Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Optimizing Inventory Management with Reinforcement Learning

Master's Thesis

Tankred Imiolczyk

timiolczyk@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Andreas Plesner, Dr. Benjamin Estermann
Prof. Dr. Roger Wattenhofer

August 16, 2025

Acknowledgements

I would like to express my sincere gratitude to my supervisors, especially Andreas Plesner, and Dr. Benjamin Estermann, for their unlimited support, invaluable guidance, and continuous encouragement throughout this research journey. Their expertise and mentorship have been instrumental in shaping this work.

I am deeply grateful to my corporate advisors from Digitec Galaxus AG, Pius Kriemler and Bernhard Walser, for their generous sharing of domain knowledge, practical insights, and for helping me understand and refine the problem statement. Their industry expertise and real-world perspective have been essential in grounding this research in practical applications.

I also extend my appreciation to Prof. Dr. Wattenhofer for his academic oversight and formal supervision of this thesis.

Abstract

Inventory management remains a central challenge in supply chain operations, where classical heuristic methods often rely on unrealistic assumptions such as stationary demand and deterministic lead times. This thesis investigates Reinforcement Learning-based inventory management with a particular emphasis on pretraining strategies and supplier-side stochasticity. We propose a framework that explicitly models variable supplier lead times, enabling dynamic supplier selection under uncertainty. Furthermore, we systematically evaluate different pretraining methods and their effects on convergence speed, training stability, and policy performance. Our results show that RL agents can match optimal baselines in single-supplier environments and that pretraining, especially behavior cloning from data generated by Mixed Integer Linear Programming accelerates learning and improves robustness when modeling multiple products with the same model. However, multi-supplier experiments reveal a qualitative complexity gap: while agents readily master simple cost differentiation, optimization in cost-gradient environments remains challenging, with incomplete convergence under practical training horizons.

Contents

Acknowledgements i

Abstract ii

1 Introduction 1

1.1 Problem Motivation 1

1.2 Contributions 1

2 Related Work 3

3 Problem Formulation 5

3.1 Mathematical Formulation 5

3.2 MDP Formulation 7

3.3 Stochastic Formulation of Delays 11

4 Methodology 13

4.1 Baseline Methods 13

4.2 Reinforcement Learning Agent 14

5 Evaluation 17

5.1 Ablation Studies 17

5.1.1 Action Space Design 18

5.1.2 Observation Space Design 21

5.1.3 Training Space Design 24

5.2 Single Supplier Setup 27

5.2.1 Training Under Noisy Demand 27

5.2.2 Multi-Product Generalization 30

5.3 Multisupplier Setups 38

5.3.1 Supplier Selection with Homogeneous Cost Structures . . 38

5.3.2 Supplier Selection with Variable Cost Differentials 42

6 Conclusion **48**

Bibliography **50**

A Baseline Environment Configuration **A-1**

B Model Architecture **B-1**

C Additional Plots **C-1**

Chapter 1

Introduction

1.1 Problem Motivation

Inventory management is a fundamental challenge in supply chain management, as it involves designing replenishment policies that effectively balance holding, purchasing, and transportation factors under uncertainty. Over the past decades, classical methods have relied on heuristics—such as (s, Q) policies and fixed reorder points—to keep operations tractable under simplifying assumptions. Despite their widespread adoption and ease of implementation, these methods often rest on assumptions of stationary demand and known lead times that fail to capture the dynamism of real-world supply chains. Such limitations are particularly evident in environments characterized by fluctuating demand patterns, unanticipated supply disruptions, that underscore the complexity of modern operations.

Recent advances in reinforcement learning (RL) have demonstrated considerable promise for addressing complex inventory management problems, as early applications of deep RL methods—such as PPO and A3C—have shown that learned policies can outperform classical heuristics in simplified, single-supplier, single-product settings while even extensions to multi-echelon contexts typically assume fixed, deterministic supplier roles. Moreover, many of these approaches deliberately remove stochastic variability by assuming constant lead times, thus neglecting critical elements like supplier variability and fluctuating demand patterns that are inherent to real-world operations.

1.2 Contributions

To address these challenges, we propose a novel inventory management framework that explicitly incorporates stochasticity, particularly on the supplier side by modeling delays. In contrast to traditional approaches that assume constant lead times and consequently default to the cheapest supplier, our setup allows for the integration of comprehensive supplier data, enabling the model to dynamically learn which supplier to select based on varying operational conditions.

Secondly, we systematically investigate different pretraining strategies for the reinforcement learning agent. Specifically, we compare configurations that pretrain solely the policy network, solely the value network, and a hybrid approach combining both. This analysis aims to quantify the improvement in convergence speed, assess training stability relative to conventional RL methods, and evaluate the overall impact on the performance of the learned policies.

Finally, we benchmark our framework against established baselines to demonstrate its effectiveness in realistic scenarios characterized by supplier variability and fluctuating lead times.

Chapter 2

Related Work

Reinforcement Learning for Inventory Management Early applications of deep reinforcement learning to inventory control focused on simplified settings. Oroojlooyjadid et al. applied DQN to the beer game—a classic supply chain problem—training separate agents for each stage under deterministic demand and lead times [1]. Gijsbrechts et al. extended this work using A3C for stockout costs and dual-sourcing scenarios, with one model learning policies for two different suppliers [2]. However, their dual-sourcing formulation fixes suppliers a priori rather than learning selection policies.

To handle stochastic lead times, Meisheri et al. developed delay-resolved DQN (DRDQN), augmenting the state space with historical actions [3]. However, this work remains restricted to single-supplier assumptions per product.

Differentiable Methods Recent work has explored differentiable simulators to improve sample efficiency. Madeka et al. introduced *DirectBackprop* for gradient-based policy optimization in single-supplier inventory systems [4]. Alvo et al. propose *HDPO (Hindsight Differentiable Policy Optimization)*, a related approach that leverages differentiability of cost functions for policy optimization in supplier networks with multiple warehouses and stores [5]. Both methods assume single upstream suppliers.

Supervised Approaches Qi et al. developed a supervised learning framework that maps inventory state variables (demand history, vendor lead times, and current stock levels) to replenishment quantities using neural networks trained on optimal decisions generated by dynamic programming [6].

Specialized Applications Several studies address specific inventory contexts. De Moor et al. stabilized DQN training for perishable goods using reward shaping [7]. Multi-agent approaches include Moussa et al.'s centralized critic for supply chain coordination [8] and Kotecha et al.'s graph neural networks for topology embedding [9].

Research Gaps Existing RL approaches for inventory management exhibit three key limitations. First, except for [3], methods assume fixed supplier assignments, with dual-sourcing studies pre-specifying two suppliers (cheap and slow versus expensive and fast) sources rather than learning selection policies. Second, most work focuses on deterministic lead times, neglecting supplier reliability variations that drive real-world sourcing decisions. Third, systematic pretraining strategies for RL agents remain unexplored in this domain.

Our work addresses these gaps by incorporating stochastic supplier delays, enabling dynamic supplier selection, and investigating pretraining strategies for improved convergence and performance.

Chapter 3

Problem Formulation

3.1 Mathematical Formulation

We consider a single-product inventory management problem over a discrete time horizon $\mathcal{T} = \{1, \ldots, T\}$ with multiple suppliers/vendors $\mathcal{V} = \{1, \ldots, N\}$. The objective is to determine optimal ordering decisions to maximize the total expected profit, which is defined as revenue minus expected costs (ordering, holding, and stockout costs), under stochastic demand and supplier lead times.

Objective Function:

$$\max_{X_{v,t}} \mathbb{E}\left[\sum_{t=1}^{T}\left(p_t \cdot S_t - b \cdot B_t - \sum_{v=1}^{N}\left[k_{v,t} \cdot \mathbb{1}_{\{X_{v,t}>0\}} + c_{v,t}X_{v,t}\right] - hI_{t+1}\right)\right] \quad (3.1)$$

Decision Variables:

- $X_{v,t} \in \mathbb{Z}_+$: Quantity ordered from vendor v at time t

State Variables (derived from decisions and dynamics):

- $I_t \in \mathbb{Z}_+$: Inventory level at beginning of day t (before demand and order arrivals)
- $S_t \in \mathbb{Z}_+$: Items sold on day t
- $B_t \in \mathbb{Z}_+$: Stockout quantity on day t
- $A_t \in \mathbb{Z}_+$: Total quantity of orders arriving on day t

Parameters:

- d_t: Random demand at time t

- p_t: Selling price at time t
- $k_{v,t}$: Fixed ordering cost from vendor s at time t
- $c_{v,t}$: Unit cost from vendor v at time t
- h: Per-unit holding cost per period
- b: Per-unit stockout cost
- L_v: Lead time for vendor v
- $M_{v,t}$: Maximum order quantity from vendor v at time t

System Dynamics:

For $t = 1, \ldots, T$:

Items sold (limited by available inventory):

$$S_t = \min(d_t, I_t) \tag{3.2}$$

Stockout quantity (unmet demand):

$$B_t = \max(0, d_t - I_t) \tag{3.3}$$

Orders arriving on day t:

$$A_t = \sum_{s=1}^{N} X_{v,t-L_v} \tag{3.4}$$

where $X_{v,t-L_v} = 0$ if $t - L_v < 1$.

End-of-day inventory:

$$I_{t+1} = I_t - S_t + A_t = \max(0, I_t - d_t) + A_t \tag{3.5}$$

Constraints:

- Order Capacity: $0 \leq X_{v,t} \leq M_{v,t}$, $\forall s, t$
- Non-negativity: $I_t, S_t, B_t, A_t \geq 0$
- Initial Conditions: I_1 given

3.2 MDP Formulation

We cast the inventory management problem as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition kernel, r is the reward function, and γ is the discount factor.

State Space

The state at time t, $s_t \in \mathcal{S}$, encodes all information required for decision making:

- I_t: Inventory level at the beginning of period t
- $\mathbf{Q}_t$: Pending orders (vector), i.e., quantities ordered but not yet received, indexed by remaining lead time and aggregated over all suppliers
- $\mathbf{F}_t$: Demand forecast for the next H periods, H being a hyperparameter
- $\hat{\boldsymbol{I}}_t$: Forecasted inventory vector, defined as pending orders minus demand forecast
- For each vendor v:
  - $k_{v,t}$: Fixed cost at time t
  - $c_{v,t}$: Unit cost at time t
  - $L_{v,t}$: Lead time at time t

Formally,

$$s_t = (I_t, \mathbf{Q}_t, \mathbf{F}_t, \hat{\boldsymbol{I}}_t, \{k_{v,t}, c_{v,t}, L_{v,t}\}_{v=1}^{N}) \tag{3.6}$$

The pipeline vector $\mathbf{Q}_t$ is defined as:

$$\mathbf{Q}_t = \{Q_\tau : \text{total quantity arriving in } \tau \text{ periods}, \forall \tau \in \{1, ..., \max_v L_{v,t}\}\} \tag{3.7}$$

where $Q_\tau = \sum_{s=1}^{N} Q_{v,\tau}$ aggregates pending orders across all suppliers for each lead time.

The forecasted inventory vector $\hat{\boldsymbol{I}}_t$ is defined as:

$$\hat{\boldsymbol{I}}_t = \mathbf{Q}_t^{padded} - \mathbf{F}_t^{padded} \tag{3.8}$$

where $\mathbf{Q}_t^{padded}$ and $\mathbf{F}_t^{padded}$ are aligned to have the same length through padding:

- If $|\mathbf{Q}_t| < |\mathbf{F}_t|$: $\mathbf{Q}_t$ is padded with zeros to match the length of $\mathbf{F}_t$
- If $|\mathbf{F}_t| < |\mathbf{Q}_t|$: $\mathbf{F}_t$ is padded with $\bar{F}_t = \frac{1}{|\mathbf{F}_t|} \sum_{i=1}^{|\mathbf{F}_t|} F_{t,i}$ to match the length of $\mathbf{Q}_t$

Action Space

The action at time t, $a_t \in \mathcal{A}$, specifies the order quantity to place with each supplier:

$$a_t = (X_{1,t}, X_{2,t}, \ldots, X_{N,t}) \tag{3.9}$$

where $X_{v,t} \in \mathbb{Z}_+$ is the quantity ordered from vendor v at time t.

We consider two variants of the action space:

1. **Continuous Action Space:**

$$\mathcal{A} = \{(X_1, \ldots, X_N) \mid 0 \leq X_v \leq M_{v,t},\ \forall v\} \tag{3.10}$$

2. **Discrete Action Space:** As an alternative way of framing the action space, we consider a discretized version where the agent predicts multipliers $\{m_v\}_{v=1}^N$ with $m_v \in \{0, 1, 2, \ldots, K_v\}$, and the actual order quantities are computed as:

$$X_{v,t} = \min(m_v \cdot \Delta_v, M_{v,t}) \tag{3.11}$$

The discretized action space is thus:

$$\mathcal{A}_{discrete} = \{(m_1, \ldots, m_N) \mid m_v \in \{0, 1, 2, \ldots, K\},\ \forall v\} \tag{3.12}$$

where K is an upper bound, usually being set to $K = \max_v \lfloor M_{v,t}/\Delta_s \rfloor$ is t℔0pche upper bound across all suppliers and Δ_s is the scaling factor for supplier s.

Reward Function

The reward at time t, $r_t = r(s_t, a_t)$, is defined to align with the mathematical formulation's objective of maximizing expected profit:

$$r_t = p_t \cdot S_t - b \cdot B_t - h \cdot I_{t+1} - \sum_{s=1}^{N} \left[k_{v,t} \cdot \mathbb{1}_{\{X_{v,t}>0\}} + c_{v,t} X_{v,t}\right] \tag{3.13}$$

where:

- p_t: Selling price at time t
- S_t: Items sold in period t
- b: Per-unit stockout cost
- B_t: Stockout quantity in period t
- h: Per-unit holding cost per period

- I_{t+1}: End-of-period inventory
- $k_{v,t}$: Fixed ordering cost for vendor v at time t
- $c_{v,t}$: Unit cost for vendor v at time t
- $X_{v,t}$: Order quantity from vendor v at time t

Reward Function

The reward at time t, $r_t = r(s_t, a_t)$, is defined to align with the mathematical formulation's objective of maximizing expected profit:

$$r_t = p_t \cdot S_t - b \cdot B_t - h \cdot I_{t+1} - \sum_{s=1}^{N} \left[k_{v,t} \cdot \mathbb{1}_{\{X_{v,t}>0\}} + c_{v,t} X_{v,t} \right] \tag{3.14}$$

where:

- p_t: Selling price at time t
- S_t: Items sold in period t
- b: Per-unit stockout cost
- B_t: Stockout quantity in period t
- h: Per-unit holding cost per period
- I_{t+1}: End-of-period inventory
- $k_{v,t}$: Fixed ordering cost for vendor v at time t
- $c_{v,t}$: Unit cost for vendor v at time t
- $X_{v,t}$: Order quantity from vendor v at time t

Additionally, we impose a maximum inventory constraint to prevent unrealistic inventory accumulation. When the agent's inventory exceeds this predetermined limit, a substantial negative penalty is applied to the reward, ensuring realistic inventory management behavior.

Transition Dynamics

Given state s_t and action a_t, the state transitions according to the system dynamics defined in the mathematical formulation:

Order arrivals:

$$A_t = \sum_{s=1}^{N} X_{v,t-L_v} = \sum_{s=1}^{N} Q_{v,1}^{(t)} \tag{3.15}$$

Sales and stockouts:

$$S_t = \min(d_t, I_t + A_t) \tag{3.16}$$

$$B_t = \max(0, d_t - I_t - A_t) \tag{3.17}$$

Next period inventory:

$$I_{t+1} = \max(0, I_t + A_t - d_t) \tag{3.18}$$

Pipeline update:

$$Q_{v,\tau}^{(t+1)} = \begin{cases} Q_{v,\tau+1}^{(t)} & \text{if } \tau < L_v \\ X_{v,t} & \text{if } \tau = L_v \end{cases} \tag{3.19}$$

The agent's objective is to maximize the expected cumulative discounted reward:

$$\max_{\pi} \ \mathbb{E}_{\pi} \left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t) \right] \tag{3.20}$$

where π is the policy and $\gamma \in [0, 1)$ is the discount factor.

In initial experiments, we observed that the agent would drastically accumulate inventory to unrealistic levels. To address this behavior, we introduced a maximum inventory constraint $I_{\max}$ and implement episode truncation whenever the inventory level I_t exceeds this threshold, accompanied by a substantial negative reward penalty to discourage such policies.

3.3 Stochastic Formulation of Delays

We extend the deterministic model to incorporate stochastic delivery delays using a quantile-based approach. Instead of fixed lead times, suppliers now have probabilistic lead times characterized by quantile distributions.

Modified Parameters:

For each vendor v, we replace the deterministic lead time L_v with:

- $\mathbf{q} = [q_1, q_2, \ldots, q_K]$: Lead time quantiles where $0 < q_1 < q_2 < \cdots < q_K < 1$
- $\mathbf{d}_v = [d_{v,1}, d_{v,2}, \ldots, d_{v,K}]$: Corresponding lead time values for vendor v

The lead time distribution satisfies $P(L_v \leq d_{v,k}) = q_k$ for all $k \in \{1, 2, \ldots, K\}$. Compared to the original formulation, the expected lead time $\hat{L}_v$ corresponds to the 50% quantile (median) of the distribution.

Modified State Space:

The state representation becomes:

$$s_t = (I_t, \mathbf{Q}_t, \mathbf{F}_t, \hat{\boldsymbol{I}}_t, \{k_{v,t}, c_{v,t}, \mathbf{d}_{v,t}\}_{v=1}^{N}) \tag{3.21}$$

where the fixed lead times $L_{v,t}$ are replaced by the lead time quantile vectors $\mathbf{d}_{v,t}$.

Modified System Dynamics:

Lead Time Sampling: When order $X_{v,t_{\text{order}}}$ is placed at time t_{order}, the actual lead time is sampled as:

$$\tilde{L}_{v,t_{\text{order}}} = \text{Quantile}^{-1}(u; \mathbf{q}, \mathbf{d}_{v,t_{\text{order}}}) \tag{3.22}$$

where $u \sim \text{Uniform}(0, 1)$.

However, the agent is not informed of any delay until the **Delay Information** day $\tilde{I}_{v,t_{\text{order}}}$:

$$\tilde{I}_{v,t_{\text{order}}} \sim \text{Uniform}(\{1, 2, \ldots, \hat{L}_{v,t_{\text{order}}} - 1\}) \tag{3.23}$$

Pending Inventory Vector Update: The pending inventory vector update depends on whether the delay information has been revealed:

- Before delay information ($t \leq t_{\text{order}} + \tilde{I}_{v,t_{\text{order}}}$): Pending inventory vector uses expected lead time $\hat{L}_{v,t_{\text{order}}}$

- After delay information ($t > t_{\text{order}} + \tilde{I}_{v,t_{\text{order}}}$): Pending inventory vector uses actual sampled lead time $\tilde{L}_{v,t_{\text{order}}}$

Chapter 4

Methodology

We study a multi-supplier inventory control problem with stochastic demand and supplier-specific lead times. The objective is to compare the performance of a reinforcement learning (RL) agent with several classical heuristic and optimization-based baselines. The baselines considered are:

1. (s, R)**-policy:** A fixed-order-quantity reorder policy using R = Economic Order Quantity
2. **Mixed-Integer Linear Programming (MILP):** An optimization-based approach that plans orders over a finite horizon given demand forecasts.
3. **MILP-Imitation Policy:** Neural network policy trained to mimic the actions of the MILP solver via behavior cloning.

RL Agent. The RL agent learns an ordering policy through interaction with the environment, using Proximal Policy Optimization (PPO) for policy updates. We evaluate four initialization schemes:

1. Random initialization of the Value and Policy network (referenced throughout this thesis as **Vanilla PPO**)
2. Pretrained Policy Network: MILP-Imitation as starting point (referenced throughout this thesis as **PI-PPO**)
3. Pretrained Value network (referenced throughout this thesis as **VI-PPO**)
4. Combining (2) and (3) (referenced throughout this thesis as **BI-PPO**)

4.1 Baseline Methods

(s, R) **Policy** The (s, R) policy is a classical inventory control strategy where:

- R is the *reorder point*: the inventory level that triggers a replenishment order
- s is the *fixed order quantity* placed whenever inventory drops to or below R

We use the Economic Order Quantity (EOQ) to determine the order size s. The EOQ minimizes total inventory costs under assumptions of constant demand and is given by:

$$Q^* = \sqrt{\frac{2dk}{h}}, \tag{4.1}$$

In the multi-supplier setting, the policy selects the supplier that minimizes total cost while accounting for supplier-specific lead times and capacity constraints.

Mixed-Integer Linear Programming (MILP) The MILP baseline frames the problem 3.1 for multi-supplier inventory control problem as a deterministic optimization problem over a finite planning horizon, minimizing total expected cost.
If the MILP cannot find a feasible solution because current inventory levels are inadequate to satisfy safety stock constraints, these constraints are eliminated by setting the safety stock to zero.

MILP-Imitation Policy The MILP-Imitation Policy is a neural network trained to mimic the actions of the MILP solver via behavior cloning. For a given environment configuration, we generate various rollouts using the MILP policy to collect state-action pairs (s_t, a_t). The neural network is then trained in a supervised manner on this dataset. For continuous action space definition, we use mean squared error (MSE) loss to predict order quantities. For discrete action spaces, we use cross-entropy loss treating the problem as multi-class classification.

4.2 Reinforcement Learning Agent

We implement the RL agent using Proximal Policy Optimization (PPO) [10] from the Stable-Baselines3 library [11]. We choose PPO as it has been successfully applied in previous inventory management studies[12] and while being known for its training stability. PPO is a policy gradient method that maintains training stability through clipped objective functions while being sample efficient. The algorithm alternates between collecting experience through environment interaction and updating the policy and value networks using the collected trajectories.

The PPO implementation uses separate networks for policy and value function estimation, both sharing a common feature extraction backbone. Since the

shared feature extractor has no learnable parameters, this allows for separate pretraining of the policy and value networks when ensuring that both use the same normalization afterwards. This approach enables the four initialization schemes outlined in the overview.

Pretraining Methods

To accelerate convergence and improve sample efficiency, we employ two pretraining strategies that leverage domain knowledge from classical inventory management theory and optimization-based solutions to initialize the RL agent's networks.

Value Function Pretraining We generate supervised training data by systematically enumerating state combinations and computing target values using an (s, R)-based function approximation (similar to 4.1). The sampling process covers discrete inventory levels, various pending order configurations, constant demand forecasts, and all valid cost parameter combinations specified in the test environment configurations.

For each sampled state s_t, we compute the target value $\hat{V}(s_t)$ by simulating an $(s, R = EOQ)$ policy over a finite planning horizon. This policy follows EOQ principles, assuming constant demand and applying reorder point calculations based on safety stock requirements and lead times. The value function uses the same reward structure as the MDP formulation, consisting of holding, ordering, and stockout costs computed as discounted profit with the same discount factor γ used during RL training.

State Space Filtering: To focus training on states visited under an optimal policy, we filter the dataset to exclude configurations unreachable under near-optimal behavior. Specifically, we remove pending order patterns that our MILP-following baselines would never encounter - for instance, if optimal policies/underlying environment would never schedule orders more than n periods ahead in a given scenario, we exclude pending order configurations with non-zero quantities beyond n. This filtering reduces dataset size while maintaining coverage of behaviorally relevant state regions.

State Representation: States are encoded as normalized vectors containing the state space. Order quantities are normalized by their corresponding EOQ values to improve numerical stability and generalization across different cost regimes.

Training Objective: The value network is trained using supervised learning with Mean Squared Error loss between predicted and target values. Input normalization follows the same preprocessing pipeline used during RL training to ensure consistency.

Policy Function Pretraining Policy pretraining employs the behavior cloning methodology described for the MILP-Imitation baseline. State-action pairs (s_t, a_t) are collected from MILP solver rollouts across diverse environment configurations, and the policy network is trained via supervised learning to replicate these expert decisions.

Integration with Reinforcement Learning

Both pretraining approaches are implemented as independent supervised learning phases that provide network initialization for the RL agent. The pretrained value function offers improved state value estimates, while the pretrained policy provides a (near-)optimal initialization that avoids inefficient random exploration during early RL training.

This pretraining framework enables systematic evaluation of different initialization strategies: random initialization, value-only pretraining, policy-only pretraining, and combined pretraining of both networks.

Chapter 5

Evaluation

All plots showing training dynamics presented in this section display the median performance and their corresponding interquartile ranges, computed across 5 random seeds. Other plots may follow different notation as specified in their respective captions.

5.1 Ablation Studies

Reinforcement learning approaches involve numerous design decisions that can significantly impact agent performance. In the following sections, we examine how to model the action space design and develop effective representations for the observation space. Finally, we investigate the impact of parameter smoothness on policy learning—specifically comparing training on smooth parameter distributions versus sparse, to identify what is essential for successfully training policies across a set of products with varying characteristics.

5.1.1 Action Space Design

Motivation and Problem Statement The choice of action space representation fundamentally impacts both the learning dynamics and convergence properties of reinforcement learning algorithms. In inventory management, the challenge lies in balancing precision of control with the stability and efficiency of policy learning. We systematically compare continuous and discrete action space formulations to quantify their relative performance under varying degrees of environmental stochasticity.

Experimental Design We conduct two complementary experiments to isolate the effects of action space discretization and normalization on learning performance. All experiments use the baseline configuration from Appendix A with different action space definitions as specified below.

Experiment 1: Discretization vs. Normalization We compare discrete normalized actions against continuous unnormalized actions to disentangle the effects of discretization from EOQ-based scaling:

- **Discrete (Scaled):** Action space discretized into 31 intervals with $\alpha \in \{0, 0.1, 0.2, \ldots, 3.0\}$, where $X_t = \alpha \cdot \text{EOQ}$
- **Continuous (Unscaled):** Direct prediction of order quantities $X_t \in [0, X_{\max}]$ without EOQ normalization

Experiment 2: Pure Discretization Effect We isolate the impact of discretization by comparing both formulations under identical EOQ-based normalization:

- **Discrete (Scaled):** Categorical classification over $\alpha \in \{0, 0.1, 0.2, \ldots, 3.0\}$ with $X_t = \alpha \cdot \text{EOQ}$
- **Continuous (Scaled):** Continuous multiplier $\alpha \in [0, 3]$ with $X_t = \alpha \cdot \text{EOQ}$

Environmental Conditions Using our default configuration (see Appendix), we evaluate both experiments under controlled stochasticity levels with modified lead time distributions:

- **No Delay:** Fixed lead time of 4 periods
- **Delay:** Stochastic lead times following quantile distribution $\mathbb{P}(L = 4) = 0.9$, $\mathbb{P}(L = 5) = \mathbb{P}(L = 6) = 0.05$

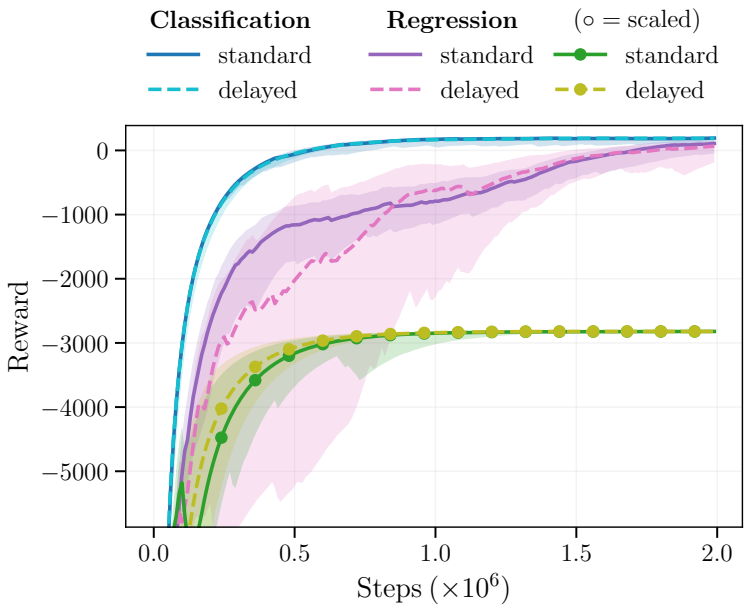


Figure 5.1: Training dynamics for standard vs. delayed training on classification and regression tasks.

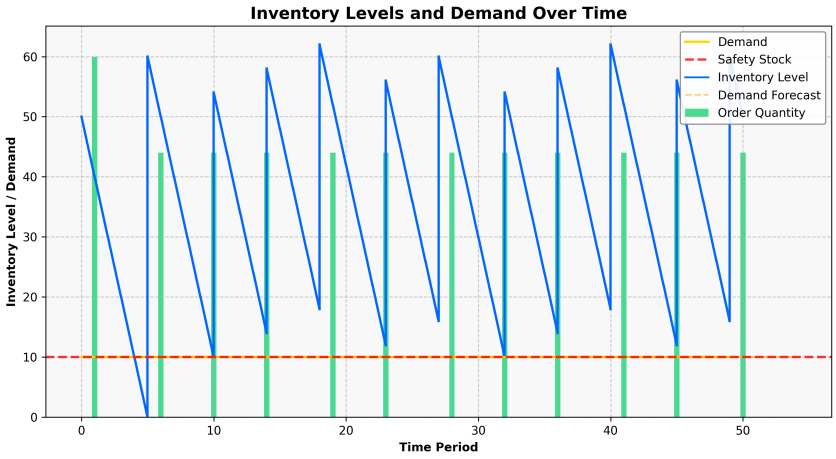
Result Figure 5.1 illustrates the learning dynamics for both action space formulations across environmental conditions. The discrete formulation demonstrates faster convergence and reduced variance throughout training, particularly evident in the stochastic environment.
The performance patterns observed in both experiments reveal a counterintuitive result: continuous unscaled actions achieve substantially better performance than continuous scaled actions. This suggests a potential interaction between PPO's internal action scaling mechanism and the constrained EOQ-normalized action space, though this hypothesis was not further investigated.
The discrete scaled formulation consistently outperforms the continuous approaches across both environmental conditions, while the continuous unscaled approach exhibits poor performance. Given these results, all subsequent analyses employ discrete action spaces.

Table 5.1: Experiment 1: Discretization vs. Normalization - Performance Across All Seeds (Mean ± Std)

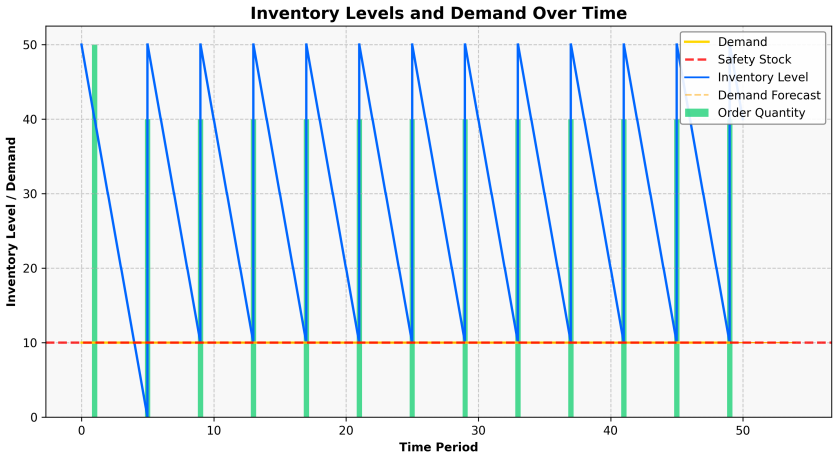
Setting	Discrete (Scaled)	Continuous (Unscaled)
No Delay	**221.48 ± 23.91**	-431.38 ± 231.44
Delay	**202.04 ± 37.69**	-589.12 ± 358.86

Table 5.2: Experiment 2: Pure Discretization Effect - Performance Across All Seeds (Mean ± Std)

Setting	Discrete (Scaled)	Continuous (Scaled)
No Delay	**221.48 ± 23.91**	-936.88 ± 647.63
Delay	**202.04 ± 37.69**	-754.12 ± 406.01



(a) Rollout from PPO using Discrete Action Space Formulation



(b) Rollout using MILP for planning

Figure 5.2: Exemplary inventory management rollouts over 50 timesteps comparing PPO-trained discrete policy against MILP optimal baseline. The PPO policy usually does not utilizes full safety stock levels (red dashed line), while the MILP policy consistently utilizes the full safety stock, allowing inventory to drop to the minimum threshold.

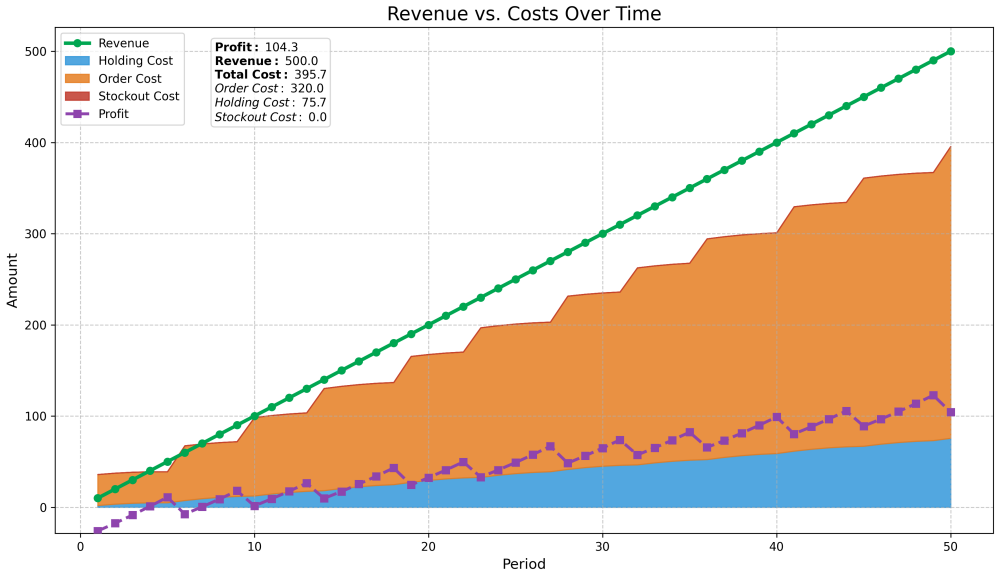


Figure 5.3: Cost and revenue breakdown for discrete PPO policy rollout, showing the trade-off between holding costs and order quantities over the simulation period.

5.1.2 Observation Space Design

Motivation and Problem Statement Beyond current inventory levels, inventory management systems must track several critical state variables that collectively determine optimal ordering decisions. Specifically, three key components provide forward-looking information essential for policy learning:

1. **Pending Inventory:** Outstanding orders that will arrive in future periods
2. **Demand Forecast:** Predicted future demand requirements
3. **Forecasted Inventory:** Projected inventory levels incorporating both pending orders and forecasted demand

However, these components exhibit inherent collinearity, as forecasted inventory represents an aggregated transformation of pending inventory and demand forecasts. This redundancy creates potential learning challenges, as agents may struggle to effectively utilize overlapping information signals during policy optimization.

The theoretical considerations suggest three potential approaches: (1) providing complete information despite redundancy, (2) selective removal of individual components to reduce collinearity, or (3) utilizing only the forecasted inventory as

the most informationally dense representation. We systematically evaluate these alternatives to quantify their impact on learning performance and convergence reliability.

Experimental Design We conduct a comprehensive ablation study examining four distinct observation space configurations to isolate the effects of information aggregation and component separation on learning dynamics.

Configuration Comparison We evaluate the following observation space formulations:

- **All:** Complete state information including current inventory, pending orders, demand forecasts, and forecasted inventory
- **Forecasted Inventory Only:** Aggregated forecasted inventory values without component separation
- **No Inventory Forecast:** Individual components (current inventory, pending orders, demand forecasts) without aggregated forecasted inventory
- **No Pending Quantity:** Current inventory, demand forecasts, and forecasted inventory without pending order information
- **No Demand Forecast:** Current inventory, demand forecasts, and forecasted inventory without pending order information

Environmental Conditions To evaluate the robustness of different observation space configurations, we identify key challenges that commonly occur in practical inventory management scenarios: noisy demand patterns, forecast bias, and their combination. These conditions allow us to assess how different state representations handle information uncertainty and misalignment.

Environmental Scenarios:

- **Deterministic**: Demand $d_t \equiv 10$ (constant), Forecast $F_t \equiv 10$ (aligned, no noise)
- **Noisy Demand**: Demand $d_t \sim \mathcal{N}(10, 3^2)$, Forecast $F_t \equiv 10$ (aligned forecast, stochastic demand)
- **Forecast Bias**: Demand $d_t \equiv 10$ (constant), Forecast $F_t \equiv 12$ (systematic forecast overestimation)
- **Combined Uncertainty**: Demand $d_t \sim \mathcal{N}(10, 3^2)$, Forecast $F_t \equiv 12$ (biased forecast with stochastic demand)

Each configuration is trained across 5 random seeds to assess both mean performance and training reliability.

Results

Performance Analysis The experimental results demonstrate that observation space configuration significantly impacts both performance and training stability. In the deterministic environment, **All** (blue solid line) achieves the highest median performance, indicating that complete state information is beneficial when environmental conditions are simple and predictable.

However, as environmental complexity increases with forecast bias, noisy demand, and combined uncertainty, **No Inventory Forecast** (red dashed line) consistently outperforms all other configurations. While **All** may achieve peak performance in ideal conditions, it shows degraded performance and increased variance in challenging environments.

No Inventory Forecast emerges as the optimal configuration due to its consistent performance across all environmental scenarios, maintaining stable training dynamics and reliable convergence regardless of demand uncertainty or forecast bias. This finding suggests that providing separated state components without aggregated forecasted inventory offers the most robust approach for diverse inventory management conditions.

Given these reliability and performance advantages across varying environmental complexities, all subsequent experiments employ the **No Inventory Forecast** configuration.

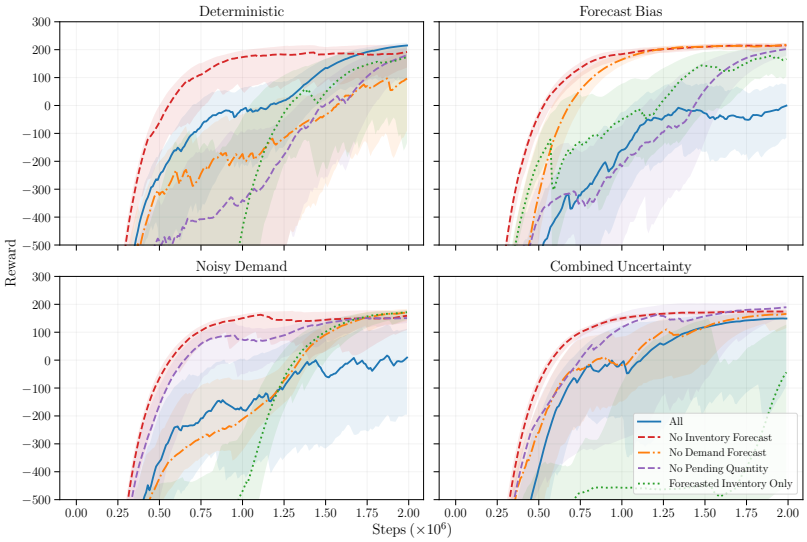


Figure 5.4: Training dynamics across observation space configurations under varying environmental conditions. "No Inventory Forecast" configuration (red dashed line) demonstrates consistent convergence and robust performance across all environmental scenarios, while other configurations show variable stability and performance depending on the complexity of demand patterns and forecast accuracy.

5.1.3 Training Space Design

When training reinforcement learning agents for deployment across multiple parameter configurations, a fundamental question arises regarding optimal training distribution design: should the training distribution sample exclusively from target deployment parameters (sparse sampling) or incorporate intermediate values across the parameter space (dense sampling)?

We investigate this question through a systematic evaluation of parameter space sampling strategies, examining their impact on learning dynamics and generalization performance across distinct training paradigms.

Experimental Design We evaluate two distinct approaches to isolate parameter space modeling effects. **Curriculum Learning**, which involves sequential fine-tuning from a single-parameter baseline (previously trained on one product), and **Training from Scratch**, where we employ direct multi-parameter optimization.

Evaluation Protocol: To assess generalization across parameter configurations with distinct optimization landscapes, we evaluate performance on two

products sharing identical underlying dynamics as our baseline product but exhibiting significantly different cost structures:

- **Product 1**: Low holding cost configuration (h = 0.05)
- **Product 2**: High holding cost configuration (h = 0.15)

The threefold increase in holding costs fundamentally alters optimal ordering policies, inducing substantially reduced order quantities for Product 2, thereby creating a rigorous test case for parameter space modeling effectiveness.

Parameter Sampling Strategies:

- **Dense Sampling**: Uniform distribution over 11 discrete holding cost values [0.05, 0.06, ..., 0.15]
- **Sparse Sampling**: Binary distribution over boundary values {0.05, 0.15}

The holding cost parameter represents per-unit inventory carrying cost, a critical determinant of optimal ordering policies in inventory management systems.

Performance under Curriculum Learning

Evaluation	Dense CL	Sparse CL
Product 1 (h=0.05)	182.28 ± 39.99	171.24 ± 61.87
Product 2 (h=0.15)	22.94 ± 21.86	-56.76 ± 54.85
Mean Performance	102.62	57.24

Dense Curriculum Learning (DCL) sampling demonstrates superior performance with a 45.4-point advantage over sparse sampling. Critically, dense sampling maintains profitable operations on Product 2 (22.94), while sparse sampling results in negative returns (-56.76), indicating policy degradation under parameter shift.

Parameter Space Modeling Under Training From Scratch

Evaluation Scenario	DMT (Dense)	BMT (Sparse)
Product 1 (h=0.05)	116.36 ± 131.67	-108.88 ± 458.64
Product 2 (h=0.15)	25.28 ± 19.02	-73.76 ± 59.85
Mean Performance	70.82	-91.32

Dense parameter sampling exhibits a substantial 162.14-point performance advantage over sparse sampling (70.82 vs -91.32). Dense sampling maintains

positive profitability across both configurations, whereas sparse sampling demonstrates systematic training instability with negative returns across all evaluation scenarios.

Analysis and Interpretation

Dense parameter space modeling consistently outperforms sparse sampling across both training paradigms, with effect sizes varying by training approach: moderate for curriculum learning (+22.7 points) but critical for direct training (+81.1 points), where sparse sampling leads to complete training failure.

Hypothesis: During training with sparse parameter configurations, stochastic parameter sampling generates divergent gradient signals from distinct cost structures that may exhibit destructive interference, resulting in unstable optimization dynamics. Dense sampling across a continuous parameter space may provide more coherent gradient information, facilitating convergence to superior optima.

Implications: For reinforcement learning systems requiring deployment across heterogeneous parameter configurations, training distributions should incorporate intermediate parameter values rather than focusing exclusively on target deployment points, enabling more robust policy generalization.

5.2 Single Supplier Setup

5.2.1 Training Under Noisy Demand

Motivation and Problem Statement

A fundamental tenet in reinforcement learning for stochastic optimization problems is that training agents under conditions that mirror deployment scenarios enhances robustness and generalization performance. In inventory management contexts, this principle suggests that exposing agents to demand variability during training should develop adaptive policies for managing uncertainty during deployment.

However, this intuitive assumption lacks empirical validation within inventory management domains. The presence of demand noise during training introduces additional complexity to the learning process, potentially interfering with the agent's ability to learn good policies. This raises the following research question: Does training under noisy demand conditions actually enhance agent performance when evaluated under stochastic demand environments, or does deterministic training provide sufficient robustness while avoiding learning complications?

Experimental Design

We systematically investigate training strategy efficacy across two primary experimental dimensions to isolate the effects of demand variability and safety stock considerations on learning outcomes.

Training Environment Configurations:

- **Deterministic Training**: Agents trained under constant demand conditions ($d_t \equiv 10$) with perfectly aligned forecasts ($F_t \equiv 10$)
- **Noisy Training**: Agents trained under stochastic demand ($d_t \sim \mathcal{N}(10, 3^2)$) while maintaining constant forecasts ($F_t \equiv 10$)

Safety Stock Policy Variants:

- **Standard Configuration**: Safety stock penalty applied during training when inventory falls below 10 items, encouraging buffer inventory maintenance
- **No Safety Stock Configuration**: No safety stock penalty during training, allowing agents to operate with leaner inventory policies

Evaluation Protocol: Each training configuration undergoes evaluation under both deterministic and stochastic demand scenarios to assess cross-condition

generalization capacity. During evaluation, all safety stock penalties are removed and only standard profit metrics are measured. For deterministic training, 5 different random seeds are evaluated against the same deterministic evaluation environment. For noisy training, 5 different training runs are evaluated against 5 different evaluation seeds to account for stochastic variance. All experiments employ 5 mio. training steps to ensure convergence.

Results

Training Dynamics and Convergence Analysis The learning trajectories reveal distinct convergence patterns across training configurations, as illustrated in Figure 5.5.

Deterministic demand training (solid blue line) demonstrates steady convergence, reaching approximately 220 reward units and maintaining stable performance throughout the later training phases. When safety stock is removed from deterministic training (dashed blue line), performance improves significantly, achieving nearly 300 reward units with slightly higher variance.

Noisy demand scenarios exhibit more challenging learning dynamics. Standard noisy demand training (solid green line) shows lower overall performance, converging to approximately 180 reward units with moderate variance. The noisy demand configuration without safety stock (dashed green line) displays the most volatile behavior, with higher peak performance around 240 reward units but significantly greater variance and instability throughout training.

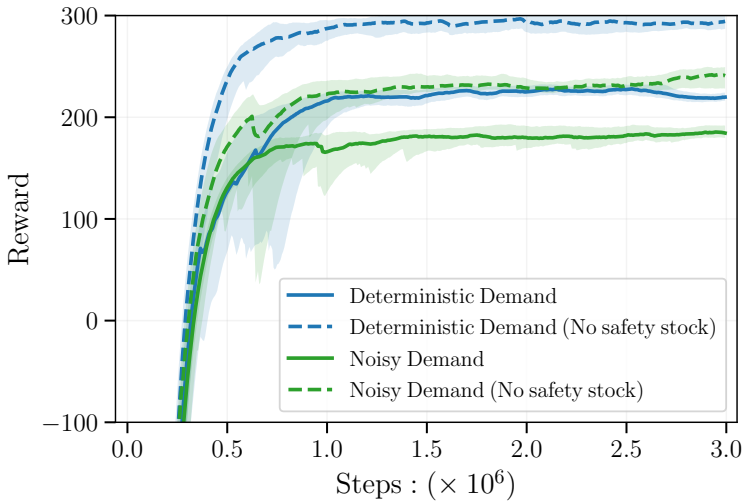


Figure 5.5: Performance comparison across training configurations under deterministic and stochastic evaluation conditions for scenarios with and without safety stock

Cross-Condition Performance Analysis The evaluation results, summarized in Table 5.3, reveal counterintuitive findings that challenge conventional assumptions about noise-matched training benefits.

Training Method	Deterministic Eval.	Stochastic Eval.
Deterministic	219.48 ± 16.48	202.90 ± 18.40
Deterministic (No Safety Stock)	286.68 ± 14.03	211.88 ± 22.45
Noisy	181.16 ± 30.32	183.86 ± 25.52
Noisy (No Safety Stock)	251.16 ± 17.26	207.28 ± 24.76

Table 5.3: Performance Under Different Evaluation Conditions

Empirical Findings When evaluated under stochastic demand conditions, deterministic training achieves superior performance (202.90) compared to noisy training (183.86), representing a 10.4% performance advantage. This contradicts the expectation that noise-matched training would yield better results under uncertain conditions.

Cross-Condition Robustness: Deterministic training methods maintain more consistent performance across evaluation conditions, with only a 7.5% performance drop when moving from deterministic to stochastic evaluation. In contrast, noisy training shows minimal performance variation (1.4% improvement), but from a substantially lower baseline.

Training Stability: Noisy training with safety stock exhibits higher performance variance (± 25.52 to ± 30.32) compared to deterministic training (± 16.48 to ± 18.40), indicating less reliable policy convergence.

Conclusion

This investigation provides compelling evidence that deterministic training environments are superior for developing inventory management policies capable of operating under demand uncertainty. Deterministic training consistently outperforms noisy training when evaluated under stochastic conditions, while also demonstrating improved training stability and convergence characteristics.

However, these findings should be interpreted within the context of the experimental parameters. The demand variability in our study ($d_t \sim \mathcal{N}(10, 3^2)$) represents relatively modest deviations around the mean, with deviations being smaller than the safety stock buffer range. For scenarios involving substantially higher demand volatility, where stochastic realizations frequently exceed safety stock levels, training on noisy demand may become advantageous. Under such conditions, the optimal policy structure would require fundamentally different

threshold and ordering behaviors, including adaptation to higher safety stock levels, that deterministic training may fail to capture.

The consistent performance advantages of deterministic training across evaluation scenarios, combined with improved training stability, establish a strong empirical foundation for adopting simplified training protocols in inventory management applications with moderate demand uncertainty.

5.2.2 Multi-Product Generalization

Motivation and Problem Statement

From previous studies, we have demonstrated that reinforcement learning agents can effectively learn optimal policies for individual product configurations. However, a critical question emerges: can a single PPO instance generalize across multiple product categories while maintaining the same underlying product dynamics? This investigation examines whether agents can develop robust procurement strategies that adapt to varying economic parameters—specifically different holding costs and supplier cost structures—while keeping demand patterns constant.

The core research question becomes: To what extent can reinforcement learning agents develop generalized inventory management capabilities across diverse product configurations with varying economic parameters, and how do different pretraining strategies affect this generalization capacity?

Experimental Design

We construct a comprehensive evaluation framework using the environmental setup detailed in Appendix A, with the following parameter modifications from the baseline configuration:

Parameter Variation:

- **Holding costs**: Discrete distribution over {0.1, 0.2}
- **Supplier characteristics**:
  - Fixed costs: Discrete distribution over {6, 8, 10}
  - Unit costs: Discrete distribution over {0.4, 0.5, 0.6}
  - Maximum capacity: 350 units
  - Lead time: 4 periods

This experimental framework systematically evaluates agent performance across 18 distinct product configurations, representing the Cartesian product of parameter variations (2 holding cost levels $\times$ 3 fixed cost levels $\times$ 3 unit cost levels), to isolate multiproduct effects.

Training Configuration: Based on insights from Section 5.1.3 studies, we employ both training methods - with continuous parameter distributions during training: holding cost uniformly distributed over [0.1, 0.2], fixed cost over [6, 10], and unit cost over [0.4, 0.6] as well as on discrete parameter distributions. Both approaches were tested and the results are shown in Figure 5.7.

Results

Pretraining Duration and Performance Saturation The relationship between pretraining duration and initial policy quality, shown in Figure 5.6, reveals critical insights about optimal pretraining strategies. Performance improves dramatically from 50 epochs (123.32) to 75 epochs (197.94), with continued but smaller gains through 100 epochs (220.52) and 125 epochs (229.82). The policy reaches near-optimal performance at 150 epochs (229.48) and maintains this level at 175 epochs (233.34), closely matching and slightly exceeding the MILP baseline (227.88).

This saturation pattern indicates that while some pretraining is essential for effective policy initialization, extending pretraining beyond 150 epochs yields minimal additional benefit. The policy's ability to slightly exceed the MILP baseline stems from its more flexible approach to safety stock constraints, allowing

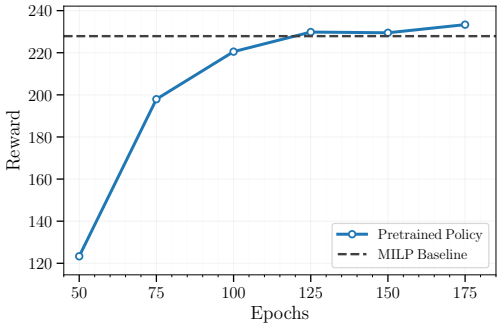


Figure 5.6: Initial policy performance versus pretraining duration for behavioral cloning training epochs.

for slightly smaller holding costs compared to the strictly enforced constraints in the MILP formulation.

Training Dynamics Analysis The training dynamics over 10 million timesteps reveal fundamental differences between continuous and discrete parameter environments across all pretraining methodologies, as demonstrated in the training curves.

Vanilla PPO: Vanilla PPO demonstrates contrasting performance between environments with high seed dependency. The median trajectories show that PPO trained on the discrete environment - although being more unstable in the beginning - outperforms PPO trained on the continuous environment. The continuous environment shows slightly smoother median convergence, while the discrete environment exhibits more fluctuations around 4-6 million steps. Both environments display wide confidence intervals throughout training, particularly in the middle phases (2-6 million steps), indicating that individual seed performance varies dramatically from the median. Notably, detailed seed-level analysis reveals that in continuous settings, 2 out of 5 seeds perform well reaching around 150 reward units while 3 seeds struggle at around -100 units, whereas the discrete environment shows improved success rates with 3 out of 5 seeds performing well around 150 units, though 2 seeds still achieve only modest performance at -100 units. This bimodal distribution pattern explains the large confidence intervals observed in the median plots and suggests a potential advantage for discrete environments, though this could be a random effect given the limited sample size.

VI-PPO: VI-PPO exhibits the most dramatic environment-dependent performance difference. In continuous settings, it achieves stable performance around 150-200 reward units with minimal variance. However, discrete parameter sampling severely impairs performance, with median rewards around -500 units, representing a 700-unit performance degradation that highlights the substantial optimization challenges introduced by discrete sampling.

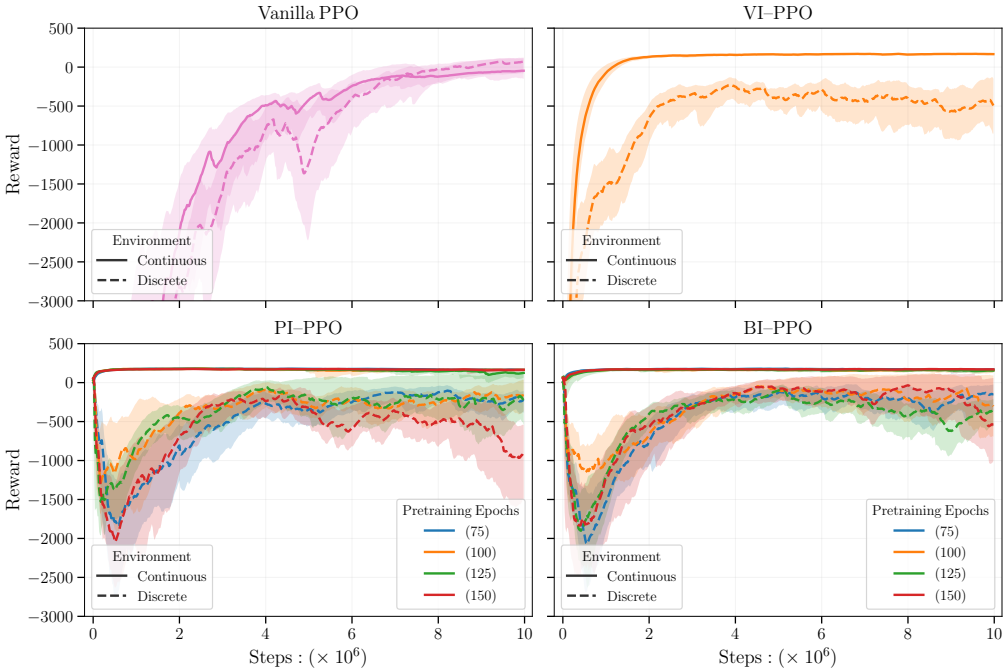


Figure 5.7: Comparison of training dynamics between continuous and discrete parameter environments across different pretraining methodologies over 10 million training steps. Solid lines represent continuous environments, dashed lines represent discrete environments.

PI-PPO: PI-PPO demonstrates rapid convergence in continuous environments, with all seeds reaching top performance quickly, though variance increases throughout training until the end. In discrete settings, the median performance reaches top levels between 4-5 million steps. Interestingly, variants where the underlying policy networks were trained for 125 epochs show particularly strong performance—however, those trained for 150 epochs perform the worst in the end, suggesting an optimal pretraining duration for discrete environment adaptation.

BI-PPO: BI-PPO shows roughly similar stability to VI-PPO across both environments. While it achieves slightly worse top performance compared to PI-PPO, BI-PPO maintains more consistent performance ranges throughout training in both continuous and discrete settings.

Environment Performance Characteristics: The key outcome is that the continuous environment consistently enables superior training performance when conducting pretraining across all methodologies, with the notable exception of vanilla PPO which shows better success rates in discrete settings. Training on discrete environments generally produces much worse noisier performance compared to continuous settings for all pretrained variants, which show degraded performance and increased variance compared to their continuous counterparts.

Comparative Performance Analysis

Comparative Performance Analysis The evaluation across 18 distinct product configurations reveals complex performance patterns that challenge conventional assumptions about pretraining effectiveness and parameter space design, as summarized in Tables 5.4 and 5.5.

Dramatic Environment-Dependent Performance Gaps: The results reveal a systematic and substantial performance advantage of discrete environments over continuous ones for most pretrained methods. The performance gaps are striking - for example, BI-PPO (100) achieves 149.22 ± 67.36 in discrete settings but -167.52 ± 351.77 in continuous environments, representing a 317-unit performance difference.

Pretrained Methods Outperform Baselines in Mean Performance: All pretrained variants substantially outperform vanilla PPO and VI-PPO in mean performance. The best-performing methods (PI-PPO and BI-PPO with longer pretraining) achieve mean rewards of 140-160 units in discrete environments, compared to vanilla PPO's -71.10 ± 278.60. However, when examining the best 3 seeds, vanilla PPO achieves surprisingly competitive performance at 168.52 ± 49.97 in discrete environments, demonstrating that while vanilla PPO suffers from huge variance during evaluation, its top-performing seeds can discover effective policies comparable to pretrained methods and better than VI-PPO.

Method Performance Depends on Evaluation Metric: BI-PPO (150) achieves the highest mean performance at 154.04 ± 70.44 in discrete environments, slightly outperforming PI-PPO variants when considering all seeds. However, when examining best seed performance, PI-PPO (125) emerges as the top performer, achieving 209.12 ± 32.54 in continuous environments—nearly optimal performance approaching the MILP benchmark of 225.78. This demonstrates clear seed dependence, where PI-PPO's best seeds significantly outperform its mean performance, indicating high variance with some exceptional performers.

Extreme Variance in Continuous Environments: Continuous environments exhibit dramatically higher variance, with some methods showing standard deviations exceeding 350 units (BI-PPO 100: ±351.77). This massive variance indicates highly unstable final performance despite the smoother training dynamics observed earlier.

Seed Dependence and Performance Variability: The comparison between mean and best seed performance reveals substantial seed dependence across methods. While BI-PPO shows more consistent performance across seeds, PI-PPO demonstrates higher potential with exceptional top performers. For instance, PI-PPO (125) achieves 209.12 in its best seeds versus 161.14 mean performance in continuous environments, highlighting the critical role of seed selection and the potential for discovering near-optimal policies within high-variance training.

Training vs Evaluation Environment Mismatch Training Environment Superiority vs Final Performance: Although training on continuous environments demonstrates superior learning dynamics with reduced variance and more stable convergence, the final evaluation results reveal a counterintuitive finding. Despite the smoother training trajectories observed in continuous settings, discrete environments consistently yield better final performance across most methods. This suggests that continuous parameter spaces facilitate easier learning and optimization during training, but the learned policies may not transfer effectively to the evaluation environment.

Snapshot Performance Hypothesis: The substantial difference between training stability and final performance suggests that some intermediate policy snapshots during the apparently "unstable" discrete training may actually capture better solutions than the final converged policies from stable continuous training. This indicates that the high variance observed during discrete training may reflect exploration of better policy regions rather than simple instability, with some training seeds successfully discovering superior strategies that are not captured by the mean performance metrics.

This analysis reveals a complex relationship between training dynamics and final performance that depends on the evaluation metric. While continuous environments provide superior training stability, discrete environments yield more

robust median performance across all seeds during evaluation. However, the top-performing seeds from continuous training actually outperform the best discrete-trained policies, approaching near-optimal performance. This suggests that continuous training enables discovery of exceptional policies but with high variance, while discrete training provides more consistent but lower-ceiling performance. The findings suggest that:

1. Continuous environments provide better optimization landscapes for discovering exceptional policies but with high variance
2. Discrete environments produce more consistent policy quality across seeds, improving median performance reliability
3. The choice between training environments involves a trade-off between achieving better peak performance (continuous) versus consistent performance (discrete)

Method	**Discrete**	**Continuous**
MILP (No Safety Stock)	**367.56**	**367.56**
MILP	**225.78**	**225.78**
PI-PPO (150)	147.30 ± 78.83	61.36 ± 126.28
PI-PPO (125)	143.06 ± 62.82	161.14 ± 55.91
PI-PPO (100)	131.50 ± 84.88	**120.28 ± 96.57**
PI-PPO (75)	124.64 ± 107.37	36.66 ± 142.72
BI-PPO (150)	**154.04 ± 70.44**	-9.88 ± 137.14
BI-PPO (125)	116.58 ± 112.25	-25.50 ± 227.49
BI-PPO (100)	149.22 ± 67.36	-167.52 ± 351.77
BI-PPO (75)	72.92 ± 84.33	68.34 ± 112.50
VI-PPO	62.14 ± 112.49	-194.58 ± 173.19
Vanilla PPO	-71.10 ± 278.60	-444.84 ± 328.78

Table 5.4: Mean profit performance (± standard deviation) across all seeds when evaluated on discrete parameters, comparing models fine-tuned in discrete vs continuous environments

Method	**Discrete**	**Continuous**
MILP (No Safety Stock)	**367.56**	**367.56**
MILP	**225.78**	**225.78**
PI-PPO (150)	**219.10 ± 36.24**	177.16 ± 62.58
PI-PPO (125)	196.64 ± 26.39	209.12 ± 32.54
PI-PPO (100)	200.04 ± 36.14	**200.58 ± 37.20**
PI-PPO (75)	215.90 ± 32.91	149.28 ± 29.71
BI-PPO (150)	211.12 ± 43.34	114.78 ± 71.05
BI-PPO (125)	209.34 ± 39.68	142.96 ± 47.25
BI-PPO (100)	204.74 ± 44.26	83.06 ± 73.74
BI-PPO (75)	140.92 ± 53.30	170.18 ± 52.16
VI-PPO	158.54 ± 54.62	-46.46 ± 68.43
Vanilla PPO	168.52 ± 49.97	-155.24 ± 163.00

Table 5.5: Mean profit performance (± standard deviation) across the top 3 of 5 seeds when evaluated on discrete parameters, comparing models fine-tuned in discrete vs continuous environments

5.3 Multisupplier Setups

The transition from single-supplier to multi-supplier inventory management introduces fundamental challenges in action space complexity and credit assignment. When multiple suppliers offer competing services, reinforcement learning agents must learn to differentiate between options based on subtle economic signals—such as unit costs, fixed ordering costs—as well as operational characteristics including lead times, delivery reliability, and capacity constraints.

5.3.1 Supplier Selection with Homogeneous Cost Structures

Motivation and Problem Statement

When multiple suppliers offer identical operational capabilities, the fundamental challenge becomes whether reinforcement learning agents can learn to differentiate based purely on economic advantages. This scenario isolates the core learning problem from operational complexities. The homogeneous supplier environment represents the simplest multi-supplier learning challenge, yet introduces complexity through action space expansion.

This investigation addresses the research question: To what extent can reinforcement learning agents develop cost-based supplier differentiation capabilities, and how does this learning capacity generalize across diverse product configurations?

Experimental Design

We construct a controlled environment extending our baseline configuration (see Appendix A) to include two identical suppliers. Both suppliers maintain the same operational characteristics — fixed costs, lead times — while we vary the unit cost of the second supplier to create measurable economic advantages.

Eight product configurations test unit cost differentials: 0%, 5%, 10%, 15%, 20%, 30%, 40%, and 50%, with corresponding Supplier 2 unit costs of 0.5, 0.525, 0.55, 0.575, 0.6, 0.65, 0.7, and 0.75. This experimental design isolates the pure cost differentiation challenge while maintaining all other environmental factors constant, providing clear attribution of behavioral changes to economic learning.

Results

Training Dynamics Analysis Figure 5.8 illustrates the learning dynamics across the pretraining strategies for the homogeneous supplier configuration. The training dynamics show distinct convergence patterns that highlight the complexity of multi-supplier learning.

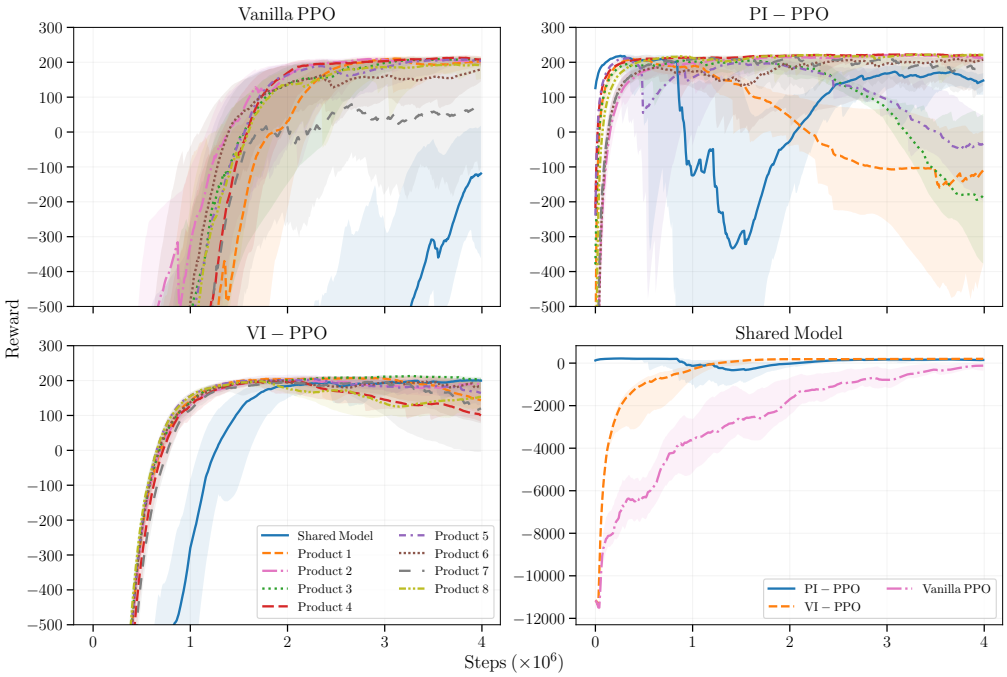


Figure 5.8: Training Performance across different strategies for homogeneous cost structures

Vanilla PPO Vanilla PPO requires approximately 2-3 million steps for gradual convergence, except for the shared model. Product 7 (gray dash-dot line) shows near zero reward throughout training. Product 1 (orange dashed line) exhibits delayed convergence, only beginning to improve above 0 reward after 2 million steps.

Pretrained Policy Network PI-PPO achieves immediate high performance through initialization but experiences performance degradation from 500k steps onwards for some products. This pattern is most pronounced for the shared model training (1M steps) as well as for Product 1 (1.5M steps). Product 3 and 5 experience the same degradation pattern, occurring later in training.

Pretrained Value Network VI-PPO exhibits the most stable learning trajectory with reduced variance compared to Vanilla PPO. The smooth convergence indicates that value initialization provides effective guidance for policy optimization. The method also shows performance degradation during fine-tuning as PI-PPO, though less severe. The shared model reaches competitive performance after 2 million steps without degradation.

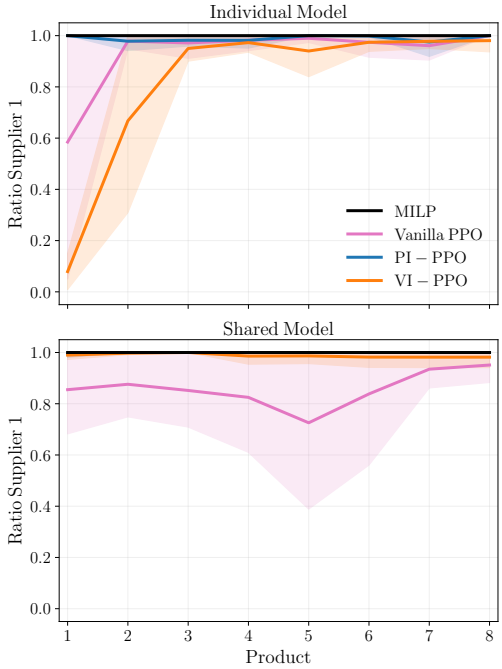


Figure 5.9: Supplier selection performance across eight products for individual models (top) and shared models (bottom).

Cross-Method Performance Variability Each method demonstrates varying sensitivity to cost differentials. With high cost differentials, all methods converge to optimal supplier selection. Intermediate cost differentials show the highest variance across methods, indicating that moderate economic signals present the greatest learning challenge for reinforcement learning agents.

Figure 5.9 presents the supplier selection behavior across the eight cost differential scenarios, demonstrating the agents' ability to develop cost-optimal procurement strategies.

Individual Model Performance Individual models demonstrate consistent cost-based differentiation across all approaches. PPO and PI-PPO achieve $\geq 96\%$ optimal supplier selection across scenarios where Supplier 2 has higher unit costs. VI-PPO shows strong performance (95-98%) except for Product 1, where identical supplier costs make any selection ratio economically equivalent.

Shared Model Scalability Multi-product training shows notable performance divergence: PPO (Shared) degrades to 73-95% optimal selection indicating multi-task learning interference, while PI-PPO (Shared) maintains $\geq$99% optimal selection and VI-PPO (Shared) achieves 98-100% optimal selection with minimal

variance.

RL agents learn cost-based supplier differentiation when trained individually. Shared model success depends on pretraining strategy: policy initialization (PI-PPO) maintains ≥99% optimal selection, while value initialization (VI-PPO) achieves 98-100% optimal selection, both significantly outperforming vanilla PPO's 73-95% selection rates.

Product	1	2	3	4	5	6	7	8
MILP	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Vanilla PPO (Ind. Model)	0.58	0.98	0.97	0.98	0.99	0.97	0.96	1.00
Vanilla PPO (Shared Model)	0.85	0.88	0.85	0.82	0.73	0.84	0.93	0.95
PI-PPO (Ind. Model)	1.00	0.98	0.98	0.98	1.00	1.00	0.98	1.00
PI-PPO (Shared Model)	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00
VI-PPO (Ind. Model)	0.08	0.67	0.95	0.97	0.94	0.97	0.98	0.98
VI-PPO (Shared Model)	0.99	1.00	1.00	0.99	0.99	0.98	0.98	0.98

Table 5.6: Supplier selection performance across eight product configurations. Values represent the ratio of procurement allocated to the optimal supplier (Supplier 1).

Performance Analysis

The quantitative results from Table 5.7 establish a clear performance hierarchy and identify scalability constraints. Individual product training shows PI-PPO achieving highest performance (mean reward = 119.8) with consistent supplier selection, value pretraining delivering moderate improvement (mean reward = 93.1-101.8) with low variance, and vanilla PPO providing baseline performance (mean reward = 99.7) but with high variance.

Shared model training demonstrates marked performance degradation where vanilla PPO drops from 99.7 to 23.8 mean reward, while pretraining methods maintain advantages with policy pretraining remaining most robust across configurations, though high variance indicates instability with shared models exhibiting inconsistent convergence patterns.

Training Stability Assessment

The analysis identifies instability patterns across all pretraining methods where policy degradation consistently occurs after episode 200, suggesting potential issues with learning rate scheduling, exploration-exploitation balance, or interference from multi-product training dynamics.

Method	1	2	3	4	5	6	7	8	Mean
MILP (No Safety Stock)	245.0	245.0	245.0	245.0	245.0	245.0	245.0	245.0	245.0
MILP (With Safety Stock)	184.5	184.5	184.5	184.5	184.5	184.5	184.5	184.5	184.5
Distilled MILP	109.3	106.9	126.5	126.5	126.5	126.5	119.4	128.6	121.3
Vanilla PPO (Ind. Model)	112.9	105.7	110.5	108.4	113.1	100.1	25.7	110.3	99.7
Vanilla PPO (Shared Model)	42.4	29.5	38.8	15.3	−47.4	45.8	39.5	26.1	23.8
VI-PPO (Ind. Model)	99.1	90.9	107.5	49.1	90.8	105.0	93.5	101.2	93.1
VI-PPO (Shared Model)	96.2	88.5	104.5	103.7	98.7	107.4	105.7	109.9	101.8
PI-PPO (Ind. Model)	118.3	121.7	118.9	119.5	121.2	117.5	120.1	121.5	119.8
PI-PPO (Shared Model)	126.5	113.8	114.1	116.6	122.9	109.9	102.2	110.2	114.0

Table 5.7: Mean profit performance across products averaged over multiple random seeds. Values represent average profit achieved for each product configuration

5.3.2 Supplier Selection with Variable Cost Differentials

Motivation and Problem Statement

While homogeneous supplier environments isolate pure cost differentiation learning, real-world procurement scenarios require agents to navigate continuously varying economic advantages across product portfolios. This investigation extends the multi-supplier learning challenge by introducing systematic cost variations that create an economic gradient, testing whether reinforcement learning agents can develop nuanced supplier selection strategies that adapt to varying degrees of cost advantage.

The core research question becomes: Can reinforcement learning agents learn to make economically rational supplier selections across a spectrum of cost differentials, and how does this gradient learning capability scale across shared model architectures?

Experimental Design

We construct a more complex environment that builds upon the baseline two-supplier configuration. Unlike the homogeneous case where suppliers differed only in unit costs, this scenario incorporates realistic trade-offs between fixed ordering costs and unit costs that mirror real-world supplier economics.

While otherwise adhering to the baseline configuration (see Appendix A), the experimental design features two suppliers with asymmetric cost structures: Supplier 1 maintains lower fixed costs (4) with moderate unit costs (0.5), while Supplier 2 operates with higher fixed costs (6) but variable unit costs. Using EOQ theory, we calculate break-even points and generate eight evaluation scenarios where Supplier 2's total costs vary from 25% cheaper to 25% more expensive than the break-even point, creating a continuous cost gradient across configurations.

Training Configuration: All experiments use identical training durations as the homogeneous cost experiments to enable direct comparison of learning progress at equivalent time horizons. This controlled experimental setup allows us to assess the relative difficulty of learning cost gradients versus simple cost differentiation within the same computational budget.

This design tests whether agents can learn to balance the trade-off between fixed ordering costs and variable unit costs - a fundamental procurement optimization challenge that requires understanding both immediate and cumulative cost implications.

Results

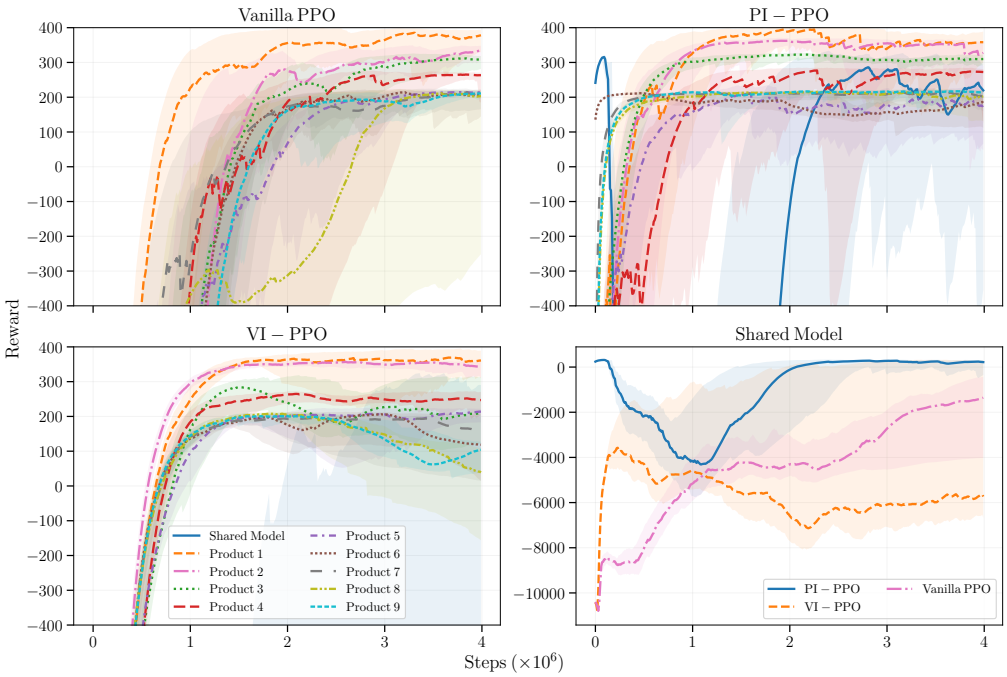


Figure 5.10: Training performance across different strategies for variable cost differential structures

Training Dynamics Figure 5.10 illustrates the learning dynamics across pretraining strategies for the cost gradient configuration. The training patterns reveal significantly more complex convergence behavior compared to the homogeneous supplier case, highlighting the increased difficulty of learning optimal policies when multiple cost factors interact.

Vanilla PPO (Scratch) shows gradual but unstable convergence across individual product configurations, requiring approximately 3-4 million steps for meaningful performance improvements. Products 1-3 reach performance of 300

units within this timeframe, however the model faces greater challenges when balancing cheaper unit costs against higher fixed costs and vice versa. The shared model training demonstrates severe learning difficulties, achieving positive median performance only after 4 million steps.

Pretrained Policy Network provides immediate performance advantages but suffers from high variance after initial convergence. The shared model shows strong early performance followed by performance degradation to approximately -4000 units, indicating that the policy initialization may create unstable learning trajectories in complex cost environments.

Pretrained Value Network exhibits the most stable learning trajectory with consistent convergence patterns across configurations. However, for products 7-9, where Supplier 1 is clearly the better option, performance degrades relatively quickly, suggesting limitations in value function generalization when the agents must learn to switch from their initial Supplier 2 preference to the economically superior Supplier 1.

Method	Mean ± Std
MILP (No Safety Stock)	183.8
MILP (With Safety Stock)	157.7
PPO (Indiv.)	118.8 ± 35.0
PPO (Shared)	-962.2 ± 628.8
VI-PPO (Indiv.)	126.8 ± 16.4
VI-PPO (Shared)	-476.8 ± 198.5
PI-PPO (Indiv.)	135.3 ± 11.3
PI-PPO (Shared)	-471.0 ± 296.2

Table 5.8: Performance comparison across methods for variable cost differential structures with mean ± standard deviation across products and random seeds

Model Performance The quantitative results from Table 5.8 establish clear performance hierarchies and reveal the complexity of gradient-based cost learning. **It is crucial to note that the poor performance of shared models stems from incomplete training rather than catastrophic failure**. The training snapshots at comparable time periods demonstrate that shared models require significantly longer training horizons to develop effective multi-product strategies, reflecting the inherent complexity of learning across cost gradients rather than fundamental algorithmic limitations.

Individual product training demonstrates significant success with PI-PPO achieving the highest performance at a mean reward of 135.3 with stable learning characterized by relatively low variance. VI-PPO delivers competitive performance with a mean reward of 126.8 and demonstrates consistent economic

rationality within the training timeframe. Vanilla PPO provides baseline performance with a mean reward of 118.8 but exhibits high variance across scenarios, indicating less reliable optimization during this training period.

The shared model approaches show dramatically negative rewards during this training snapshot, with PPO (Shared) at -962.2, VI-PPO (Shared) at -476.8, and PI-PPO (Shared) at -471.0. However, these results reflect the training state at the evaluation timepoint rather than converged performance. The magnitude of these negative values indicates that shared models are still in early learning phases, requiring extended training to develop the complex multi-task coordination necessary for cost gradient optimization.

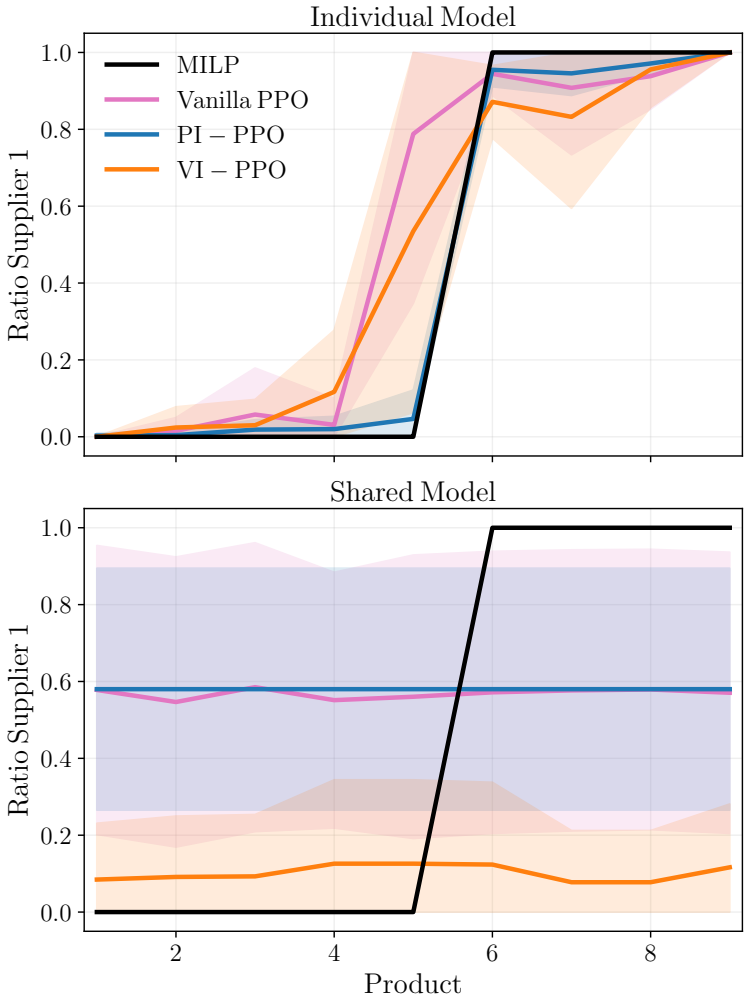


Figure 5.11: Supplier selection performance across Product 1 -9 for individual models (top) and shared models (bottom).

Product	1	2	3	4	5	6	7	8	9
MILP	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00
Vanilla PPO (Indiv. Model)	0.00	0.02	0.06	0.03	0.79	0.94	0.91	0.94	1.00
Vanilla PPO (Shared Model)	0.58	0.55	0.59	0.55	0.56	0.57	0.58	0.58	0.57
PI-PPO (Indiv. Model)	0.00	0.00	0.02	0.02	0.05	0.95	0.95	0.97	1.00
PI-PPO (Shared Model)	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58
VI-PPO (Indiv. Model)	0.00	0.02	0.03	0.12	0.53	0.87	0.83	0.96	1.00
VI-PPO (Shared Model)	0.08	0.09	0.09	0.13	0.13	0.12	0.08	0.08	0.12

Table 5.9: Supplier selection ratios across cost gradient scenarios

Supplier Selection Behavior Figure 5.11 and Table 5.9 present supplier selection behavior across the cost gradient scenarios, demonstrating the agents' ability to develop economically rational procurement strategies under varying cost structures.

Individual Model Performance: Individual models demonstrate sophisticated cost-gradient sensitivity with economically rational behavior patterns. The MILP benchmark establishes the theoretical optimal threshold between Product 5 and Product 6, where the transition from Supplier 2 to Supplier 1 occurs. Notably, even at Product 5's theoretical break-even point, MILP maintains Supplier 1 preference, suggesting that when employing full ordering strategies, the benefits of lower fixed costs outweigh marginal unit cost advantages.

Vanilla PPO (Individual Model) shows early sensitivity to cost differentials, with premature switching beginning at Product 5 (0.79 ratio) compared to the optimal Product 6 threshold. This suggests a slight preference for lower fixed costs during the training period evaluated. PI-PPO demonstrates superior economic precision, closely following the MILP pattern with minimal early switching (0.05 at Product 5) and sharp transitions at the optimal threshold.

Shared Model Behavior: The shared model results reflect their incomplete training state rather than fundamental behavioral patterns. VI-PPO Shared Model shows systematic bias toward Supplier 2 across all configurations (ratios ranging from 0.08 to 0.13), representing behavior patterns that emerge during early training phases when the model has not yet learned to differentiate cost structures effectively. PI-PPO and Vanilla PPO Shared Models converge to approximately 0.58 supplier selection ratios across all products, indicating that at this training snapshot, the models have not yet developed the capacity to distinguish between different cost gradients - a capability that would likely emerge with extended training.

Comparison with Homogeneous Cost Structures The cost gradient results reveal several critical differences from the simpler homogeneous supplier

case, though these comparisons must account for the different training requirements:

Learning Complexity: Where homogeneous environments allowed shared models to achieve 98-100% optimal selection with proper pretraining, cost gradients require significantly longer training periods to develop comparable performance. This indicates that trade-offs present fundamentally more challenging learning problems than simple unit cost comparisons, requiring extended optimization to achieve convergence.

Training Stability: The homogeneous case showed policy degradation after episode 200 but maintained positive rewards throughout training. Cost gradient learning exhibits more complex training dynamics with extended periods of negative rewards in multi-task scenarios, suggesting that the interaction between fixed and variable costs creates optimization landscapes that require more sophisticated exploration strategies and longer convergence times.

Method Robustness: While pretraining methods provided clear advantages in homogeneous environments, their benefits require longer training horizons to manifest in gradient scenarios when applied to shared models. The training snapshots suggest that with sufficient training time, pretraining advantages may emerge, but the complexity of cost gradient learning necessitates extended optimization periods compared to simpler cost differentiation tasks.

The cost gradient experiments demonstrate that while reinforcement learning agents can develop sophisticated supplier selection strategies for individual products, the complexity of EOQ-based cost trade-offs requires significantly longer training periods for shared model approaches. The results suggest that current multi-task learning approaches can potentially address these challenges but require extended training horizons and potentially modified optimization strategies to achieve convergence in complex cost environments.

Chapter 6

Conclusion

This thesis investigated the application of reinforcement learning to inventory management optimization, focusing on pretraining methodologies and their application in stochastic and multi-supplier settings. Through systematic experimentation across single-supplier, multi-supplier, and multi-product scenarios, we provide insights into the practical viability of RL-based inventory management systems.

Key Findings and Research Contributions

Our research demonstrates that RL agents can successfully learn competitive inventory management policies for single-supplier environments, achieving performance comparable to optimal MILP baselines. This finding suggests that simplified training protocols can effectively prepare agents for real-world uncertainty while maintaining superior training stability.

The investigation of pretraining strategies revealed distinct advantages across different deployment scenarios. PI-PPO (behavior cloning from optimal MILP data) emerged as the most promising approach for multi-product deployment, offering both strong individual performance combined with relatively small variance. Value function initialization provided superior training stability but exhibited systematic bias in complex supplier selection scenarios. Combining both approaches resulted in good performance during evaluation, however training process was more instable.

Our multi-supplier experiments revealed a fundamental complexity hierarchy in inventory management learning. While agents successfully mastered simple cost-based supplier differentiation, achieving near-optimal selection rates in homogeneous environments, the transition to complex cost gradient scenarios involving trade-offs between fixed and variable costs proved significantly more challenging. This highlights that learning cost optimization represents a qualitatively different challenge compared to simple cost comparisons.

Implications and Broader Impact

These findings provide crucial insights for the practical deployment of RL in inventory management. The complexity challenges in multi-supplier scenarios underscore the need for extended training horizons and potentially modified optimization strategies when transitioning from research environments to practical deployment.

Limitations and Future Directions

Our experimental scope was constrained to controlled environments with moderate demand variability and simplified supplier configurations. Real-world procurement environments involve larger supplier networks, dynamic pricing structures, and higher uncertainty levels that extend beyond our current investigation. The incomplete convergence observed in complex multi-supplier scenarios indicates that practical deployment may require substantially greater computational resources and longer training periods than our experimental evaluation captured.

Future research should focus on several critical directions. Extended investigation of multi-supplier scenarios with longer training horizons could reveal whether shared models can eventually achieve convergence in complex cost environments. Training with more noisy demand patterns and more constrained suppliers would further test the robustness of RL approaches under realistic operational conditions. Additionally, exploring fully differentiable optimization approaches for multi-supplier inventory management could provide alternatives when training stability is paramount.

Concluding Remarks

While this work does not present a complete solution to inventory management optimization, it establishes essential groundwork for applying reinforcement learning to practical procurement scenarios. Our insights regarding training environment design, pretraining strategies, and multi-supplier complexity challenges provide a foundation for future research toward deployable RL-based inventory management systems. This research represents a meaningful step toward practical inventory management solutions, though addressing multi-supplier complexity remains a key challenge for future work.

Bibliography

[1] A. Oroojlooyjadid, M. Nazari, L. Snyder, and M. Takáč, "A deep q-network for the beer game: A reinforcement learning algorithm to solve inventory optimization problems," *arXiv preprint arXiv:1708.05924*, vol. 5, pp. 10–11, 2017.

[2] J. Gijsbrechts, R. N. Boute, J. A. Van Mieghem, and D. J. Zhang, "Can deep reinforcement learning improve inventory management? Performance on lost sales, dual-sourcing, and multi-echelon problems," *Manufacturing & Service Operations Management*, vol. 24, no. 3, pp. 1349–1368, 2022.

[3] H. Meisheri, S. Nath, M. Baranwal, and H. Khadilkar, "A learning based framework for handling uncertain lead times in multi-product inventory management," *arXiv preprint arXiv:2203.00885*, 2022.

[4] D. Madeka, K. Torkkola, C. Eisenach, A. Luo, D. P. Foster, and S. M. Kakade, "Deep inventory management," *arXiv preprint arXiv:2210.03137*, 2022.

[5] M. Alvo, D. Russo, and Y. Kanoria, "Neural inventory control in networks via hindsight differentiable policy optimization," *arXiv preprint arXiv:2306.11246*, 2023.

[6] M. Qi, Y. Shi, Y. Qi, C. Ma, R. Yuan, D. Wu, and Z.-J. Shen, "A practical end-to-end inventory management model with deep learning," *Management Science*, vol. 69, no. 2, pp. 759–773, 2023.

[7] B. J. De Moor, J. Gijsbrechts, and R. N. Boute, "Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management," *European Journal of Operational Research*, vol. 301, no. 2, pp. 535–545, 2022.

[8] M. Mousa, D. van de Berg, N. Kotecha, E. A. del Rio Chanona, and M. Mowbray, "An analysis of multi-agent reinforcement learning for decentralized inventory control systems," *Computers & Chemical Engineering*, vol. 188, p. 108783, 2024.

[9] N. Kotecha and A. del Rio Chanona, "Leveraging graph neural networks and multi-agent reinforcement learning for inventory control in supply chains," *Computers & Chemical Engineering*, p. 109111, 2025.

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

[11] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of machine learning research*, vol. 22, no. 268, pp. 1–8, 2021.

[12] N. Vanvuchelen, J. Gijsbrechts, and R. Boute, "Use of proximal policy optimization for the joint replenishment problem," *Computers in Industry*, vol. 119, p. 103239, 2020.

Appendix A

Baseline Environment Configuration

This appendix provides the complete baseline configuration used throughout the experimental evaluation. This configuration serves as the reference point for all ablation studies and comparative analyses presented in the thesis.

Parameter Specifications

Environment Parameters

- **Episode Length**: 100 decision periods per episode
- **Initial Inventory**: Starting inventory level uniformly sampled between 20–80 units
- **Holding Cost (h)**: 0.05 per unit per period (as defined in Equation 3.1)
- **Stockout Cost (b)**: 1.5 per unit of unmet demand
- **Safety Stock Target**: 10 units maintained as buffer inventory
- **Inventory Capacity**: Maximum allowable inventory of 1,000 units
- **State Normalization**: Enabled to improve learning stability across different cost regimes

Supplier Characteristics

- **Fixed Ordering Cost ($k_{v,t}$)**: 4 per order placement
- **Unit Cost ($c_{v,t}$)**: 0.5 per item ordered
- **Order Capacity ($M_{v,t}$)**: Maximum 350 units per order
- **Lead Time ($\hat{L}_v$)**: 4 periods (deterministic baseline)

Demand Characteristics

- **Demand Pattern (d_t)**: Constant 10 units per period
- **Forecast Quality**: Perfect forecasts matching actual demand
- **Demand Uncertainty**: None in baseline configuration

Future Work - more nuanced appraoch for underlaying the trainig

Baseline Rationale This baseline configuration represents a simplified but realistic inventory management scenario that captures the essential trade-offs between ordering, holding, and stockout costs while maintaining computational tractability for extensive experimental evaluation. The deterministic demand and lead times provide a controlled foundation for isolating the effects of individual experimental variations in subsequent ablation studies.

The parameter values are chosen to reflect realistic business conditions while ensuring numerical stability during reinforcement learning training.

Appendix B

Model Architecture

Neural Network Architecture

The reinforcement learning agent employs a shared feature extractor architecture implemented using the Proximal Policy Optimization (PPO) algorithm from Stable Baselines3 [11]. The network consists of three main components: a shared feature extractor for normalization, an independent policy network, and an independent value network. The shared feature extractor serves as a normalization layer for the input observations. From the normalized input, the architecture branches into two independent networks:

Policy Network:

- Two hidden layers with 64 neurons each and tanh activation
- One output layer that produces action probabilities matching the action space dimensionality

Value Network:

- Two hidden layers with 64 neurons each and tanh activation
- One output layer that predicts a single scalar value representing the value of the state

PPO Configuration

The PPO model was configured with the following hyperparameters:

- Learning rate: 0.003 (with linear decay to 0.001)
- Clip range: 0.1

- Steps per update: 1024
- Batch size: 64
- Training epochs per update: 10
- Discount factor (γ): 0.99

All other parameters used Stable Baselines3 default values.

Appendix C

Additional Plots

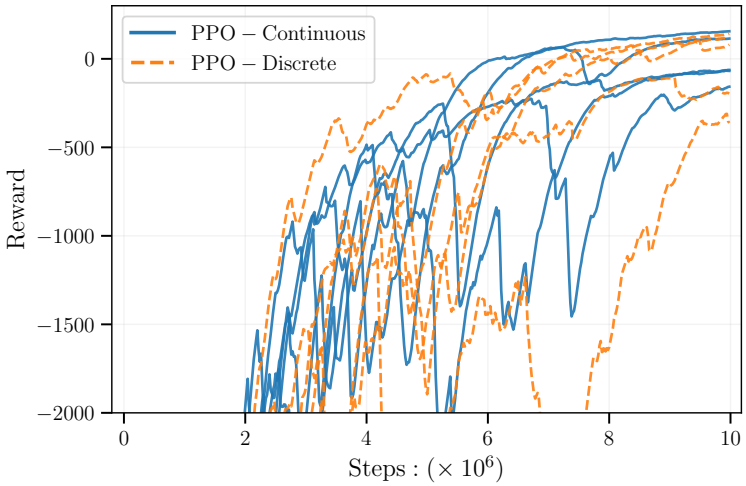


Figure C.1: PPO Training Performance Across Different Random Seeds in Continuous and Discrete Environments for multi-product environment.