA.

	Average% of Improvement over 100 Test Scenarios	**Average Computation Time per Scenario (in mins)**
DRLSA	9.61%	< 1
MSA ($J = 50$)	1.73%	3
MSA ($J = 100$)	2.20%	10

perform better than myopic when $DoD \leq 0.5$. This is expected as lower DoD means new stochastic orders are less likely to occur and anticipating a rarely occurring event becomes more challenging (which perhaps cannot be learnt). Nevertheless, this experiment is able to show that DRLSA is able to achieve good result even with small training episodes and with higher DoDs.

DRLSA vs. MSA. DRLSA achieves an average of 9.6% improvement over 100 test scenarios, outperforming MSA even with $J = 100$ and it is more than 10 times faster in terms of computation time (see Table 3). MSA needs a very large sample size in order to evaluate the re-routing decisions. To sample the instances of future new requests in the next time horizon H, it needs to take into consideration a combination of the locations of those requests, the arrival time of the requests and the corresponding time window requirements. Therefore, the performance of online approach like MSA depends on how good the stochastic model is in simulating the future events. On the other hand, offline, learning-based approach such as DRLSA is able to learn the underlying probability distribution of the dynamic events based on historical data.

To our knowledge, there is no study in the literature that compares performance between offline and online decisions approaches and no set of benchmark test parameters available. Nevertheless, based on our experiment, to outperform DRLSA, MSA needs $J >> 100$ and even longer computation time.

Table 4: Performances of DRLSA for increasing DoDs.

	Method	**0.6**	**0.7**	**0.8**
Average% of improvement over 20 experiment runs	DRLSA	1.92%	11.90%	12.72%

Further Experimentation on DRLSA. We evaluate the performance of DRLSA over higher DoDs and are able to show that DRLSA indeed performs better with increasing DoDs (see Table 4).

To evaluate its scalability, we did further experiments with 3 and 4 vehicles with average total daily orders of 30 and 40 respectively, The available datasets only allow us to scale up to 4 vehicles with 40 total daily orders. Based on 20 experiment runs, DRLSA achieves an average of 15.71% and 20.88% improvements over myopic respectively.

Experiment Discussion

Ritzinger, Puchinger, and Hartl (2016) summarized the performances of various successful pre-processed decision methods in the literature. For larger-sized customers (> 30), most approaches managed to achieve improvements in the region of $5\% - 10\%$ over myopic. Our experiment shows that the performance of DRLSA is comparable with those cited in that study. However, we note that the experiment setups across the papers may be different.

This experiment also shows that our proposed approach, DRSLA is able to outperform both AVI and MSA even with a relatively small number of training episodes. AVI's poor performance also reiterates our hypothesis that AVI may not work well in complex problems with large state spaces (due to multi-vehicle and multi-constraints settings) even with state aggregation as a very large number of training episodes (in the magnitude of millions) are required to reasonably approximate the value function of most of the states. The experiment also shows that offline approach like DRLSA is more suited than the online counterparts like MSA for problem setting that requires instantaneous decision-making such as same-day delivery operations.

Conclusion and Future Works

We propose in this paper an efficient and effective approach to cope with real-time decision making in dynamic vehicle routing. We believe our approach is fairly generic in that it can be applied to tackle other dynamic planning and scheduling problems. The approach is oblivious to probability distributions of demand uncertainty, but what is needed is a relatively small training set based on historical data.

Our immediate further work is to further evaluate the applicability and scalability of this method on different scenarios such as larger-sized problems, different datasets (even synthetic ones) and other real-world dynamic optimisation problems like scheduling or other routing problems.

Acknowledgements

This research project was partially supported by National Research Foundation Singapore under its Corp Lab @ University scheme and Fujitsu Limited as part of the A*STAR-Fujitsu-SMU Urban Computing and Engineering Centre of Excellence.

References

Agussurja, L.; Cheng, S.-F.; and Lau, H. C. 2019. A state aggregation approach for stochastic multiperiod last-mile ride-sharing problems. *Transportation Science* 53(1):148–166.

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2016. Neural Combinatorial Optimization with Reinforcement Learning. *arXiv e-prints* arXiv:1611.09940.

Bent, R., and Van Hentenryck, P. 2004a. The value of consensus in online stochastic scheduling. In *ICAPS*, volume 4, 219–226.

Bent, R. W., and Van Hentenryck, P. 2004b. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research* 52(6):977–987.

Bertsekas, D. P.; Tsitsiklis, J. N.; and Wu, C. 1997. Rollout algorithms for combinatorial optimization. *Journal of Heuristics* 3(3):245–262.

Chiang, W.-C., and Russell, R. A. 1996. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research* 63(1):3–27.

De Filippo, A.; Lombardi, M.; and Milano, M. 2018. Methods for off-line/on-line optimization under uncertainty. In *IJCAI*, 1270–1276.

Goodson, J. C.; Thomas, B. W.; and Ohlmann, J. W. 2017. A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *European Journal of Operational Research* 258(1):216–229.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Nazari, M.; Oroojlooy, A.; Snyder, L.; and Takác, M. 2018. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, 9839–9849.

Osman, I. H. 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research* 41(4):421–451.

Pillac, V.; Gendreau, M.; Guéret, C.; and Medaglia, A. L. 2013. A review of dynamic vehicle routing problems. *European Journal of Operational Research* 225(1):1–11.

Powell, W. B. 2011. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, volume 842. John Wiley & Sons.

Psaraftis, H. N.; Wen, M.; and Kontovas, C. A. 2016. Dynamic vehicle routing problems: Three decades and counting. *Networks* 67(1):3–31.

Ritzinger, U.; Puchinger, J.; and Hartl, R. F. 2016. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research* 54(1):215–231.

Secomandi, N. 2001. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research* 49(5):796–802.

Ulmer, M. W.; Goodson, J. C.; Mattfeld, D. C.; and Thomas, B. W. 2017. Route-based markov decision processes for dynamic vehicle routing problems. Technical report, Technical report, Braunschweig.

Ulmer, M. W.; Goodson, J. C.; Mattfeld, D. C.; and Hennig, M. 2018. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Science* 53(1):185–202.

Ulmer, M. W.; Mattfeld, D. C.; and Köster, F. 2017. Budgeting time for dynamic vehicle routing with stochastic customer requests. *Transportation Science* 52(1):20–37.

Ulmer, M. W.; Thomas, B. W.; and Mattfeld, D. C. 2018. Preemptive depot returns for dynamic same-day delivery. *EURO Journal on Transportation and Logistics* 1–35.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, 2692–2700.

Voccia, S. A.; Campbell, A. M.; and Thomas, B. W. 2017. The same-day delivery problem for online purchases. *Transportation Science* 53(1):167–184.

Zhang, W., and Dietterich, T. G. 2000. Solving combinatorial optimization tasks by reinforcement learning: A general methodology applied to resource-constrained scheduling. *Journal of Artificial Intelligence Reseach* 1:1–38.