ORIE 6590 Final Report

Hemeng Li, Ruifan Yang

May 16, 2021

1 Introduction

In this project, we consider the dual-sourcing inventory systems, in which the company can purchase a material from a regular supplier with lower cost, or from a faster supplier at a higher cost under emergency situations. At each timestep, a demand is realized and the company suffered a holding cost and a backorder cost. The goal of the problem is to find a policy that minimize the long run average cost (consist of holding cost, backorder cost, and supply costs) of the system.

We implement a `gym` environment for the dual-sourcing inventory system and design a Proximal Policy Optimization(PPO) algorithm [4] with a Actor-Critic framework. Instead of using a random initialization, we trained a TBS-alike network to used as the initial policy. We run experiments in two dual-sourcing system with different configuration and compare our policy with Tailored Base-Surge (TBS) Policy introduced by Allon and Mieghem [1] and the policy trained by the other group.

Our algorithm always improve the TBS-alike initial policy but the trained policy does not outperform the optimal TBS policy (found through a grid search). After visualizing our initial and trained policy through a heatmap, we think the performance of the policy might be limited because our neural net structure is too simple. Training the initial policy on a more complex network might be able to improve the performance for our algorithm. Other potential modification for the algorithms includes reward normalization and Adam annealing, as suggested in Engstrom et al. [2].

1.1 Dual-sourcing Inventory Model Description

In this subsection, we will formally define our dual-sourcing inventory problem following [5].

Let $\{D_t\}_{t\in(-\infty,\infty)}$ and $\{D'_t\}_{t\in(-\infty,\infty)}$ be mutually independent sequences of i.i.d. demand realizations in which $D_t, D'_t \sim \text{Poisson}(\lambda)$. Let $L_R \geq 1$ be the fixed lead time for the regular source (R), and $L_E \geq 0$ be the fixed lead time for express source (E) with $L_R > L_E + 1$. In addition, let c_R and c_E be the unit purchase price from regular and express source, respectively, with $c_E - c_R > 0$, and let h and b the unit holding and backorder costs, respectively. Let I_t denote the inventory at the beginning of period t, and let q_t^R, q_t^E be the order placed from regular and express source at the beginning of period t, respectively. Note, because of lead time, the orders received at period t from regular and express sources are the orders placed L_R and L_E periods ago, respectively, i.e. $q^R_{t-L_R}$ and $q^E_{t-L_E}$.

Let $\hat{G}$ be an independent Geometric random variable with $\mathbb{P}(\hat{G} = k) = 1/2^k$ for all $k \geq 1$. We assume a random initial inventory,

$$I_1 = -\sum_{i=1}^{\hat{G}} D'_{-i}$$

Let $q_t^R = q_t^E = 0$ for all $t \leq 0$. Then at time period $t = 1, 2, \ldots$, a sequence of events happen in the following order:

- On-hand inventory I_t is observed.
- Ordering decisions q_t^R and q_t^E are made.
- New inventory $q_{t-L_R}^R$ and $q_{t-L_E}^E$ is received and added to the current inventory.
- Demand D_t is realized, and filled with current inventory (back-order allowed).
- Costs for period t, denote C_t, are incurred.

Note given that back-orders are allowed, the on-hand inventory may be negative, and it is updated according to

$$I_{t+1} = I_t + q_{t-L_R}^R + q_{t-L_E}^E - D_t$$

Let $G(y)$ denote the sum of holding and back-order costs with y inventory. Then

$$G(y) = h \cdot \max(y, 0) + b \cdot \max(-y, 0)$$

We assume that the company pays for the order L_E periods after the order was placed. Then for all $t \geq L_E + 1$, the cost incurred at period t is the sum of holding and back-order costs and the ordering costs incurred for orders placed in period $t - L_E$, i.e.

$$C_t = c_R q_{t-L_E}^R + c_E q_{t-L_E}^E + G(I_t + q_{t-L_R}^R + q_{t-L_E}^E - D_t)$$

Note when $1 \leq t \leq L_E$, we have $t - L_E \leq 0$. Then C_t becomes just the sum of holding and back-order costs since we assumed $q_t^R = q_t^E = 0$ for all $t \leq 0$.

Now we want to formalize the admissible policies. An admissible possible π consists of a sequence of measurable functions, i.e. $\pi = \{f_t^\pi : t \geq 1\}$. Given $D_t \in \mathbb{Z}_+$, we consider $f_t^\pi : \mathbb{Z}_+^{L_E+L_R} \times \mathbb{Z} \to \mathbb{Z}_+^2$. To be more specific,

$$(q_t^R, q_t^E) = f_t^\pi(q_{t-L_R}^R, \ldots, q_{t-1}^R, q_{t-L_E}^E, \ldots, q_{t-1}^E, I_t)$$

Let $C(\pi)$ denote the long-run average cost incurred by a policy π. Our goal is to find a policy that minimizes $C(\pi)$, in which

$$C(\pi) \coloneqq \limsup_{T \to \infty} \frac{1}{T} \sum_{t=L_E+1}^{T} \mathbb{E}[C_t^\pi]$$

Note without loss of generality, we start the relative sum at $t = L_E + 1$ instead of $t = 1$.

1.2 MDP Parameters

In this subsection, we consider the discrete-time MDP equivalent to the model we described in Section 1.2.

Let $q_t = (q^R_{t-L_R}, \ldots, q^R_{t-1}, q^E_{t-L_E}, \ldots, q^E_{t-1}) \in \mathbb{Z}_+^{L_R+L_E}$ be the pipeline vector of orders placed, but not yet delivered, at the beginning of period t. In addition, let (s_t, a_t) be the state-action pair at time period t for all $t \geq 1$. The discrete-time MDP has state space $S = \mathbb{Z}_+^{L_R+L_E} \times \mathbb{Z}$, where

$$s_t = (q_t, I_t) = (q^R_{t-L_R}, \ldots, q^R_{t-1}, q^E_{t-L_E}, \ldots, q^E_{t-1}, I_t).$$

The action space is $\mathcal{A} = \mathbb{Z}_+^2$. For all actions $a = (a_R, a_E) \in \mathbb{Z}_+^2$, a_R is the amount to order from regular source, and a_E is the amount to order from express source. We define the reward function $r\colon S \times \mathcal{A} \to \mathbb{R}$ as the negative of the expected cost incurred, i.e.

$$r(s_t, a_t) = -\mathbb{E}[C_t] = -c_R q^R_{t-L_E} - c_E q^E_{t-L_E} - \mathbb{E}\left[G(I_t + q^R_{t-L_R} + q^E_{t-L_E} - D_t)\right]$$

Now we formalize the transition probability as follows:

$$\begin{aligned}\mathbb{P}\left(s_{t+1} \mid (s_t, a_t)\right) &= \mathbb{P}(\underbrace{(q^R_{t-L_R+1}, \ldots, q^R_t, q^E_{t-L_E+1}, \ldots, q^E_t, I_{t+1})}_{=s_{t+1}} \mid \underbrace{(q^R_{t-L_R}, \ldots, q^R_{t-1}, q^E_{t-L_E}, \ldots, q^E_{t-1}, I_t)}_{=s_t}, \underbrace{(q^R_t, q^E_t)}_{=a_t}) \\ &= \mathbb{P}\left(D_t = I_t + q^R_{t-L_R} + q^E_{t-L_E} - I_{t+1}\right),\end{aligned}$$

where $D_t \sim \text{Poisson}(\lambda)$. And $\mathbb{P}(s'|(s,a)) = 0$ for all other situations that does not fit above.

2 Existing Approach

Even though the dual-sourcing inventory model has been studied for a long time, the structure of the optimal policy still remains poorly understood except when the system is under some very specific conditions. As a result, researchers have been shift their focuses to construct various heuristic policies. In this section, we will introduce an existing heuristic policy that has generally perform pretty well.

2.1 Tailored Base-Surge (TBS) Policy

The TBS policy was first proposed and analyzed by [1]. Before we formally introduce the family of TBS policies, we firstly define so-called the *expedited inventory position*, denote $\widehat{I}_t$, for all time $t \geq 1$ as:

$$\widehat{I}_t \coloneqq I_t + \sum_{k=t-L_E}^{t-1} q^E_k + \sum_{k=t-L_R}^{t-L_R+L_E} q^R_k$$

Intuitively, the *expedited inventory position* corresponds to the on-hand inventory at the beginning of period t plus the orders that are placed before period t and can be received in periods $t, \ldots, t + L_E$.

Now we formally define the family of TBS policies. Each TBS policy, denote $\pi_{r,S}$, has parameter (r, S) in which $r, S \in \mathbb{Z}_+$. At each period, $\pi_{r,S}$ places a constant order r from regular source and orders enough from express source to raise the expedited inventory position

to S. If the expedited inventory position at period t is above S, then $\pi_{r,S}$ orders nothing from the express source. Formally, under $\pi_{r,S}$, for all period $t \geq 1$ we have:

$$q_t^R = r$$
$$q_t^E = \max(0, S - \widehat{I}_t)$$

3 Proximal Policy Optimization (PPO)

In this section, we develop a variant of Proximal Policy Optimization the finding the near optimal policies for dual-sourcing inventory problem.

3.1 Policy Gradient Methods

Let $\pi_\theta(a \mid s)$ be a stochastic policy modeled with a parameterized function respect to θ (usually a neural network). Policy gradient methods learn to maximize the expected reward following a parameterized policy based on the gradient of some performance measure $J(\theta)$.

A standard objective used to generate the gradient estimator is

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta \left(a_t \mid s_t \right) \hat{A}_t \right],$$

where $\hat{A}_t$ is an estimator of the advantage function at timestep t, and $\hat{\mathbb{E}}_t \left[\dots \right]$ is the empirical average over a finite batch of samples.

3.2 Clipped Surrogate Objective

In Policy Gradient, we use samples from the current policy to compute the policy gradient, so we need new samples for each gradient we compute. To reuse sample, we can use a surrogate objective using importance sampling:

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right],$$

where CPI refers to conservative policy iteration [3], and $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. It's not hard to see that the gradients for the original and surrogate objective are the same:

$$\nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right)|_{\theta_{\text{old}}} = \frac{\nabla_\theta \pi_\theta \left(a_t \mid s_t \right)|_{\theta_{\text{old}}}}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} = \nabla_\theta \left(\frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} \right) \Bigg|_{\theta_{\text{old}}}$$

However, maximizing L^{CPI} would result in large policy updates and unstable policies without a constraint. Hence, Schulman et al.[4] proposed a clipped surrogate objective

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), \epsilon \right) \hat{A}_t \right) \right],$$

where $\text{clip}\left(r_t(\theta), \epsilon\right) = \begin{cases} x & \text{if } x \in [1-\epsilon, 1+\epsilon] \\ 1+\epsilon & \text{if } x > 1+\epsilon \\ 1-\epsilon & \text{if } x < 1-\epsilon \end{cases}$.

For a single timestep t, if $A > 0$,

$$\min(rA, \text{clip}(r, \epsilon)A) = \begin{cases} rA & \text{if } r \leq 1+\epsilon \\ (1+\epsilon)A & \text{if } r > 1+\epsilon \end{cases};$$

if $A < 0$,

$$\min(rA, \text{clip}(r, \epsilon)A) = \begin{cases} rA & \text{if } (1-\epsilon) \leq 1-\epsilon \\ rA & \text{if } r > 1-\epsilon \end{cases}.$$

Notice that $L^{CLIP}(\theta) = L^{CPI}(\theta)$ when $r_t(\theta)$ is close to 1 (i.e., when θ is close to θ_{old}).

3.3 Actor-Critic Framework

In addition to learning the policy, Actor-Critic methods learn approximation to both policy and value functions. It consists of two models:

- An actor which updates the policy parameter θ corresponding to the policy $\pi_\theta(a \mid s)$;
- And a critic which updates a value function parameter ϕ usually corresponding to a state-value function $V_\phi(s)$ or a action-value function $Q_\phi(a \mid s)$.

In this project, we use TD(1) to estimate the value function. At each iteration, let the estimated value function be

$$\hat{V}^{\pi_k}\left(s_t\right) = \sum_{k=t}^{T} \gamma^k r_k.$$

Then, fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| \, T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi\left(s_t\right) - \hat{V}^{\pi_k}\left(s_t\right)\right)^2.$$

where $\mathcal{D}_k$ is the trajectory by running policy π_{θ_k} in the environment.

3.4 Multiple epochs for policy updating

The Clipped Surrogate Objective function allows us to run multiple epoch of gradient ascent on the same sample while maintaining meaningful policy updates.

In each iteration, PPO runs the policy using Q parallel actors, collecting T timesteps of date. Then we construct the clipped surrogate loss on these QT timesteps of data, and optimize it with minibatch Stochastic Gradient Ascent for N epochs. The full PPO algorithm is described in Algorithm 1 below.

Algorithm 1: PPO

Input : initial policy parameter θ_0, initial value function parameter ϕ_0
Output: final policy parameter θ_K

for *iteration* $k = 0, 1, 2, \ldots K-1$ **do**
  for *actor* $q = 1, 2, \ldots, Q$ **do**
    Collect trajectory $\mathcal{D}_{k,q} = \{s^{(0,q)}, a^{(0,q)}, s^{(1,q)}, a^{(1,q)}, \ldots, s^{(T,q)}, a^{(T,q)}\}$ by running policy π_{θ_k} in environment for T timesteps
    Compute estimated value function $\hat{V}^{\pi_k}(s^{(t,q)})$
    Compute estimated advantage function $\hat{A}^{\pi_{\theta_k}}(s^{(t,q)}, a^{(t,q)})$ based on current value function V_{ϕ_k}
  end
  Update the policy parameter by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \sum_{q=1}^{Q} \sum_{t=0}^{T} \min\left(r^{(t,q)}(\theta) \hat{A}^{\pi_{\theta_k}}\left(s^{(t,q)}, a^{(t,q)}\right), \quad \text{Clip}\left(r^{(t,q)}(\theta), \epsilon\right) \hat{A}^{\pi_{\theta_k}}\left(s^{(t,q)}, a^{(t,q)}\right) \right)$$

  using stochastic gradient ascent with Adam, where $r^{(t,q)}(\theta) = \frac{\pi_\theta\left(a^{(t,q)}|s^{(t,q)}\right)}{\pi_{\theta_k}\left(a^{(t,q)}|s^{(t,q)}\right)}$
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \sum_{q=1}^{Q} \sum_{t=0}^{T} \left(V_\phi\left(s_t\right) - \hat{V}^{\pi_k}(s^{(t,q)}) \right)^2$$

  using stochastic gradient descent with Adam
end

4 Implementation Details

In this section, we will discuss some key observations we made during our implementation of the model environment and PPO algorithm.

4.1 Truncated State and Action Space

Even though we would desire to implement the model environment exactly as it is described in Section 1.2, there are some implementation difficulties. Firstly, the `Multidiscrete` space in Python `gym` package requires us to specify an upper bound, i.e. it does not allow infinite state/action space. In addition, the `Multidiscrete` space does not allow negative values while the model itself actually allows back-order (negative values).

One key observation we made is if a policy is sufficiently good, then it will never lead to extremely large order or extremely large inventory (back-orders), i.e. it will keep inventory close to 0, while making smaller orders at a regular frequency. As a result, if a policy can perform well in a bounded state space (with specified maximum inventory/back-orders) and bounded action space (with maximum orders that can be made), it will perform well in

the infinite space settings as it should never escape the original bounded state/action space. With this in mind, we implemented the given model with a truncated state and action space.

With bounded state space, the second problem becomes easy to solve. Let $I_{\max} \in \mathbb{Z}_+$ denote the max inventory (back-order) allowed. We can then shift the inventory from $\{-I_{\max}, \ldots, I_{\max}\}$ to $\{0 \ldots, 2I_{\max}\}$ to avoid having negative values in a the `Multidiscrete` space.

4.2 Initial Policy

The other important observation is that initial policy really affects the performance of PPO algorithm. Good initial policy generally leads to better results.

In order to get a sufficiently good initial policy, we firstly sample a subset of the state space, and generate labels (action) using a deterministic TBS policy as discussed in Section 2.1 to create a training dataset. Randomly shuffle the training set and group data into mini batches, we then train a feed forward neural network with the generated training dataset with batch gradient descent to get the initial policy.

In Section 5.2, we will present the result of PPO with or without initialization.

5 Experimental Results

In this section, we discuss the experimental results. We will first introduce the two set of model parameters we used to experiment our algorithm with. We present the results of our implemented PPO algorithms and compare it with TBS policies and algorithms from other group. Finally we will visualize the policy generated to see how it works.

5.1 Instances Set-up

To set up the experiment, we will have to specify the model parameters. In particular, we consider the following two configurations.

	L_R	L_E	c_R	c_E	h	b	λ	m_A	m_I	starting state
Config 1	3	1	100	105	1	99	5	20	1000	[0,0,0,0,0]
Config 2	3	1	100	105	1	19	10	20	1000	[0,0,0,0,0]

Table 1: Two different model parameters set-up.

Note m_A indicates the maximum number of orders we can make (from either sources) and m_I indicates the maximum number of inventory (back-orders) allowed. Instead of starting with a random state as described in Section 1.2, we will just assume we start with no incoming orders and 0 inventory.

5.2 Numerical Results

For both configurations, we generate an initial policy by training with the optimal TBS policy ($r = 4, S = 15$ for Config 1 and $r = 9, S = 15$ for Config 2). We then use it as the

initial policy to feed into PPO algorithm. We summarize the results in Table 2.

	initial	PPO	PPO (no init)	A2C	Optimal TBS
Config 1	-607	-544	-3473	-540	-517
Config 2	-1113	-1054	-4775	-1048	-1019

Table 2: Average reward of different policies.

As we observed from Table 2, initialization drastically improve the performance. Given a sufficiently good initial policy, PPO algorithm were able to improve on it and produce a better policy. In addition, the performance of PPO and A2C algorithms is pretty close with PPO having slighly worse average reward. However, both algorithms failed to beat the optimal TBS policy.

We also ran a simulation of 1000 time steps under both configurations using initial, PPO and optimal TBS optimal. It is clear from Figure 1 that all three policies converges in the sense that the average reward stabilizes after certain steps.

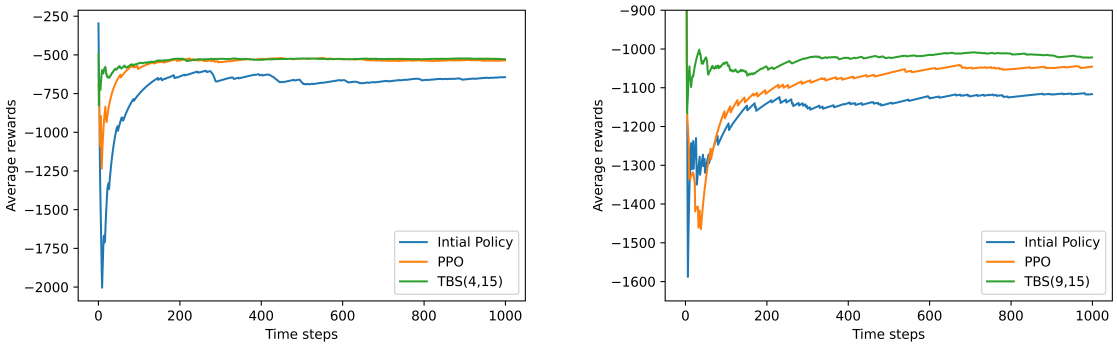

Figure 1: Average rewards simulation of intial, PPO, and optimal TBS policies under Config1 (left) and Condif2(right) for 1000 time steps.

5.3 Policy Visualization

We create some heatmaps to visualize the different policies. The figures below plot the action for express source (that is, the probability distribution for the number of order placed from express source) following different policies under configuration 1 and state space $[4, 4, 4, 4, x]$ where $x \in [-30, 30]$ is the current inventory. We can see that the initial policy almost always place no order from the express source, and the PPO-trained policy placed 19 with higher probability when the inventory is low.

6 Conclusion

In this final project, we investigate the dual-sourcing inventory problem and implement model environment and PPO algorithm to try to solve this problem with deep reinforcement

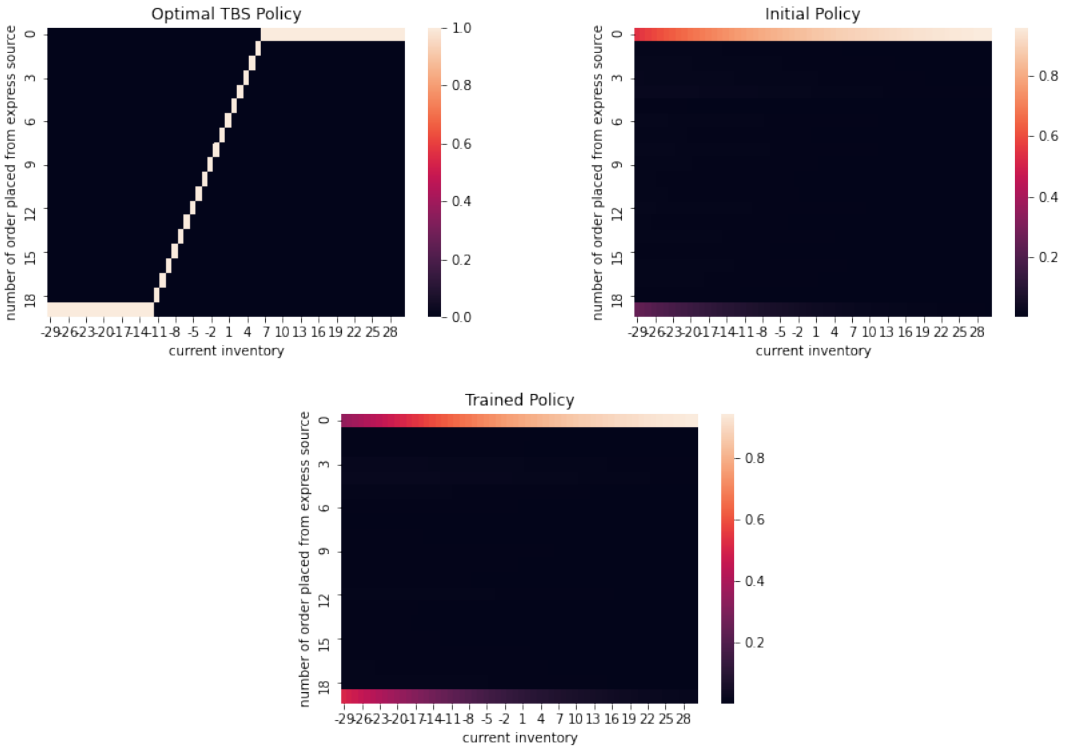

Figure 2: Number of order placed from express source following optimal TBS policy, initial policy, and PPO policy

learning approach. The policy produced by our PPO algorithm is sufficiently good, but still not as good as some of the existing heuristic approach, TBS policy.

Some potential future work might include exploring how neural net structure of our policy can affect the result of the PPO algorithm for this problem. We only used one linear hidden layer for our PPO policy in our experiment, but it might be the case that more complex neural net would help. In addition, as indicated in [2], implementation details can hugely affect the performance of PPO algorithms. Reward normalization and Adam annealing (adjust the learning rate of critic network as we progress) would be some potential techniques we could use to improve the performance of PPO in the future.

7 Github repository

All our code will be pushed to the our Github repository: `https://github.coecis.cornell.edu/hl2359/6590-project`.

References

[1] Gad Allon and Jan A. Van Mieghem. "Global Dual Sourcing: Tailored Base-Surge Allocation to Near- and Offshore Production". In: *Management Science* 56.1 (2010), pp. 110–

124. DOI: 10.1287/mnsc.1090.1099. eprint: https://doi.org/10.1287/mnsc.1090.1099. URL: https://doi.org/10.1287/mnsc.1090.1099.

[2] Logan Engstrom et al. "Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO". In: *CoRR* abs/2005.12729 (2020). arXiv: 2005.12729. URL: https://arxiv.org/abs/2005.12729.

[3] Sham Kakade and John Langford. "Approximately Optimal Approximate Reinforcement Learning". In: *IN PROC. 19TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING*. 2002, pp. 267–274.

[4] John Schulman et al. "Proximal Policy Optimization Algorithms." In: *CoRR* abs/1707.06347 (2017). URL: http://dblp.uni-trier.de/db/journals/corr/corr1707.html#SchulmanWDRK17.

[5] Linwei Xin and David A. Goldberg. "Asymptotic Optimality of Tailored Base-Surge Policies in Dual-Sourcing Inventory Systems". In: *Management Science* 64.1 (2018), pp. 437–452. DOI: 10.1287/mnsc.2016.2607. eprint: https://doi.org/10.1287/mnsc.2016.2607. URL: https://doi.org/10.1287/mnsc.2016.2607.