A Reinforcement Learning Approach to Dual-Sourcing Inventory Problem

(ORIE 6590 Final Project)

Tonghua Tian and Xumei Xi
School of Operations Research and Information Engineering
Cornell University
`{tt543,xx269}@cornell.edu`

May 2021

1 Introduction

It is common practice that companies depend on multiple suppliers for product ordering. We focus on the model where we have a regular supplier and an express supplier. The regular supplier provides a cheaper price point but takes longer to deliver, while the express supplier has faster delivery at a higher expense. In our project, we examine the dual-sourcing inventory system and implement a reinforcement learning algorithm to make prudent ordering decisions. Please see the GitHub repo (`github.com/xixumei1226/dual-sourcing-rl`) for more details.

1.1 MDP Formulation

In this subsection, we formally introduce the Markov decision process (MDP) formulation of the problem. As in the work [XG18], suppose we have a regular supplier R, with a longer lead time L_r and a lower cost c_r, and an express supplier E, with a shorter lead time L_e and a higher cost c_e. We assume $L_r > L_e + 1$ and $c_r < c_e$. Demands are generated as an i.i.d. sequence $\{D_t, t \geq 0\}$, distributed as the nonnegative random variable D. Denote the unit holding and backorder costs by $h > 0$ and $b > 0$, respectively. Let I_t denote the on-hand inventory, and $\mathbf{q}_t^r = \{q_{t-i}^r, i \in [L_r]\}$, $\mathbf{q}_t^e = \{q_{t-i}^e, i \in [L_e]\}$ denote the pipeline vectors of orders placed but not yet delivered with R and E at the start of period t, where q_{t-i}^r, q_{t-i}^e are the orders placed at period $t - i$.

At period t, a sequence of events happen in the following order:

1. The on-hand inventory I_t is observed.
2. New orders q_t^r and q_t^e are placed with R and E.
3. New inventory $q_{t-L_r}^r + q_{t-L_e}^e$ is delivered and added to the on-hand inventory.

4. The demand D_t is realized; the inventory and pipeline vectors are updated.

5. Costs for period t are incurred.

Notice the on-hand inventory is updated according to

$$I_{t+1} = I_t + q^r_{t-L_r} + q^e_{t-L_e} - D_t.$$

The pipeline vectors are updated according to

$$\mathbf{q}^r_{t+1} = (q^r_{t-L_r+1}, \dots, q^r_{t-1}, q^r_t),$$
$$\mathbf{q}^e_{t+1} = (q^e_{t-L_e+1}, \dots, q^e_{t-1}, q^e_t).$$

Let C_t be the sum of the ordering cost and holding and backorder costs incurred at time period t:

$$C_t = c_r q^r_t + c_e q^e_t + hI^+_{t+1} + bI^-_{t+1}.$$

Note that in [XG18], orders placed at period t are charged at period $t + L_e$. Here we use a different accounting method to simplify the notations for the corresponding MDP. This has no effect on the problem considered.

An admissible policy π consists of a sequence of deterministic measurable functions $\{f^\pi_t, t \geq 0\}$ from $\mathbb{R}^{L_r+L_e+1}$ to $\mathbb{R}^2_+$. Specifically, the new orders placed at period t are given by $(q^r_t, q^e_t) = f^\pi_t(\mathbf{q}^r_t, \mathbf{q}^e_t, I_t)$. Let Π denote the family of all admissible policies. The current cost under a policy π is denoted by C^π_t. We aim to minimize the long-run average cost

$$C(\pi) = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[C^\pi_t].$$

Assume the demands follow Poisson distribution: $D \sim \text{Pois}(\lambda)$, where $\lambda > 0$. Furthermore, assume the orders can only take integer values. Then the above process can be formulated as a discrete MDP. At period t, let $s_t = (\mathbf{q}^r_t, \mathbf{q}^e_t, I_t)$ be the state of the system, and let $a_t = (q^r_t, q^e_t)$ be the action taken. The state space $\mathcal{S}$ and action space $\mathcal{A}$ are given by

$$\mathcal{S} = \mathbb{Z}^{L_r}_+ \times \mathbb{Z}^{L_e}_+ \times \mathbb{Z}, \quad \mathcal{A} = \mathbb{Z}^2_+.$$

Note that C_t is a function of s_{t+1} instead of s_t. So the reward of step t is actually received at step $t-1$:

$$r_t = r(s_t, a_t) = -C_{t-1} = -c_r q^r_{t-1} - c_e q^e_{t-1} - hI^+_t - bI^-_t.$$

Then we define the long-run average reward as

$$J(\pi) = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[r_{t+1}|\pi]. \tag{1}$$

Define the function $g : \mathbb{R}^{L_r+L_e+1} \times \mathbb{R}^2 \to \mathbb{R}^{L_r+L_e+1}$ as

$$g(x_1, \dots, x_{L_r}, y_1, \dots, y_{L_e}, z, a_1, a_2) = (x_2, \dots, x_{L_r}, a_1, y_2, \dots, y_{L_e}, a_2, z + x_1 + y_1).$$

Then

$$s_{t+1} = g(s_t, a_t) - (0, \dots, 0, D_t).$$

Hence the transition probabilities are given by

$$\mathbb{P}(s_{t+1} = g(s_t, a_t) - (0, \dots, 0, k) \mid s_t, a_t) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k = 0, 1, \dots.$$

1.2 Prior Works

In this subsection, we briefly review some prior works on the dual-sourcing inventory systems. Researchers have been investigating the dual-sourcing problem for many years due to its practicality as well as its intractability. Earlier works have already showed that when the lead time difference is exactly one, order-up-to policies are optimal. However, once we step into the regime where the lead time difference grows larger, such policies fall short. In our project, we focus on the general case where the lead time difference is relatively large. Over the years, people have developed various heuristic policies to better approximate the optimum. The work [VSW08] proposed the idea of dual-index (DI) policies which have two order-up-to levels for the two suppliers. Under such policies, we keep track of two inventory positions and make orders in an effort to hold the corresponding inventory positions up to certain levels. Another simple and intuitive policy is the tailored base-surge (TBS) policy proposed in the work [AM10]. With a TBS policy, a constant order is placed at the regular source in each period to meet a base level of demand, while the orders we place at the express source follow an order-up-to rule to manage demand surges. As showed in the work [KKM11], TBS policies are comparable to DI policies in practice, and outperform DI policies for some problem instances, especially with an increase in the lead time difference. It has also been prove theoretically in the paper [XG18] that when the lead time of the express source is fixed, a simple TBS policy is asymptotically optimal as the lead time of the regular source increases.

1.3 Project Goals

In this subsection, we state our goals for the project.

1. Develop practical reinforcement learning algorithm to approximate the optimal policy.
2. Compare our performance with heuristic policies like the TBS policy.
3. Gain insights by potentially analyzing the intrinsic structure within the output policy.
4. Acknowledge the challenges encountered and the limitation of our approach. Provide possible future directions.

2 Actor-Critic Algorithm

To maximize the long-run average reward (1), we use the actor-critic framework. As pointed out in the survey [GBLB12], actor-critic methods combine the advantages of actor-only and critic-only methods. The basic idea is that an actor approximates the policy while a critic estimates the value function, both using function approximation scheme. Note that since we have a large state space, it is indeed suitable to use function approximation.

2.1 Natural Actor-Critic (NAC) for Long-Run Average Reward

In contrast to the regular policy gradient, natural gradient incorporates knowledge about the curvature of the space into the gradient. The use of natural gradient can produce better conditioning and is capable of further reducing variance in the gradient estimate. We present the algorithm in the context of linear function approximation, in alignment with the paper [BSGL09]. However, in our experiments, we use a neural network in lieu of a simple linear function. The authors note that we do not find a proof of convergence for general neural networks. Theoretical convergence is only guaranteed in the linear case.

The basic idea is that the value function parameters are updated using temporal difference learning while the policy parameters are learned by stochastic gradient descent. Please see Appendix A for a complete introduction and technical discussion on the NAC algorithm.

2.2 Advantage Actor-Critic (A2C) for Discounted Reward

The advantage actor-critic (A2C) method is a synchronous version of the asynchronous advantage actor-critic (A3C) method proposed in the work [MBM+16]. It is widely used in practice and various mainstream reinforcement learning packages support this algorithm. We consider the discounted setting where we have a discount parameter $\gamma \in (0,1)$. The discounted reward is defined as

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| s_0, \pi\right],$$

where the initial state s_0 is either determined or random. For a policy π, define the state value function $V^\pi(s)$ and the state-action value function $Q^\pi(s,a)$ according to

$$\begin{aligned} V^\pi(s) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, \pi\right], \quad \forall s \in \mathcal{S}, \\ Q^\pi(s,a) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, a_0 = a, \pi\right], \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \end{aligned}$$

The advantage function is defined as

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s).$$

We parameterize the policy π with θ and value function V with v. Instead of constructing two separate neural networks for the actor and the critic, we employ the idea of sharing some parameters between them, as pointed out in the work [MBM+16]. Specifically, we use one neural network that has one softmax output for the policy π_θ and one linear output for the value function V_v, sharing the same set of non-output layers.

In each learning episode, we simulate multiple steps of the environment, according to the policy π_θ from the last episode, to get n-step returns. Using n-step return instead of one-step return is beneficial since it stabilizes the training process. Afterwards, we obtain the rollout trajectory obtained as $\{(s_t, a_t, r_t, s_{t+1}\}_{t=T}^{T+n}$. Meanwhile, we get the value function estimate V_v, where v is also from the last episode. Then we minimize the combined loss from the actor and the critic

$$L = L_{\text{actor}} + W \cdot L_{\text{critic}}, \tag{2}$$

where $W > 0$ is a weight hyperparameter that controls the relative update speed. The loss functions for the actor and critic are defined as

$$L_{\text{actor}} = -\sum_{t=0}^{n} \log \pi_\theta(s_{T+t}, a_{T+t}) \left(\sum_{i=t}^{n} \gamma^{i-t} r_{T+i} - V_v(s_{T+t}) \right), \tag{3}$$

$$L_{\text{critic}} = \sum_{t=0}^{n} \left(\sum_{i=t}^{n} \gamma^{i-t} r_{T+i} - V_v(s_{T+t}) \right)^2. \tag{4}$$

Recall that by the policy gradient theorem, we have

$$\nabla J(\pi) = \frac{1}{1-\gamma} \sum_{s \in \mathcal{S}} d^{\pi,\gamma}(s) \sum_{a \in \mathcal{A}} \nabla \pi(s,a) A^\pi(s,a), \tag{5}$$

where $\{d^{\pi,\gamma}(s), s \in \mathcal{S}\}$ represents the discounted future state distribution. We use the state value function as the baseline to reduce variance in the policy gradient. The term inside the parentheses in equation (3) serves as an estimate for the advantage function:

$$\hat{A}(s_{T+t}, a_{T+t}) = \underbrace{\sum_{i=t}^{n} \gamma^{i-t} r_{T+i}}_{\hat{Q}(s_{T+t}, a_{T+t})} - V_v(s_{T+t}), \quad t = 0, \ldots, n.$$

In the above equation, we use the cumulative discounted reward from time $T+t$ to $T+n$ as our approximation of the state-value function at (s_{T+t}, a_{T+t}). The approximation of the value function is simply $\hat{V}(s_{T+t}) = V_v(s_{T+t})$. As a result, by treating $\hat{A}(s_{T+t}, a_{T+t})$ as a constant, the negative gradient of L_{actor} is a stochastic version of the policy gradient multiplied by a constant:

$$\sum_{t=0}^{n} \nabla \log \pi_\theta(s_{T+t}, a_{T+t}) \hat{A}(s_{T+t}, a_{T+t}).$$

Note that we use the negative gradient here since we want to maximize the reward. The critic loss L_{critic} is the squared error of the form

$$\sum_{t=0}^{n} \left[V_{\text{target}}(s_{T+t}) - V_v(s_{T+t}) \right]^2,$$

where the labels V_{target} are defined as

$$V_{\text{target}}(s_{T+t}) = \sum_{i=t}^{n} \gamma^{i-t} r_{T+i}, \quad t = 0, \ldots, n.$$

Namely, we use the rollout trajectory to get the target value via the true rewards from the environment. We present the details of A2C in Algorithm 1.

3 Implementation and Results

In this section, we present our implementation details and empirical results.

Algorithm 1: Advantage Actor-Critic (A2C)

```
Input: Initial parameters.
Init: Initial state $s_0$. $T \leftarrow 1$. $t \leftarrow 1$.
1  while $T \leq T_{max}$ do
2      $t_{\text{start}} \leftarrow t$.
3      while $t - t_{start} \leq t_{max}$ do
4          Draw action $a_t \sim \pi_{\theta_t}(s_t, \cdot)$.
5          Get reward $r_t$.
6          Observe next state $s_{t+1} \sim P(\cdot|s_t, a_t)$.
7          $T \leftarrow T + 1$.
8          $t \leftarrow t + 1$.
9      end
10     $R \leftarrow V(s_t)$.
11     $L_{\text{actor}} \leftarrow 0$. $L_{\text{critic}} \leftarrow 0$.
12     for $i = t - 1, \dots, t_{start}$ do
13         $R \leftarrow \gamma R + r_i$.
14         $L_{\text{actor}} \leftarrow L_{\text{actor}} - \log \pi_{\theta_t}(s_i, a_i)(R - V_v(s_i))$.
15         $L_{\text{critic}} \leftarrow L_{\text{critic}} + (R - V_v(s_i))^2$.
16     end
17     $L \leftarrow L_{\text{actor}} + W \cdot L_{\text{critic}}$.
18     Gradient descent step to minimize $L$.
19 end
```

3.1 NAC for Long-Run Average Reward

We first use NAC introduced in Algorithm 2. Please refer to NAC_test for the code. Since we are dealing with long-run average reward, random initialization is prone to failure. Hence we first train the policy network using a known stable heuristic policy as the labels in supervised learning. The policy we employ is the tailored base-surge (TBS) policy. A TBS policy $\pi_{r,S}$ is characterized by two parameters $r \in \mathbb{Z}_+$ and $S \in \mathbb{Z}_+$. In each period, the policy always orders r products from R and follows an order-up-to rule from E, where we maintain the express inventory position above S. That is to say, for all time step t, we set

$$\begin{aligned} q_t^r &= r, \\ q_t^e &= \max\left(0, S - \widehat{I}_t\right), \end{aligned}$$

where $\hat{I}_t := I_t + \sum_{i=t-L_e}^{t-1} q_i^e + \sum_{i=t-L_r}^{t-L_r+L_e} q_i^r$ is the so-called inventory position, which corresponds to the net inventory at the start of period t plus all orders to be received in periods $t, \dots, t + L_e$. The optimal TBS policy can be computed by solving a convex program, as showed in the paper [JSS15].

We use two separate neural networks for the actor and the critic, since we do not know how to perform the natural gradient update if we only use one network. To stabilize the gradient, we perform the average reward update and the TD update in Algorithm 2 for multiple steps before updating the parameters. The final output policy produces similar average reward as the initial policy. We observe that the algorithm is extremely sensitive to the choice of step sizes. With large step sizes, the value function blows up in just a few episodes since, as pointed out in the work [FHM18], the overestimation error occurs easily in function approximation and it is detrimental to training. With small step sizes, the parameters seem to oscillate around the initial point probably because the optimization landscape is too flat to escape. In conclusion, the long-run average reward setting is challenging because of several issues:

1. The assumption of the Markov chain being irreducible and aperiodic is hard to satisfy.
2. The value function is relative in the sense that its accuracy relies on the accuracy of the estimate of the long-run average reward, making it highly unstable and extremely difficult to learn.
3. We currently do not know how to adaptively tune the step sizes to boost learning.

We believe the method of using supervised learning to initialize provides a good starting point. However, the step sizes proposed in the work [BSGL09] cannot yield much improvement in our experiment. We suggest that finding a good way to adaptively update the step sizes is crucial in this learning situation, which is left for future work.

3.2 A2C for Discounted Reward

Due to the difficulties discussed in the previous subsection, we turn to algorithms in the discounted setting since the discount factor provides more stability. We implement A2C as in Algorithm 1. Please refer to A2C_test for the code. Similarly to what we did before, the initial parameters of the neural network are obtained by supervised learning. Henceforth, we call the policy yielded by our neural network the NN policy.

For numerical experiments, we consider the demand distribution $D \sim \text{Pois}(\lambda)$, where $\lambda \in \{5, 10\}$. We fix the express lead time $L_e = 1$, the costs $c_r = 100, c_e = 105$ and holding cost $h = 1$, and vary the lead-time difference $\Delta L = L_r - L_e \in \{2, 10\}$ and backorder cost $b \in \{19, 99\}$. The discount factor is set as $\gamma = 0.99$. In every test case, we choose an arbitrary sub-optimal TBS policy, and initialize the neural network based on it. After a short period of stochastic gradient descent steps, the initial NN policy is sufficiently close to the TBS policy of choice, which already gives a stable long-run average reward. We compare the performance of the optimal TBS policy, the initial NN policy and the output NN policy computed by A2C in Table 1 and Table 2. Each entry of the table records the mean and standard deviation of the average rewards obtained in 100 simulations, with 5000 time steps in each simulation. The results show that the initialization scheme often gives us a policy with a high variance, and the A2C algorithm reduces the variance while also pushing the average reward closer to the optimal TBS policy.

At each episode of A2C, we use ADAM to update the parameters of the neural network. The default learning rate we use is 0.01. But when doing experiments, we found that the learning rate 0.01 can sometimes make the algorithm highly unstable. Tuning it to 0.001, while resulting in a much slower convergence rate, can solve the problem. Another hyperparameter that can largely affect the performance of A2C is the weight W of the critic loss. We choose $W \in \{0.1, 1, 10\}$ accordingly under different configurations of the environment.

$(\Delta L, b)$	$(2, 99)$	$(10, 99)$	$(2, 19)$	$(10, 19)$
TBS	-515.31 ± 3.55	-516.67 ± 3.14	-515.21 ± 3.48	-516.48 ± 3.29
initial NN	-572.23 ± 40.91	-579.88 ± 41.55	-539.78 ± 3.25	-659.57 ± 3.52
NN	-539.58 ± 3.66	-547.30 ± 5.30	-520.80 ± 3.15	-553.76 ± 3.10

Table 1: $\lambda = 5$.

$(\Delta L, b)$	$(2, 19)$	$(10, 19)$
TBS	-1016.55 ± 7.30	-1019.55 ± 6.73
initial NN	-1095.06 ± 67.06	-1116.21 ± 60.64
NN	-1047.78 ± 4.92	-1040.43 ± 5.24

Table 2: $\lambda = 10$.

Finally, we present some visualizations under a specific setting: $\lambda = 10$, $\Delta L = 10$ and $b = 19$.

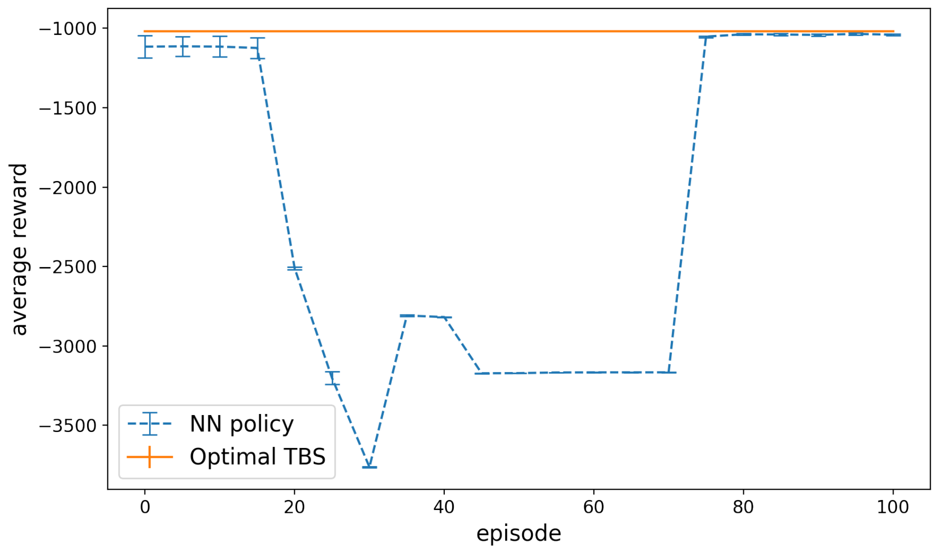
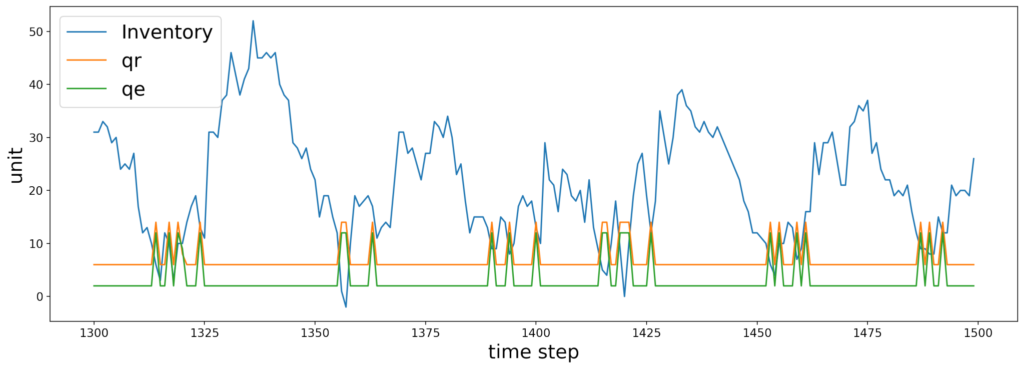
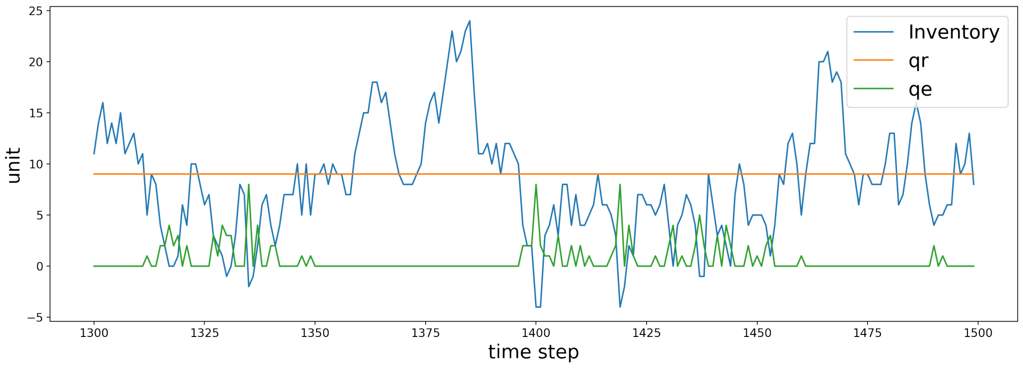


Figure 1: Learning curve obtained from A2C for the dual-sourcing inventory problem. The blue dashed line shows the performance of the NN policy every 5 episodes. The orange solid line shows the performance of the optimal TBS policy.

In Figure 1, we show the learning curve from our implementation of A2C, as the blue dashed line. The variance of the policy is represented by the vertical bar. We also plot the performance of the optimal TBS policy as the orange line. At first glance, the average reward travels along a bizarre curve, on which it first dips down and then shoots up. Our interpretation is that the algorithm first escapes a local minimizer, the initial point, and then converges to a better local minimizer, the output. It is clear that the output NN policy provides an average reward that is closer to the one using the optimal TBS policy, comparing to the initial NN policy. Furthermore, the output NN policy has a much smaller variance than the initial one. That is to say, our algorithm indeed improves upon the initial policy and it has similar performance to the optimal TBS policy.



(a) Output NN policy



(b) Optimal TBS policy

Figure 2: Comparison of the output NN policy and the optimal TBS policy on the dual-sourcing inventory problem. We excerpt a period $[1300, 1500]$ from the simulation. The blue line indicates changes in the inventory level while the orange and the green line represent the actions taken, according to the two policies. Specifically, the orange line is the order placed with the regular supplier at each time step and the green line is the order placed with the express supplier.

In Figure 2, we show two plots depicting the states and actions made by the output NN policy and the optimal TBS policy, respectively. In our problem, the state should include the inventory level as well as the pipeline vector. We omit the pipeline vector for simplicity. As introduced in Subsection 3.1, the TBS policy orders a fixed number of products from the regular supplier at all time steps, which corresponds to the horizontal orange line in Figure

2b. The order placed with the express supplier acts as a regulating force to account for the surge of the demands. In Figure 2b, the green line stays at 0 if the blue line is way above 0 and the green line soars up if the blue line is close to or below 0, which is exactly how TBS works. When the inventory level is too high, we do not need to order more products. When the inventory is almost up or when we have backorders, we must order more products from the express supplier. The output NN policy has a slightly different dynamic. The two orders placed with the regular supplier and the express one both adapt to the current inventory level. When the inventory level is high, we order a small amount of product from the two suppliers. When the inventory level is low or when we have backorders, we increase the orders with both suppliers. The output NN policy seems to inherit some of the properties possessed by the TBS policy, which is not surprising since our initialization is done by approximating a TBS policy. The regular order has a base order of 6 while the express order has a base order of 2. The surge in the demands is handled by both regular and express orders, as opposed to only the express order in the TBS policy.

Appendices

A Natural Actor-Critic

Recall that the long-run average reward is

$$J(\pi) = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[r_{t+1}|\pi].$$

We assume under any policy π, the MDP is irreducible and aperiodic. For a policy π, define the state value function $V^\pi(s)$ and the state-action value function $Q^\pi(s,a)$ according to

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} (r_{t+1} - J(\pi)) \mid s_0 = s, \pi\right], \quad \forall s \in \mathcal{S},$$
$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} (r_{t+1} - J(\pi)) \mid s_0 = s, a_0 = a, \pi\right], \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A},$$

which essentially calculate the expected differential reward. The advantage function is defined as

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s).$$

We parameterize the value function to be $\hat{V}_{v_t}(s) = v_t^\top f_s$ with $v_t \in \mathbb{R}^{d_2}$, where $f_s \in \mathbb{R}^{d_2}$ is the feature vector for state s. The critic learns the value function by using the temporal difference (TD)

$$\delta_t = r_{t+1} - \hat{J}_{t+1} + \hat{V}_{s_{t+1}} - \hat{V}_{s_t},$$

where $\hat{V}_{s_i}$ is an unbiased estimate of the value function in states $s_i, i = t, t+1$ and $\hat{J}_{t+1}$ is an unbiased estimate of the average reward. The average reward estimate is usually updated according to

$$\hat{J}_{t+1} = \hat{J}_t + \xi_t \left(r_{t+1} - \hat{J}_t \right) = (1 - \xi_t) \hat{J}_t + \xi_t r_{t+1},$$

where $\xi_t > 0$ is the step size. We can show that the TD is an unbiased estimate of the advantage function. Namely, we have

$$\mathbb{E}[\delta_t | s_t, a_t, \pi] = A^\pi(s_t, a_t). \tag{6}$$

Then we can update the parameter v as follows:

$$v_{t+1} = v_t + \alpha_t \delta_t f_{s_t},$$

where $\alpha_t > 0$ is the step size for the critic.

We parameterize the policy $\pi = \pi_\theta$ with $\theta \in \mathbb{R}^{d_1}$. The actor updates by following the gradient direction. Assume the policy function π is C^1-smooth in θ. Classic policy gradient theorem says that

$$\nabla J(\pi) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \nabla \pi(s, a) \left(Q^\pi(s, a) - b(s) \right), \tag{7}$$

where d^π denotes the stationary distribution of the MDP under policy π and b is any baseline function. The natural gradient $\tilde{\nabla} J(\pi)$ is obtained by multiplying the regular gradient $\nabla J(\pi)$ by the inverse Fisher information matrix (FIM) of the policy,

$$\tilde{\nabla} J(\pi) = G(\theta)^{-1} \nabla J(\pi).$$

The FIM $G(\theta)$ has the expression

$$G(\theta) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi(s, a) \nabla \log \pi(s, a) \nabla \log \pi(s, a)^\top,$$

which is clearly positive definite. Unfortunately, we do not have access to the true action-value function in the gradient formula (7). Nevertheless, we may use an approximation instead. We form the linear approximation of the state-value function as $\hat{Q}^\pi_w(s, a) = w^\top \psi_{sa}$ where ψ_{sa}'s are the compatible features defined as $\psi_{sa} = \nabla \log \pi(s, a)$. Thus, compatibility holds: $\nabla_w \hat{Q}^\pi_w(s, a) = \nabla \log \pi(s, a)$. We set out to find the parameter that minimizes the mean squared error

$$\mathcal{E}^\pi(w) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi(s, a) \left[Q^\pi(s, a) - w^\top \psi_{sa} - b(s) \right]^2,$$

where $b(s)$ is any baseline. We can show that the optimum $w^\star = \arg\min_w \mathcal{E}^\pi(w)$ is independent of baseline $b(s)$. Hence we can first minimize the variance by plugging in $w^\star$ and then consider $\mathcal{E}^\pi(w^\star)$ as a function of $b(s)$. Subsequently, the optimum $b^\star = \underset{b}{\arg\min} \mathcal{E}^\pi(w^\star)$ is exactly the value function V^π. In this sense, the variance of the estimated policy gradient is minimized by using the value function as the baseline. Furthermore, the linear function ${w^\star}^\top \psi_{sa}$ is the least-squared optimal parametric representation for the advantage function. We can leverage this result by setting $b(s) = V^\pi(s)$ in equation (7). Combining with the fact that TD δ_t is an unbiased estimate of the advantage function, showed in equation (6), we obtain the update of θ using natural gradient in the form $\tilde{\nabla} J(\pi) = G(\theta)^{-1}\nabla J(\pi)$,

$$\theta_{t+1} = \theta_t + \beta_t G(\theta_t)^{-1}\delta_t \psi_{s_t a_t}, \tag{8}$$

where $\beta_t > 0$ is the step size for the actor.

We present one version of the natural actor-critic algorithm in Algorithm 2, which incorporates the advantage parameters w. As discussed before, the compatible feature $w^\top \psi_{sa}$ is an approximation of the advantage function. We want w to minimize the mean squared error

$$\mathcal{E}^\pi(w) = \mathbb{E}_{s\sim d^\pi, a\sim\pi}\left[\left(w^\top\psi_{sa} - A^\pi(s,a)\right)^2\right].$$

The gradient of $\mathcal{E}^\pi(w)$ is

$$\nabla_w \mathcal{E}^\pi(w) = 2\sum_{s\in\mathcal{S}} d^\pi(s)\sum_{a\in\mathcal{A}}\pi(s,a)\left[w^\top\psi_{sa} - A^\pi(s,a)\right]\psi_{sa}.$$

We use $2\left(\psi_{s_t a_t}\psi_{s_t a_t}^\top w - \delta_t \psi_{s_t a_t}\right)$ to approximate the above gradient. Hence the parameter w can be updated following this direction. The natural gradient used by the actor is $\tilde{\nabla} J(\theta_t) = w_{t+1}$. It is later proved in Lemma 8, [BSGL09] that $w_t \to G(\theta)^{-1}\mathbb{E}\left[\delta_t^{\pi_\theta}\psi_{s_t a_t}\right]$ as $t\to\infty$ with probability one. As a result, even if our update of θ is different from the one introduced earlier in equation (8), the convergence analysis shows that the trajectory still moves in the direction of the natural gradient. The step sizes are chosen according to

$$\begin{aligned}&\sum_t \alpha_t = \sum_t \beta_t = \infty, \quad \sum_t \alpha_t^2, \sum_t \beta_t^2 < \infty \\ &\beta_t = o(\alpha_t).\end{aligned} \tag{9}$$

Therefore, β_t tends to 0 faster than α_t, meaning that critic converges faster than the actor. The average reward update employs step sizes $\xi_t = c\alpha_t$, where $c > 0$ is a constant.

Algorithm 2: Natural Actor-Critic Algorithm with Advantage Parameters

```
   Input: Initial parameters $\theta_0, v_0, w_0$. Step sizes $\alpha = \alpha_0, \beta = \beta_0, \xi = c\alpha_0$.
   Init: Initial state $s_0$.
1  for $t = 0, 1, \ldots,$ do
2      Draw action $a_t \sim \pi_{\theta_t}(s_t, \cdot)$.
3      Observe next state $s_{t+1} \sim P(\cdot|s_t, a_t)$.
4      Get reward $r_{t+1}$.
5      Average reward update: $\hat{J}_{t+1} = (1-\xi_t)\hat{J}_t + \xi_t r_{t+1}$.
6      TD update: $\delta_t = r_{t+1} - \hat{J}_{t+1} + v_t^\top f_{s_{t+1}} - v_t^\top f_{st}$.
7      Critic update: $v_{t+1} = v_t + \alpha_t \delta_t f_{s_t}$.
8                     $w_{t+1} = \left[I - \alpha_t \psi_{s_t a_t}\psi_{s_t a_t}^\top\right] w_t + \alpha_t \delta_t \psi_{s_t a_t}$.
9      Actor update: $\theta_{t+1} = \theta_t + \beta_t w_{t+1}$.
10 end
```

References

[AM10] Gad Allon and Jan A. Van Mieghem. Global dual sourcing: Tailored base-surge allocation to near- and offshore production. *Management Science*, 56(1):110–124, 2010.

[BSGL09] Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

[FHM18] Scott Fujimoto, H. V. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *ArXiv*, abs/1802.09477, 2018.

[GBLB12] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[JSS15] Ganesh Janakiraman, Sridhar Seshadri, and Anshul Sheopuri. Analysis of tailored base-surge policies in dual sourcing inventory systems. *Management Science*, 61(7):1547–1561, 2015.

[KKM11] Steffen Klosterhalfen, Gudrun Kiesmüller, and Stefan Minner. A comparison of the constant-order and dual-index policy for dual sourcing. *International Journal of Production Economics*, 133(1):302–311, 2011. Leading Edge of Inventory Research.

[MBM$^+$16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[VSW08] Senthil Veeraraghavan and Alan Scheller-Wolf. Now or later: A simple policy for effective dual sourcing in capacitated systems. *Operations Research*, 56(4):850–864, 2008.

[XG18] Linwei Xin and David A. Goldberg. Asymptotic optimality of tailored base-surge policies in dual-sourcing inventory systems. *Management Science*, 64(1):437–452, 2018.