ORL: Reinforcement Learning Benchmarks for Online Stochastic Optimization Problems

Anonymous Authors[1]

Abstract

Reinforcement Learning (RL) has achieved state-of-the-art performance in a wide variety of domains: from robotics, to gaming, to traffic control. The domain of Operations Research (OR) is particularly amenable to RL approaches, because many of the canonical problems can be characterized as online stochastic optimization problems where the distribution of data is unknown. While there is a nascent literature at the intersection of RL and OR, there are no commonly accepted benchmarks which can be used to compare proposed approaches rigorously in terms of performance, scale, or generalizability. This paper aims to fill that gap by introducing open source OR+RL benchmarks for three canonical OR problems with a wide range of practical applications: Bin Packing, Newsvendor, and Vehicle Routing. We apply both well-known OR approaches and newer RL algorithms to these problems and analyze results. For each of these problems, we find that RL is competitive with or superior to the OR baselines, pointing the way for future theoretical work and highlighting RL's immediate potential utility in a host of real-world problems.

1. Introduction

Reinforcement learning (RL) has gained steam in recent years with state of the art results in gaming (Mnih et al., 2013; Silver et al., 2017), robotics (Hwangbo et al., 2019; Andrychowicz et al., 2018), computing (Mao et al., 2017; Marcus & Papaemmanouil, 2018; Mirhoseini et al., 2017), recommendation systems (Chen et al., 2019; Zhao et al., 2018), and many others. In RL, an agent interacts with an environment in a Markov Decision Process (MDP). The agent takes actions with the environment state as input and receives a scalar reward and the next state. RL algorithms learn a policy that maximizes the cumulative discounted reward from repeated interactions with the environment. Classic RL algorithms, such as Q-learning (Watkins & Dayan, 1992), require visitation to every discrete state, action pair for convergence. However, with the use of neural networks, RL can learn compact state representations and successful policies for high-dimensional state spaces (Riedmiller, 2005; Mnih et al., 2013). With enough interactions, state-of-the-art RL algorithms can learn to make intelligent decisions that even beat human strategies (Silver et al., 2017).

Operations Research has many combinatorial optimization problems, and solving them is often hard due to the problem size. Many recent works have shown promising results applying RL to these problems. Bello et al. (2016) show RL techniques produce near optimal solutions for the traveling salesman (TSP) and knapsack problems. Kool et al. (2018) use RL to solve TSP and its variants: vehicle routing, orienteering, and a stochastic variant of prize-collecting TSP. Nazari et al. (2018) solve both static and online versions of the vehicle routing problem. Gijsbrechts et al. (2018) apply RL to the dual sourcing inventory replenishment problem, and further demonstrate results on a real dataset. Kong et al. (2018) apply RL to online versions of the knapsack, secretary and adwords problems. Oroojlooyjadid et al. (2017) apply RL to the beer game problem. Lin et al. (2018) use RL for fleet management of taxis on a real life dataset.

Our contribution is to extend the existing OR/RL literature to a set of OR problems with direct applicability to real-world problems. In particular, we present benchmarks for three classic problems: Bin Packing, Newsvendor and Vehicle Routing. In each case, we show that out-of-the-box RL algorithms are competitive with or superior to well-known OR approaches. We aim to spur further work in two directions. First, we open source our code and parameterize the complexity of the problems in order to encourage fair comparisons of algorithmic contributions. Second, we highlight the close parallels between these problem formulations and a host of practical problems, showing how further research can directly impact production scenarios.

2. Bin Packing

In the classic bin packing problem, we are given items of different sizes and need to pack them into as few fixed size bins as possible. In the online stochastic version of this problem, items arrive one at a time and item sizes are drawn from an unknown distribution.

Many resource allocation problems in OR and Computer Science face uncertain future demand, and can be cast as variants of the online bin packing problem. For example, in warehouse and transportation operations, variants of bin packing can be seen in: the order assignment problem (where we assign orders to fulfillment resources), the tote packing problem (where we fill items as they arrive into totes for shipment), and the trailer truck packing problem. In computing, bin packing problems arise in datacenters where virtual machines must be placed on servers, thus allocating memory and computing to processes within a machine.

2.1. Problem Formulation

In the stochastic bin packing problem, items arrive online, one in each time period t, with $t \in \{1, \dots, T\}$. Items can be of different types $j \in \{1, ..., J\}$. The size of type j is s_j and the probability that an item is of type j is p_j. Without loss of generality, we assume item types are indexed in the increasing order of their size: $s_1 < s_2 < ... < s_J$. Upon arrival, the item needs to be packed into one of the bins, each with size B (we assume that $s_J < B < \infty$). A packing is considered feasible if the total size of the items packed in each bin does not exceed the bin size. The task is to find a feasible packing that minimizes the number of bins used to pack all of the items that arrive within the time horizon. We assume the item sizes s_j and bin size B are integers. We assume the number of bins one can open is unlimited and denote the sum of item sizes in a bin k as *level* h_k. After t items have been packed, we denote the number of bins at some level h as $N_h(t)$, where $h \in \{1, ..., B\}$. The formulation of online bin packing is from (Gupta & Radovanovic, 2012).

It can be shown that minimizing the number of non-empty bins is equivalent to minimizing the total waste (i.e. empty space) in the partially filled bins. Hence our objective is to minimize total waste $\sum_{t=0}^{T} W(t)$, where

$$W(t) \triangleq \sum_{h=1}^{B-1} N_h(t)(B-h) \tag{1}$$

We use $W_F^A(t)$ to denote the total waste after step t of algorithm A when the input samples come from distribution F. To train our RL algorithm, we define the cumulative reward up to time step t to be $W_F^{RL}(t)$. (Courcobetis & Weber, 1990) showed that any discrete distribution falls into one of three categories based on expected distribution $E[W_F^{OPT}(t)]$.

1. Linear waste (LW): $E[W_F^{OPT}(t)] = \Theta(t)$, e.g. $B = 9$, two item types of size $\{2, 3\}$ with probability $\{0.8, 0.2\}$ respectively.
2. Perfectly Packable (PP): $E[W_F^{OPT}(t)] = \Theta(\sqrt{t})$, e.g. $B = 9$, two item types of size $\{2, 3\}$ with probability $\{0.75, 0.25\}$ respectively.
3. PP with bounded waste (BW): $E[W_F^{OPT}(t)] = \Theta(1)$, e.g. $B = 9$, two item types of size $\{2, 3\}$ with probability $\{0.5, 0.5\}$ respectively.

We will train an RL policy for each of the three distribution types and compare our policy to the appropriate baseline.

2.2. Related Work

Bin packing is a well-studied problem in the operations research and computer science literature. The problem is already NP-hard in its basic form. As a result, many of the classical approaches to bin packing analyze the performance of approximation algorithms. We refer the readers to the survey (Coffman Jr. et al., 2013) for algorithmic approaches to classical bin packing and its generalizations.

For online bin packing, a simple heuristic – Best Fit – is known to use at most 1.7 times the optimal number of bins in the worst case (Johnson et al., 1974). Best Fit places an item in a bin where, if the item were to be packed, would leave the least amount of space. Another competitive heuristic is Sum of Squares (SS) heuristic (Csirik et al., 2006). In particular, SS is proven to be asymtotically optimal (up to constants) as the episode length gets large.

The simple heuristics described above are distribution agnostic. More sophisticated algorithms learn an empirical estimate of the item size distribution, leverage such distribution to solve a linear program, and use its dual to guide the online policy (Adelman & Nemhauser, 1999)(Rhee & Talagrand, 1993)(Iyengar & Sigman, 2004). This approach has been used to solve online packing and covering problems (Gupta & Molinaro, 2014)(Agrawal & Devanur, 2015).

2.3. Baseline Algorithms

We use the Sum of Squares (SS) heuristic and Best Fit (BF) as our baseline algorithm. When the tth item of size s arrives, SS picks a bin of level h^* that minimizes the value of the following sum-of-squares potential:

$$\sum_{h=1}^{B-1} (N_h(t))^2. \tag{2}$$

It can be shown that minimizing (2) is equivalent to minimizing:

$$h^* = \underset{h: N_h(t-1)>0 \text{ and } h+s \leq B}{\arg\min} [N_{h+s}(t-1) - N_h(t-1)], \tag{3}$$

where, $N_0 = N_B = 0$. Intuitively, SS tries to equalize the number of bins at each level. Due its simplicity, we implemented (3) version of SS.

BF selects a bin at the highest level that can still fit the item:

$$h^* = \underset{h:N_h(t-1)>0 \text{ and } h+s\leqslant B}{\arg\max} \quad h \tag{4}$$

2.4. Reinforcement Learning Formulation

We formulate this as an MDP, where the state $S_t \in \mathcal{S}$ is current item size s_j and the number of bins at each level: $N_h(n)$, where $h \in \{1, ..., B\}$. The action A is to pick a bin level which can fit the item. Thus, the number of actions possible is B with one action for each level and action 0 corresponds to opening a new bin. Initially, all the bins are empty. The reward R_t is the negative of incremental waste as each item is put into a bin. If the item is put into an existing bin, the incremental waste will reduce by item size. If the item is put into a new bin, the waste increases by the empty space left in the new bin. We mask the invalid actions such as picking a level for which bins do not exist yet.

2.5. Reinforcement Learning Algorithm

We use the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017). PPO is an actor-critic algorithm (Konda & Tsitsiklis, 2000), where the actor is represented by a policy network and takes the environment state as input and produces an action as the output. The critic is represented by a value network and takes the environment state as input and predicts the cumulative discounted reward that will be obtained from this state. Intuitively, the actor tells the agent how to act and the critic informs the agent how good an action was. The two neural networks are initialized with random weights, i.e. the agent takes random actions, and the agent interacts with the environment to generate a dataset of tuples: (S_t, A_t, R_t, S_{t+1}). The dataset is used to update the weights of the two neural networks. The updated neural networks are used to interact with the environment to generate more data and the cycle continues until training stops. We refer the reader to the original paper (Schulman et al., 2017) for a formal explanation of the algorithm.

We use a two-layer neural network with 256 hidden nodes each. The input to both policy and value network is the state, the output of the policy network is a vector giving the probabilities of taking any action in the action space, and the output of the value network is predicted value. During training, the agent explores the state space by sampling from the probability distribution of the actions generated by the policy network. During evaluation, the agent takes the action with the highest action probability. We mask actions by reducing the probability of invalid actions to -∞. Table 1 lists the hyperparameters we use.

2.6. Results

For each sample item size distribution (BW, PP, LW), we train the RL algorithm (PPO) and compare to the baseline algorithms (SS and BF). We consider two variations, bin size of 9 with distributions listed in section 2.1, and bin size of 100 and the following item size distribution:

Discount factor	0.995	KL coefficient	1.0
Experience Buffer	320000	Learning rate	0.0001
SGD Mini-batch	32768	Epochs	10
Entropy coefficient	0	# Workers	31
Episode length	10000	clip param	0.3

Table 1: Hyperparameters used in PPO for Bin Packing

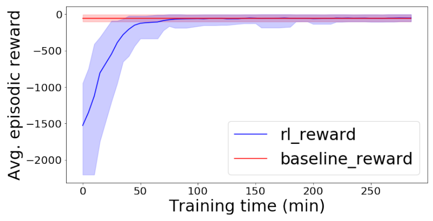
1. item sizes: $[1, 2, 3, 4, 5, 6, 7, 8, 9]$
2. item probabilities for BW: $[0.14, 0.10, 0.06, 0.13, 0.11, 0.13, 0.03, 0.11, 0.19]$
3. item probabilities for PP: $[0.06, 0.11, 0.11, 0.22, 0, 0.11, 0.06, 0, 0.33]$
4. item probabilities for LW: $[0, 0, 0, 1/3, 0, 0, 0, 0, 2/3]$.

We use a single machine with 4 GPUs and 32 CPUs for our experiments. At a high level, by the end of training RL outperforms or matches the baseline irrespective of distribution, and converges to a sensible learned policy.

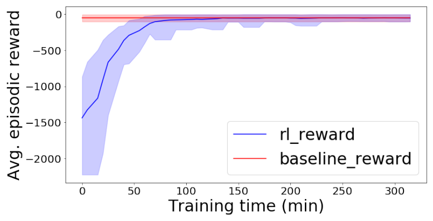
Figure 1 plots the reward function of the RL policy in training (blue) vs the Best Fit baseline (red) for bin size 100 and different item size distributions (BW, PP, and LW) as a function of training time (measured in minutes). The solid lines represent the mean reward of each policy, and the shaded bands represent the min/max rewards. By the end of training, RL either matches or outperforms the baseline policy for all three item size distributions. In particular, the reward gap between RL and baseline is the largest for LW distribution (which is expected, as both BF and SS are known to be suboptimal for LW distribution).

In Table 2, we inspect numerically the trained RL policy vs. baseline for bin size 100. Note that the exploration is turned off for the trained RL policy. Supporting what we observed in the initial figures, this table shows the final RL policy outperformes or matches the baseline for each distribution.

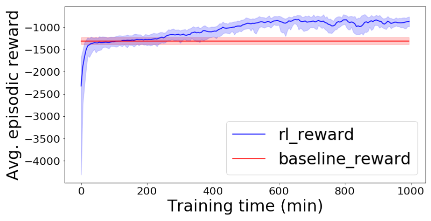
We test generalization of the RL policy by evaluating the trained policy with a different item distribution than the one it was trained on. For PP and BW distributions, the trained policy mutually translate well. Both the PP and BW policies perform as well as the baseline solutions for the LW distribution. The policy trained on the LW distribution generalizes reasonably well but does not do as well as the baseline solutions in the BW and PP distributions. We did observe overfitting if we pick model iterations from much later in training. We leave study of overfitting and generalization across distributions as future work. A note on scaling: the training time for bin size 100 compared to bin size 9 is about 3x, 4x and 10x more for PP, BW and LW respectively.



(a) RL vs baseline for BW distribution



(b) RL vs baseline for PP distribution



(c) RL vs baseline for LW distribution

Figure 1: Comparison of episodic rewards between RL and Best Fit baseline during training.

Algorithm	Perfect Pack		Bounded Waste		Linear Waste	
	μ	σ	μ	σ	μ	σ
RL with PP	-49.0	29.5	-48.0	29.5	-1358	44.2
RL with BW	-47.6	29.3	-53.9	26.4	-1368	48.0
RL with LW	-258.6	69.3	-143.9	84.9	-880.2	43
SS	-56.54	28.9	-56.61	30.2	-2091	92
Best Fit	-52.01	29.5	-51.4	28.9	-1314	53

Table 2: RL and baseline solution comparison for bin packing. Mean and standard deviations are calculated across 100 episodes.

Finally, we inspect the relative structure of the policies to ensure that RL is learning a sensible solution. In particular, we plot the state variable values as a function of the number of steps in an episode. Intuitively, the integral of these plots represents the waste, which we want to minimize. An optimal policy should show a (relatively) flat surface. We use bin size of 9 for this analysis for ease of manual inspection and study the linear waste distribution that highlights the difference between the Sum of Squares baseline and RL distinctly. From Figure 2, we see that the baseline policy leaves more open bins at a lower fullness, whereas RL only leaves open bins at level 8 (which cannot be closed once they reach that level). For other distributions, the graphs for both the baseline and RL policy look similar to each other.

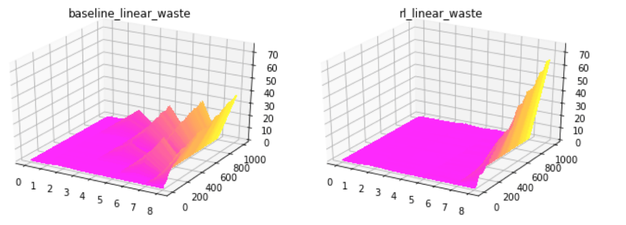


Figure 2: RL vs baseline solution for LW distribution

3. Multi-Period Newsvendor Problem with Lead Times

The Newsvendor problem (see e.g. (Zipkin, 2000)) is a seminal problem in inventory management wherein we must decide on an ordering decision (how much of an item to purchase from a supplier) to cover a single period of uncertain demand. The objective is to trade-off the various costs incurred and revenues achieved during the period, usually consisting of sales revenue, purchasing and holding costs, loss of goodwill in the case of missed sales, and the terminal salvage value of unsold items.

In practice, decisions are rarely isolated to a single period, and they are repeatedly and periodically taken and thus have a downstream impact. This makes the problem non-trivial, as compared to the single-period Newsvendor which has a known solution when the demand distribution is known. Additionally, purchased units do not, in general, arrive quasi-instantaneously, but rather after a few periods of transit from the vendor to their final destination, known as the lead time. The presence of lead times further complicates the problem. The multi-period newsvendor problem with lead times under the lost-sales model, where customers leave if there is no stock to satisfy their demand, is known not to admit a simple solution such as an order-up-to policy (which maintains an inventory level, and orders up to that level whenever inventory levels fall).

Solving the multi-period newsvendor problem with lead times and lost sales is a notoriously difficult problem (Zipkin, 2008). It requires keeping track of orders placed in different periods, leading to what is known as the *curse of dimensionality*, rendering any exact solution impractical even for small lead times of 2 and 3 periods, and outright infeasible at higher dimensions. As a result, the problem forms a good test-bed for RL algorithms given that the observation of rewards is delayed by the lead time and that it can be formulated as a Markov Decision Problem.

3.1. Related work

Some recent work has started looking at the newsvendor problem using a data-centric approach (Rudin & Vahn, 2014) or a reinforcement learning approach (Oroojlooyjadid et al., 2016). These have so far still remained focused on the single period problem and often trying to learn some of the inputs, such as demand. Few other papers have considered Reinforcement Learning in the context of inventory management, such as (Gijsbrechts et al., 2018) where a dual sourcing problem is tackled using RL.

3.2. Problem formulation

We consider the stationary, single-product, multi-period dynamic inventory management problem with vendor lead time (VLT) and stochastic demand. Here, the VLT l refers to the number of time steps between the placement and receipt of an order. The demand D is assumed to be stationary and Poisson distributed with mean μ. Items are purchased at a cost c and sold at a price p, and incur a penalty for lost sales k for each unit of unmet demand while any unit left over at the end of a period incurs a holding cost h. A discount factor γ is used.

The problem is formulated as a Markov Decision Process:

State: The state S of the problem is given by

$$S = (p, c, h, k, \mu, x_0, \dots, x_{l-1})$$

where x_0 is the on-hand inventory, x_1 the units to be received one period hence, and so on.

Action: In each period the state of the system is observed and an action $A = q$ is taken, consisting of the size of the order placed and to arrive l time periods later.

Reward: We first incur the purchasing cost corresponding to the procured units given the action a. A realization d of the demand D, which we recall is Poisson distributed with mean μ, is then observed, and demand is satisfied as much as is possible given on-hand levels. Missed sales incur a loss of goodwill k per unit, while leftover units incur a holding cost h:

$$R = p\min(x_0, d) - ca - h(x_0 - d)^+ - k(d - x_0)^+.$$

where $(x)^+ = \max(x, 0)$.

Transition: The state of the system S is then updated to S_+ by moving all pipeline units downstream and incorporating the newly purchased units:

$$S_+ = (p, c, h, k, \mu, (x_0 - d)^+ + x_1, x_2, \dots, x_{l-1}, a).$$

3.3. Baseline solution

As noted in Section 3, it is impractical or even infeasible to solve the problem exactly to get a baseline. However, it is possible to use heuristics that provide good approximations to the optimal solution. In particular, a way to tackle the problem is to approximate it by its backlogging counterpart, for which a closed form solution of the optimal policy exists in the form of an order-up-to policy characterized by the following critical ratio:

$$CR = \frac{p - \gamma c + k}{p - \gamma c + k + h}.$$

As a result, letting $z^* = F_l^{-1}(CR)$, where F_l is the cumulative distribution function of the l period demand, the policy is given by:

$$a = \left(z^* - \sum_{i=0}^{l-1} x_i\right)^+.$$

3.4. RL solution

We used a Proximal Policy Optimization algorithms (PPO) (Schulman et al., 2017) as implemented in the RLLib package (Liang et al., 2017), where the policy is represented by a neural network. We used a neural network of size (256,256) and the hyperparameters presented in Table 3.

Learning rate	0.0001
SGD Mini-Batch Size	32768
Train Batch Size	320000
Episode length	40

Table 3: Hyperparameters used in PPO for Newsvendor

3.5. Results

We present the results obtained using a VLT of 5. The economic parameters were chosen so that $p, c \in [0, 100]$, $h \in [0, 5]$ and $k \in [0, 10]$, while the demand mean μ was such that $\mu \in [0, 200]$.

In the course of our experiments we saw that how we sampled the initial state and item characteristics is critical. We modified the sampling scheme so as to sample "interesting" states and avoid trivial or misleading initial configurations. For example, it should be easily learned that nonprofitable items (including the penalty for lost sale) should not be purchased, but this is not really indicative of a good performance since in practice such items would not be part of inventory. Similarly, high initial inventory levels would not lead to any purchase either because inventory can simply be drained over long periods of time, resulting in a trivial decision that results in a high reward that would unfairly favor the perceived quality of the RL solution.

As a result, the sampling was performed as follows: $p \sim U[0, 100]$, $c \sim U[0, p]$, $h \sim U[0, \min(c, 5)]$, $k \sim U[0, 10]$ and $\mu \sim U[0, 200]$ for the economic and demand parameters; where $U[a, b]$ denotes a uniformly random variable between a and b. The initial state was simply set to be $\mathbf{0}$.

Figure 3 compares the results obtained by the RL algorithm to the baseline. We observe that the RL solution was nearing the benchmark when the experiment was stopped.

While solving this problem numerically is intractable, the optimal inventory policy structures are well known. It is thus of interest to check whether their properties are being learned by the RL algorithm. Given the dimension of the problem, we cannot observe the entire policy, but can investigate slices thereof. We thus fix price, cost, holding

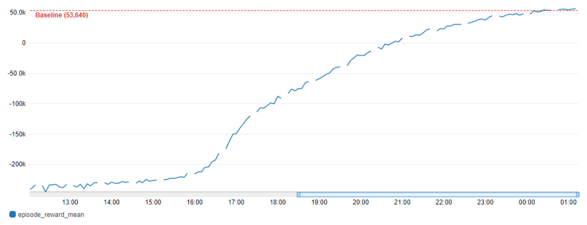


Figure 3: RL solution reward in the newsvendor problem.

cost, penalty for lost sale and mean demand to 50, 25, 0.5, 5 and 100, respectively, and plot the optimal policy in the space $(0, 0, x_2, x_3, 0)$ in Figure 4. The figure shows contour curves of the buying quantity as a function of the inventory state. We observe that the algorithm is slowly learning the policy structure and we can start to observe monotonicity of the policy along most directions.

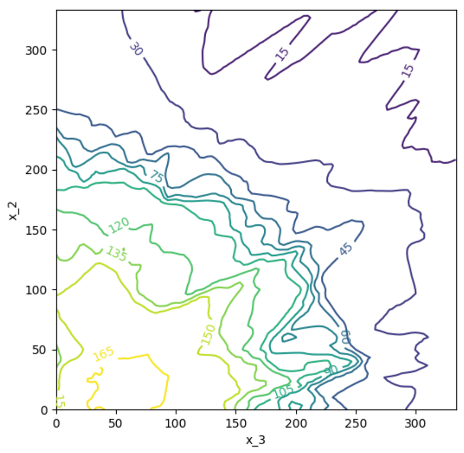


Figure 4: Slice of the learnt policy for the multi-period newsvendor.

4. Vehicle Routing Problem

In the traveling salesman problem (TSP), we find the shortest route that visits each node in a graph exactly once and returns to the starting node. TSP is one of the most widely studied problems in combinatorial optimization both due to its NP-hard nature and as well as its wide variety of practical applications. The vehicle routing problem (VRP) is a generalization of TSP where one or more vehicles are expected to visit the nodes in a graph, usually to satisfy customer demand. VRP is also a well studied topic and has very important applications, especially in supply chain and logistics. These real-life applications lead to many variants of VRP with different constraints, such as capacitated vehicles, pickups and deliveries on the route, time windows associated with each pickup and delivery etc.

An important extension of VRP is where some of the information about the graph is revealed over time, such as demand at each node and travel time. This class of VRP is called dynamic VRP (DVRP, also known as real-time or online VRP). Stochastic VRP (SVRP) is where one or more problem parameters are stochastic with some known probability distributions (as opposed to arbitrary or adversarial distributions). In many real-life applications, the relevant VRP is both stochastic and dynamic (SDVRP), which is also focus of this work. We formulate a variant of SVRP and compare solution approaches from the Operations Research (OR) and Reinforcement Learning (RL) literature.

4.1. Problem Formulation

We consider a decentralized version of the VRP and introduce the corresponding default parameters. Consider a delivery contractor for a crowd-sourced app. Orders arrive at the driver's phone app over time in a dynamic manner. Each order has a reward (e.g. delivery fee) associated with it, known to the driver at the time of order creation, and it is assigned to a location in the city. "City" here means the whole Euclidean space in which the VRP problem lives. The city consists of mutually exclusive regions ($num\_zones = 4$) that generate orders at different rates ($order\_probs\_per\_zone = (0.5, 0.3, 0.1, 0.1)$) and with rewards according to truncated normal distribution with different ranges ($order\_reward\_max = (12, 8, 5, 3), order\_reward\_min = (8, 5, 2, 1)$). The orders need to be delivered within a time limit ($order\_promise = 60$), the timer starts with the order creation and is same for all orders. The driver has to accept an order and pick up the package from a given location prior to delivery. Orders that are not accepted disappear probabilistically ($order\_timeout\_prob = 0.15$) when other drivers accept the orders. The vehicle has a capacity limit ($driver\_capacity = 4$), but the driver can accept unlimited orders and plan their route accordingly. Each time step and unit distance travelled adds a fixed cost ($penalty\_per\_timestep = 0.1$, $penalty\_per\_move = 0.1$), and the whole episode length is 1000. The driver's goal is to maximize the total net reward. Our formulation is known as stochastic and dynamic capacitated vehicle routing problem with pickup and delivery, time windows and service guarantee.

4.2. Related Work

There is a substantial literature on VRP (Eksioglu et al., 2009). The closest VRP variant to the problem considered in this paper is the Pickup and Delivery Problem with Time Windows (PDPTW) (Cordeau et al., 2008), which has some additional complexities over vanilla VRP. Due to such complexities, there are fewer exact solution approaches (Lu & Dessouky, 2004; Mahmoudi & Zhou, 2016), and a majority of the literature focuses on heuristics. When the problem is also stochastic and dynamic, exact solution methods become intractable except for very specific problem settings. In such cases, anticipatory algorithms that simulate sample future scenarios and merge solutions to those samples are a common choice (Berbeglia et al., 2010; Berhan et al., 2014; Ritzinger et al., 2016; Ghiani et al., 2012).

Reinforcement Learning (RL) methods have been successfully used for solving the Traveling Salesman Probelm (TSP). Bello et al. (2016) employ a pointer network (Vinyals et al., 2015) to optimize the policy, and train an actor-critic algorithm with the negative tour length as the reward signal. Khalil et al. (2017) develop a single model based on graph embeddings. They use the DQN algorithm to train a greedy policy and graph embedding network simultaneously. For VRP, Kool et al. (2018) utilize the Transformer architecture (Vaswani et al., 2017) to develop a model fully based on attention layers. Their proposed model is trained by policy gradients with a greedy baseline, and evaluated on both standard Capacitated VRP (CVRP) and Split Deliverry VRP (SDVRP). Nazari et al. (2018) further improve the algorithm using embedded inputs and allow the customers and their demands to be stochastic.

4.3. Baseline Algorithm

We modify the classical three-index Mixed Integer Programming (MIP) formulation (Lu & Dessouky, 2004; Ropke & Cordeau, 2009; Furtado et al., 2017). This deterministic MIP is solved for the available orders in the environment. It is further resolved when a new order arrives, if one of the existing orders expires or, when all of the actions are executed. When we solve the MIP, orders already accepted or in transit are modeled as starting conditions. We leave anticipatory models to future work (see Section 4.2).

4.4. Reinforcement Learning Algorithm

State: We include pickup location $\mathbf{p}_t$, driver info $\mathbf{d}_t$, and order info $\mathbf{o}_t$. Driver info contains the driver's position $\mathbf{h}_t$ and the capacity left c_t. Order info contains the orders' location $\mathbf{l}_t$, status $\mathbf{w}_t$ (open, accepted, picked up or delivered), the time elapsed since each order's generation $\mathbf{e}_t$ and the corresponding dollar value $\mathbf{v}_t$. Thus, the state is $S_t = (\mathbf{p}_t, \mathbf{d}_t, \mathbf{o}_t)$, in which $\mathbf{d}_t = (\mathbf{h}_t, c_t)$, $\mathbf{o}_t = (\mathbf{l}_t, \mathbf{w}_t, \mathbf{e}_t, \mathbf{v}_t)$.

Action The agent chooses an action A_t from five options – accept an order, pick up an accepted order, go to a customer's node for delivery, head to a specific pickup location, or wait and stay unmoved.

Reward: The reward R_t is the total value of all delivered orders $\mathbf{f}_t$ minus the cost $\mathbf{q}$. $\mathbf{f}_t$ is divided into 3 equal parts for reward shaping: when the order gets accepted, picked up, and delivered respectively. Thus we have:

$$R_t = \frac{1}{3}\Big(\mathbb{1}_{accepted} + \mathbb{1}_{picked-up} + \mathbb{1}_{delievered}\Big)\mathbf{f}_t - \mathbf{q}_t,$$

where $\mathbf{q}_t = (q_{time} + q_{move} + q_{failure})$. q_{time} is the time cost, q_{move} is the moving cost (per time step). $q_{failure}$ is a large penalty ($order\_miss\_penalty = 50$) if the agent accepts an order but fails to deliver within the promised time.

The vehicle's capacity remains unchanged if an order is accepted but not picked up. In effect, this grants the agent the flexibility to accept more orders than its capacity, which can be picked up later when space allows. The action of heading to a specific pickup location enables the agent to learn to stay near popular pick up locations.

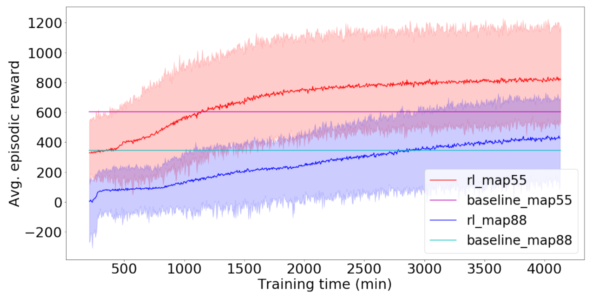
We impose action masking during the policy training. The agent cannot perform the following invalid actions: *(i)* pick up an order when its remaining capacity is 0; *(ii)* pick up an order that is not yet accepted; *(iii)* deliver an order that is not yet picked up.

To train the policy, we apply the APE-X (Horgan et al., 2018) DQN (Mnih et al., 2013) algorithm due to its ability to scale by generating more experience replays and picking from them in a distributed prioritized fashion. We feed the input into a two-layer neural network with 512 hidden units each to compute the Q values.

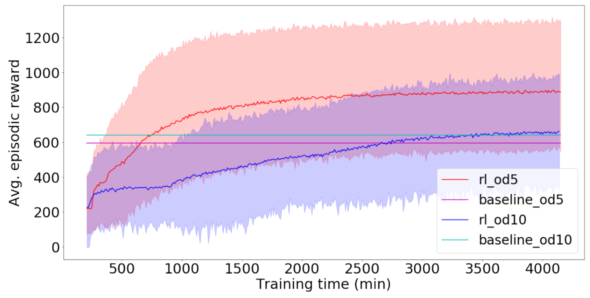
4.5. Results

For multiple problem scales determined by map size ($map\_size \in \{5 \times 5, 8 \times 8\}$), maximum number of orders ($order \in \{5, 10\}$) and number of pick-up location ($n \in \{2, 3\}$), we conduct experiments to compare the behavior of RL and the MIP baseline solutions. We examine the trained RL policy's ability to generalize to different order distributions. The hyperparameters used for algorithm training can be found in Table 5. Overall, the RL approach outperforms the baseline across different instance sizes, and generalizes well for unseen order patterns.

Figure 5a-5b compares the episodic rewards for the RL policy and the baseline algorithm during training. The shaded band around the mean line shows the minimum and maximum rewards. For readability, the graphs are clipped to skip the initial 3.5 hours of training as the rewards are highly negative and skew the Y-axis scale. With larger map size or higher order number, the training time required for the agent to achieve rewards equivalent to baseline is higher. This is expected as both the observation and action space increase, the agent requires more exploration to converge to a reasonable policy. Even after three days of training, the rewards for larger instances keeps growing gradually. The agent slowly learns to fully utilize the vehicle capacity. Over time, the agent also learns to reject (not accept) orders



(a) RL vs baseline solution for VRP with 3 pick-up locations, 5 orders and map sizes 5×5 and 8×8



(b) RL vs baseline solution for VRP with 2 pick-up locations, map 5×5, and number of orders 5 and 10.

Figure 5: RL vs baseline during policy training process.

which are likely to incur penalty.

As the agent is trained longer, there is potential for the policy to overfit. In order to test this issue, we train another policy with a shifted hot order-zone distribution $((0.1, 0.5, 0.3, 0.1))$, and evaluate against the baseline results both using the original order-zone distribution $((0.5, 0.3, 0.1, 0.1))$. Table 4 summarizes the evaluation results. It is observed that the policy is able to outperform the baseline consistently during evaluation phase.

We also present the rewards with and without the order miss penalty $q_{failure}$ to further understand the agent's behavior about order delivery misses. The reward values are close for problems with less number of pick up locations and fewer orders. As the number of pick-up locations become larger, the gap between the rewards increases. One explanation is the agent cannot multiplex order deliveries from different pick-up locations and the likelihood of missing the order delivery increases. This behavior is also seen if the number of orders is higher. Even though the RL agent reward is better than the baseline, there is still scope for improvement by reducing the number of order delivery misses.

Problem Instance	RL Evaluation Reward		MIP Reward
	Without $q_{failure}$	With $q_{failure}$	
5 by 5 map, 5 orders 2 pick-up locations	854.45 (136.03)	838.30 (154.12)	595.91
5 by 5 map, 5 orders 3 pick-up locations	754.27 (116.48)	730.40 (132.75)	642.62
5 by 5 map, 10 orders 2 pick-up locations	774.63 (143.34)	692.34 (200.65)	640.01
8 by 8 map, 5 orders 2 pick-up locations	548.53 (107.40)	536.55 (112.33)	410.58
8 by 8 map, 5 orders 3 pick-up locations	429.20 (102.37)	373.7 (129.98)	246.25

Table 4: RL and baseline solution comparison for VRP. Values in the brackets are standard deviations and mean reward is calculated using 50 episodes.

5. Conclusion

In this paper, we have established Deep Reinforcement Learning (DRL) benchmarks for three canonical Operations Research problems: Bin Packing, Newsvendor, and Vehicle Routing. We formulated an online stochastic version of each problem, and compared state-of-the-art OR approaches with vanilla RL techniques. In each case, RL either outperforms or is competitive with the baseline. While we do not overcome the NP-hardness of the problems, as wall-clock training time scales with problem size, we find that DRL is a good tool for these problems because neural networks are good at state space approximation. These results illustrate the potential value of RL for a wide range of real-world industrial OR problems, from order assignment, to retail buying, to real-time routing. Our experiments indicate the following issues as important for making RL solutions more practical in the future: overfitting to a particular distribution, initialization of the RL model, and enforcement of constraints (via e.g. masking).

Replay buffer alpha	0.5	# steps for Q	3
Replay buffer eps	0.1	Learning rate	1e-3
Final explore eps	0.01	Adam epsilon	1.5e-4
Replay buffer size	1e6	# Workers	7
Episode length	10000	Training Batch	512

Table 5: Hyperparameters used in APEX-DQN for VRP

We identify two major areas for future work. First, we used out-of-the-box RL algorithms, with almost no problem-specific tweaking. Further research can add value by testing various RL algorithms, neural net structures, etc. and seeing their relative value in each problem especially as complexity scales up. Second, in this paper we only looked at canonical, theoretical models. Further research should endeavor to apply these RL techniques to real-world industrial problems.

References

Adelman, D. and Nemhauser, G. L. Price-directed control of remnant inventory systems. *Operations Research*, 47 (6):889–898, 1999. doi: 10.1287/opre.47.6.889.

Agrawal, S. and Devanur, N. R. Fast algorithms for online stochastic convex programming. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pp. 1405–1424, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.

Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Berbeglia, G., Cordeau, J.-F., and Laporte, G. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8 – 15, 2010. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2009.04.024. URL `http://www.sciencedirect.com/science/article/pii/S0377221709002999`.

Berhan, E., Beshah, B., Kitaw, D., and Abraham, A. Stochastic vehicle routing problem: A literature survey. *Journal of Information & Knowledge Management*, 13(03):1450022, 2014. doi: 10.1142/S0219649214500221. URL `https://doi.org/10.1142/S0219649214500221`.

Chen, M., Beutel, A., Covington, P., Jain, S., Belletti, F., and Chi, E. H. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 456–464. ACM, 2019.

Coffman Jr., E. G., Csirik, J., Galambos, G., Martello, S., and Vigo, D. *Bin Packing Approximation Algorithms: Survey and Classification*, pp. 455–531. Springer New York, New York, NY, 2013. ISBN 978-1-4419-7997-1. doi: 10.1007/978-1-4419-7997-1_35.

Cordeau, J.-F., Laporte, G., and Ropke, S. Recent models and algorithms for one-to-one pickup and delivery problems. In Golden, B., Raghavan, S., and Wasil, E. (eds.), *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43. Springer, Boston, MA, 2008.

Courcobetis, C. and Weber, R. Stability of on-line bin packing with random arrivals and long-run-average constraints. *Probability in the Engineering and Informational Sciences*, 4(4):447–460, 1990.

Csirik, J., Johnson, D. S., Kenyon, C., Orlin, J. B., Shor, P. W., and Weber, R. R. On the sum-of-squares algorithm for bin packing. *Journal of the ACM (JACM)*, 53(1):1–65, 2006.

Eksioglu, B., Vural, A. V., and Reisman, A. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472 – 1483, 2009. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2009.05.009. URL `http://www.sciencedirect.com/science/article/pii/S0360835209001405`.

Furtado, M. G. S., Munari, P., and Morabito, R. Pickup and delivery problem with time windows: A new compact two-index formulation. *Operations Research Letters*, 45(4):334 – 341, 2017. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2017.04.013. URL `http://www.sciencedirect.com/science/article/pii/S0167637717302651`.

Ghiani, G., Manni, E., and Thomas, B. W. A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. *Transportation Science*, 46 (3):374–387, 2012. doi: 10.1287/trsc.1110.0374.

Gijsbrechts, J., Boute, R. N., Van Mieghem, J. A., and Zhang, D. Can deep reinforcement learning improve inventory management? performance and implementation of dual sourcing-mode problems. *Performance and Implementation of Dual Sourcing-Mode Problems (December 17, 2018)*, 2018.

Gupta, A. and Molinaro, M. How experts can solve lps online. In Schulz, A. S. and Wagner, D. (eds.), *Algorithms - ESA 2014*, pp. 517–529, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

Gupta, V. and Radovanovic, A. Online stochastic bin packing. *arXiv preprint arXiv:1211.2687*, 2012.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

Iyengar, G. and Sigman, K. Exponential penalty function control of loss networks. *Ann. Appl. Probab.*, 14(4):1698–1740, 11 2004. doi: 10.1214/105051604000000936.

Johnson, D., Demers, A., Ullman, J., Garey, M., and Graham, R. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3(4):299–325, 1974. doi: 10.1137/0203025.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pp. 6348–6358, 2017.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *Advances in neural information processing systems*, pp. 1008–1014, 2000.

Kong, W., Liaw, C., Mehta, A., and Sivakumar, D. A new dog learns old tricks: Rl finds classic optimization algorithms. 2018.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! 2018.

Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I. Rllib: Abstractions for distributed reinforcement learning. *arXiv preprint arXiv:1712.09381*, 2017.

Lin, K., Zhao, R., Xu, Z., and Zhou, J. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1774–1783. ACM, 2018.

Lu, Q. and Dessouky, M. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38(4):503–514, 2004. doi: 10.1287/trsc.1030.0040.

Mahmoudi, M. and Zhou, X. Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state–space–time network representations. *Transportation Research Part B: Methodological*, 89:19 – 42, 2016. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2016.03.009. URL `http://www.sciencedirect.com/science/article/pii/S0191261516301497`.

Mao, H., Netravali, R., and Alizadeh, M. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 197–210. ACM, 2017.

Marcus, R. and Papaemmanouil, O. Towards a hands-free query optimizer through deep learning. *arXiv preprint arXiv:1809.10212*, 2018.

Mirhoseini, A., Pham, H., Le, Q. V., Steiner, B., Larsen, R., Zhou, Y., Kumar, N., Norouzi, M., Bengio, S., and Dean, J. Device placement optimization with reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2430–2439. JMLR. org, 2017.

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

Nazari, M., Oroojlooy, A., Snyder, L., and Takác, M. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, pp. 9861–9871, 2018.

Oroojlooyjadid, A., Snyder, L., and Takáč, M. Applying deep learning to the newsvendor problem. *arXiv preprint arXiv:1607.02177*, 2016.

Oroojlooyjadid, A., Nazari, M., Snyder, L., and Takáč, M. A deep q-network for the beer game: A reinforcement learning algorithm to solve inventory optimization problems. *arXiv preprint arXiv:1708.05924*, 2017.

Rhee, W. and Talagrand, M. On-line bin packing of items of random sizes, ii. *SIAM Journal on Computing*, 22(6): 1251–1256, 1993. doi: 10.1137/0222074.

Riedmiller, M. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pp. 317–328. Springer, 2005.

Ritzinger, U., Puchinger, J., and Hartl, R. F. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231, 2016. doi: 10.1080/00207543.2015.1043403. URL `https://doi.org/10.1080/00207543.2015.1043403`.

Ropke, S. and Cordeau, J.-F. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009. doi: 10.1287/trsc.1090.0272.

Rudin, C. and Vahn, G.-Y. The big data newsvendor: Practical insights from machine learning. 2014.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

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

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Zhao, X., Xia, L., Zhang, L., Ding, Z., Yin, D., and Tang, J. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 95–103. ACM, 2018.

Zipkin, P. H. *Foundations of inventory management*. McGraw-Hill, 2000.

Zipkin, P. H. Old and new methods for lost-sales inventory systems. *Operations Research*, 56(5):1256–1263, 2008.