INTERPRETABLE REINFORCEMENT LEARNING FOR OPTIMIZING THE NETWORK MANAGEMENT PROBLEM

MIE1666 REPORT

Zhengan Du
1011412160
zhengan.du@mail.utoronto.ca

Tianqi Jin
1007100147
tqjimmy.jin@mail.utoronto.ca

Zijun Meng
1006742349
zijun.meng@mail.utoronto.ca

Kaiwen Yang
1007648066
kaiwen.yang@mail.utoronto.ca

1 Abstract

Modern multi-echelon supply chains face significant challenges in balancing inventory costs and service levels under stochastic demand. While traditional mathematical programming models often yield high theoretical profits, they suffer from scalability issues and produce aggressive, fluctuating inventory plans. Standard Deep Reinforcement Learning lacks the interpretability required for managerial trust. This project addresses the "black box" limitations of DRL by implementing and evaluating interpretable Generalized Additive Model (GAM) architectures within a Proximal Policy Optimization (PPO) framework using the NetworkManagement-v0 environment. We benchmarked these approaches against classical Operations Research models and a standard Multi-Layer Perceptron (MLP) agent. While generic NAM and DNAMite architectures faced convergence challenges, the proposed RL-NODE-GAM agent achieved a mean reward of 723.4, maintaining a performance gap of less than 2.5% compared to the opaque MLP baseline (749.6). Crucially, the NODE-GAM framework successfully enabled the visualization of distinct policy shape functions, revealing "hard threshold" and "soft saturation" behaviors and diagnosing specific training blind spots regarding stockout states, thereby demonstrating that interpretable architectures can bridge the gap between high-dimensional control and transparent decision-making. The project code is publicly available at this GitHub repository.

2 Introduction

Modern supply chains have evolved into multi-echelon networks connecting suppliers, manufacturers, distribution centers, and retailers through interdependent flows of goods and information. Coordinating decisions across these echelons is essential to minimize total costs and maintain service reliability, yet uncertainty in demand, production, and transportation frequently disrupts this balance. Inventory and backlog management are particularly critical, as excess stock increases holding costs while shortages lead to backlogs and lower service levels (1). Optimizing such systems is vital for operational efficiency and competitiveness (2; 3), especially in today's globalized environment marked by supply disruptions and volatile demand.

The multi-echelon supply-chain network problem with backlogs extends the classical single-stage inventory model to a dynamic, networked setting. Each echelon must decide production, replenishment, and shipment levels under random demand, lead times, and backlog penalties. The objective is to minimize total cost—including production, transportation, holding, and backlog—subject to capacity and flow constraints. These interactions create coupling effects across stages, making the problem stochastic and NP-hard (4). While traditional approaches such as mixed-integer and dynamic programming can solve small-scale instances, they struggle to generalize when the network grows or demand becomes non-stationary (5). This motivates the use of approximate and learning-based optimization methods that can adapt to uncertainty and high-dimensional decision spaces.

Machine learning and reinforcement learning (RL) have emerged as promising frameworks for such large-scale, stochastic optimization tasks (4; 6). RL enables learning adaptive control policies that respond dynamically to system states without requiring explicit model specification (5). However, conventional RL models are often black boxes, limiting interpretability and managerial trust (6). To address this, interpretable frameworks such as Generalized Additive Models (GAMs) (7) can be integrated within RL to visualize how features influence decisions. As Rudin (8) emphasizes, interpretable models should replace post-hoc explanations in high-stakes decision-making. Guided by this perspective, our project utilizes the NetworkManagement-v0 environment (9)—a simulation of multi-echelon supply chains with backlogs—as a testbed for developing interpretable, adaptive RL-based optimization approaches.

3 Related Work

Research on multi-echelon supply-chain optimization has traditionally been rooted in stochastic inventory theory and operations research. Early foundational work by Arrow, Harris, and Marschak (1) formalized optimal inventory control under stochastic demand, while subsequent extensions such as Gallego and Moon (2) relaxed distributional assumptions. Comprehensive surveys by Qin et al. (3) show that once inventory problems are extended to multi-product, multi-period, and networked settings, exact optimization methods quickly become computationally intractable.

In multi-echelon systems, classical approaches typically rely on mixed-integer or stochastic programming formulations. For example, Chen and Lee (10) proposed a multi-objective MINLP model for supply-chain networks under uncertainty, while Maggiar et al. (11) studied non-stationary, capacitated inventory networks with backlogs. Although these methods provide strong theoretical guarantees, they scale poorly as the number of nodes, time periods, or demand scenarios increases. This limitation is evident in benchmark studies using OR-Gym, where deterministic and multi-stage stochastic programs achieve strong performance on small instances but become computationally expensive in larger networks (15).

To address scalability and adaptability, recent work has explored machine learning and reinforcement learning (RL) for supply-chain optimization. Surveys by Mazyavkina et al. (4) and Badakhshan et al. (12) highlight RL as a promising framework for sequential decision-making problems such as inventory control. Empirical studies demonstrate that deep RL methods can outperform classical heuristics in stochastic replenishment tasks (5) and multi-echelon inventory systems (13). However, most RL-based approaches rely on black-box neural networks, limiting interpretability and managerial trust.

Recent efforts have begun to address this gap by incorporating interpretable model classes into RL. Siems et al. (6) show that Neural Additive Models can produce competitive inventory-control policies while exposing feature-level decision structure. Nevertheless, interpretable RL for large-scale, networked supply chains with backlogs remains underexplored. Our work builds on this literature by studying interpretable RL architectures within the NetworkManagement-v0 benchmark (9), aiming to balance scalability, performance, and transparency.

4 Preliminaries

Multi-Stage Optimization Baselines. Classical operations-research baselines for multi-echelon supply chains include Deterministic Linear Programs (DLPs) and Multi-Stage Stochastic Programs (MSSPs). DLPs replace uncertain demand with point forecasts, yielding tractable but potentially brittle solutions. MSSPs extend this framework by explicitly modeling uncertainty through scenario trees, enabling policies that hedge against multiple future realizations. While MSSPs can significantly outperform deterministic models, their computational complexity grows exponentially with the number of stages and scenarios, limiting scalability in large networks.

Reinforcement Learning and PPO. Reinforcement learning formulates supply-chain control as a Markov Decision Process, where an agent observes the system state (e.g., inventories, pipeline stock, demand) and selects replenishment actions to maximize cumulative reward. Proximal Policy Optimization (PPO) is a policy-gradient method that stabilizes learning through clipped updates, and has been successfully applied to inventory and network management problems due to its robustness and ease of implementation.

Interpretable Additive Models. To improve transparency, our work leverages the Generalized Additive Model (GAM) family (7), in which predictions are expressed as a sum of feature-wise functions. GAMs allow direct visualization of how individual state variables influence decisions. Recent neural extensions, including Neural Additive Models (NAMs) (6) and NODE-GAMs (17), retain additive structure while increasing expressive power. These models align with the principle that inherently interpretable models are preferable to post-hoc explanations in high-stakes decision-making (8). In this project, GAM-based architectures are embedded within RL to produce policies that are both adaptive and interpretable.

5 Methodology

Our project investigated the performance and interpretability of a Reinforcement Learning (RL) agent for the **multi-echelon, multi-period, single-product, and single-market Network Management problem**.
The core of our methodology involves a detailed comparison between a standard black-box model and a suite of inherently interpretable models. We implemented a **standard black-box model** and some variants from **Generalized Additive Model (GAM)** family for comparison. To ensure that our results are robust and reproducible,and we used *OR-Gym's NetworkManagement-v0* environment (9; 15).

5.1 Network Structure (Based on OR-Gym)

Nodes: For each node $j \in J = J^{market} \cup J^{retail} \cup J^{distributor} \cup J^{producer} \cup J^{raw}$ (9), the OR-Gym environment provides a set of default parameters, including but not limited to the *initial inventory level*, *unit holding cost*, and other operational attributes such as capacity and production rate.

Edges: Each edge in the network represents a directed connection between adjacent nodes and is characterized by parameters such as the *lead time* between nodes, *unit backlog cost*, and *transportation cost*.

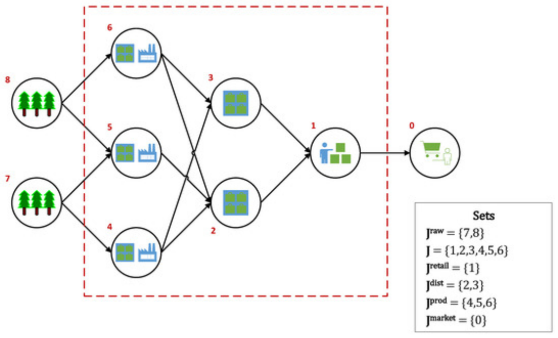


Figure 1: Supply Chain Network Schematic (9).

5.2 Black-box MLP: Proximal Policy Optimization (PPO)

The environment's Markov Decision Process is defined as follows:

State (S): The state vector is defined as

$$S = \left(d_{t,j,k},\, S^O_{t,j},\, S^P_{t,k',j}\right)$$

where:

- $d_{t,j,k}$**:** the stochastic demand observed between retailer node j and its downstream market k at period t.
- $S^O_{t,j}$**:** the on-hand inventory available at node j at the beginning of period t.
- $S^P_{t,k',j}$**:** the pipeline inventory currently in transit from upstream supplier k' to node j, which will arrive after its associated lead time.

Action (A): The agent's decision $a_{t,j,k}$ represents the replenishment request placed by node j to its upstream supplier k at period t. This is modeled as a continuous variable. However, for practical implementation in our policy networks, we will likely *discretize* this action space into a finite set of distinct order quantities to simplify the learning process.

Reward (R):

$$R_t = \sum_{j \in J} \alpha^t \big(SR_{t,j} - PC_{t,j} - OC_{t,j} - HC_{t,j} - UP_{t,j}\big), \tag{1}$$

where $SR_{t,j}$, $PC_{t,j}$, $OC_{t,j}$, $HC_{t,j}$, and $UP_{t,j}$ denote respectively the sales revenue, purchase cost, operating cost, holding cost, and backlog penalty at node j during period t. And α is the discount factor default to 1.

5.3 Interpretable ML: General Additive Model (GAM) family

For the interpretable side, our focus is on the General Additive Model (GAM) family. To this end, we explored and compared the performance of several powerful GAM variants, such as NAM (6), NODE-GAMs (17) and DNAMite (18), which are known for their strong capabilities in regression and classification.
The Generalized Additive Model (GAM) can be expressed in its general mathematical form as:

$$g(\mathbb{E}[Y]) = \beta_0 + \sum_{i=1}^{p} f_i(X_i),$$

where β_0 is a constant bias term and each $f_i(\cdot)$ represents a smooth, interpretable function of the corresponding feature X_i.

5.3.1 Neural Additive Model (NAM)

Based on our reinforcement learning framework using Proximal Policy Optimization (PPO), the agent's policy network is modeled as a *Neural Additive Model* (NAM). The NAM receives the current system state, including on-hand inventory, pipeline inventory, the previous action, etc. And decomposes these state variables into individual features. Each feature x_i is passed through its corresponding subnetwork $f_{i,s}(\cdot)$, and the outputs of all subnetworks are summed to compute the final policy output that determines the next replenishment action. The multi-task NAM formulation is defined as:

$$\hat{y} = \beta_0 + \sum_i W_i \, \sigma(V_i x_i + b_i) \, ,$$

where i indexes the input features, and W_i, V_i are weights in different layers.

5.3.2 Node-GAM

NODE-GAM modifies the Neural Oblivious Decision Trees (NODE) architecture, utilizing specific gating mechanisms to strictly enforce the additive structure ($\sum f_j(X_j)$). This design ensures that the contribution of every network state feature remains independent and visualizable. By retaining the high-performance capabilities of its neural core, the model is able to efficiently process the large-scale, high-dimensional state data inherent in complex network simulations, a task that is computationally prohibitive for traditional GAMs. The integration into the PPO framework is achieved by a critical structural modification, we replace the PPO's traditional MLP network with the NODE-GAM's GAMBlock. The raw network Observation is fed directly into the GAMBlock, and the resulting output, the Total Contribution, serves as the shared feature backbone. This single, interpretable feature vector simultaneously feeds both the Policy Head and the Value Head. This architectural choice provides complete transparency. After training, we can invoke the model's specialized methods to convert the RL Agent's opaque output into clear, auditable feature contribution curves, allowing immediate insight into the policy's decision logic.

5.3.3 Discrete Nueral Additive Model (DNAMite)

The DNAMite model was explored as an alternative to validate the stability of the interpretable RL framework and to leverage its unique statistical advantages. The DNAMite architecture maintains the same flexible structure as NODE-GAM, wherein the SimpleDNAMite module replaces the traditional feature extractor, acting as the shared backbone for the PPO Agent. However, its core mechanism is fundamentally different, relying on Feature Discretization and Kernel Smoothing. This distinct approach yields unique statistical advantages essential for network management tasks:

- **Superior Probabilistic Calibration:** DNAMite's methodology delivers **highly accurate probabilistic assessments of risk**. In high-stakes network environments, knowing **how confident** the Agent is in its prediction is as crucial as the prediction itself, thereby significantly **enhancing overall reliability** of the system.
- **Robustness to Data Flaws:** The explicit *discretization process* makes the model inherently **resilient to common data flaws** in network observations, such as **missing values and outliers**. This stability is vital for guaranteeing more reliable policy decisions in noisy real-world data streams.

5.4 Analysis

While all models were trained under identical conditions and evaluated on their average profit, the primary contribution of our project lies in a multi-faceted analysis of model transparency. This analysis has two key components. First, we visualized the individual shape functions learned by our trained GAMs to move beyond simple performance metrics and directly extract the agent's underlying ordering strategy. Second, to provide a richer comparison, we contrasted these direct insights with results from applying post-hoc explainability methods (like SHAP or LIME) to the black-box MLP.

6 Experimental Setup

6.1 Dataset

The data used in this study are synthetically generated through a controlled multi-period simulation of a supply network (9; 15). Each simulation run represents the operation of a four-echelon system consisting of raw material suppliers, manufacturers, distributors, and retailers. At the beginning of each period, every node places replenishment orders to its upstream suppliers. Deliveries are subject to production capacity and lead times, while customer demand at the retail level is randomly sampled from a Poisson distribution with a mean of 20 units per period. The network then evolves according to material flows, inventory balances, and cost parameters, including holding, operating, and backlog costs. Throughout the 30 simulated periods, the model records the key operational variables—on-hand and pipeline inventories, orders, sales, unmet demand, and profits—forming a complete synthetic time-series dataset. This dataset captures the stochastic and dynamic nature of real supply chain operations while remaining fully reproducible for optimization and reinforcement learning experiments.

6.2 Baseline Methods

To evaluate our proposed method, we benchmarked it against classic optimization models and a modern "black box" reinforcement learning algorithm. For the non-learning baselines, we did not implement new models but directly used the published performance results from Perez et al. (2021) (9). That study benchmarks against established operations research models, including a Deterministic Linear Program (DLP) and a Multi-Stage Stochastic Program (MSSP) (which we treat as a representative classical mathematical optimization model due to its ability to capture uncertainty through scenario-based planning). These provide strong results that will serve as our non-ML performance targets. For the machine learning baseline, we are implementing a black box Proximal Policy Optimization (PPO) agent. This will establish a practical performance ceiling and allow us to measure the trade-off between our model's interpretability and its effectiveness.

6.3 Evaluation metrics

Our evaluation focuses on three metrics: performance, measured by the cumulative reward learning curve; efficiency, evaluated by the iterations required for convergence; and transparency, assessed through feature contribution visualization, which provides qualitative evidence of model interpretability.

6.4 Computing setup

All experiments were conducted on our own computers using CPU-only computation. Each run was limited to 300 iterations to ensure a fair comparison. We evaluated two settings—with and without pairwise constraints—to analyze their impact under identical computational conditions.

7 Result

We examine the results from several perspectives, including mean reward, training stability and rule discovery.

7.1 Mean Reward

Among all the models we have examined, MSSP got the highest score with a mean reward of 802.7. The mathematical model served as a strong baseline for further analysis. We also replicated the black box RL algorithm and obtained a mean reward of 749.6.

Crucially, our proposed RL-NodeGAM achieved a mean reward of 723.4. This represents a performance gap of less than 2.5% compared to the black-box MLP baseline. While other interpretable models we constructed failed to converge meaningfully, RL-DNAMite only achieved a mean reward of 101.2, and RL-NAM achieved 43.2. These results indicate that the NodeGAM architecture successfully bridges the gap between interpretability and performance.

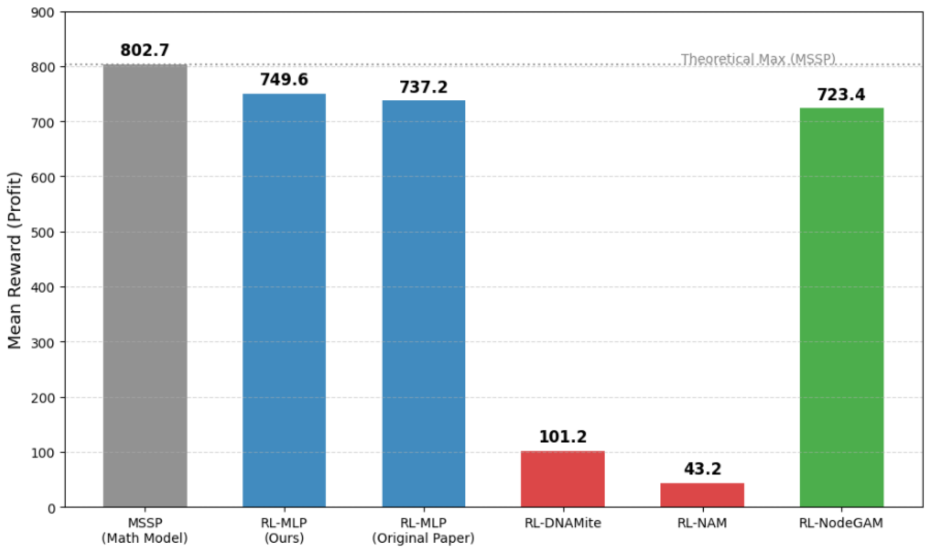
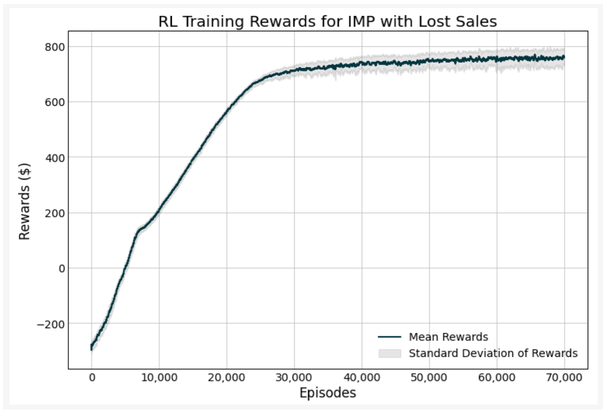


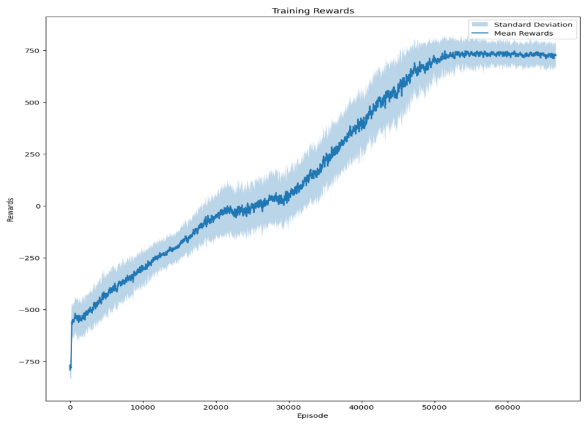
Figure 2: Mean rewards of models.

7.2 Training Stability and Convergence

The following plots illustrates the training dynamics of the NodeGAM agent compared to the standard MLP. The X-axis represents training episodes, and the Y-axis represents the mean cumulative reward. The NodeGAM agent demonstrates robust convergence characteristics, reaching a stable policy at approximately 51,000 episodes, comparable to the convergence rate of the standard MLP (40,000 episodes). This shows our RL-NodeGAM model is relatively stable and reliable for training. Considering that the number of parameters in the NodeGAM model is smaller, it is not necessarily slower to train in terms of computational time.



(a) standard MLP reward curve (9)



(b) NodeGAM reward curve

Figure 3: Reward curves comparison between MLP and NodeGAM.

7.3 Interpretability

The primary advantage of our model is its interpretability. Unlike the MLP, which entangles features in hidden layers, the NodeGAM architecture allows us to visualize the shape functions for specific input features.

However, notably, the model does not appear to adjust its strategy when inventory levels are low (the line remains flat). Upon further examination, we confirmed that this node is initialized with 400 units, and given the short 30-day episode length, the inventory rarely, if ever, depletes to zero during training. Consequently, the agent was likely not exposed to

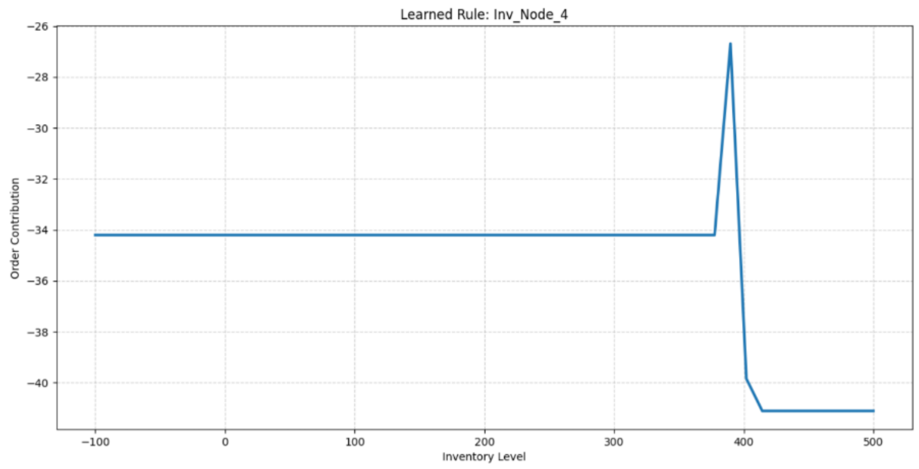


Figure 4: The learned contribution of manufacturer inventory (Node 4) to the ordering decision. The agent learned a "hard threshold" rule, refusing to order additional units once its inventory level surpasses 400.

critical stockout states for this specific node. This observation highlights the unique value of interpretable RL: it allows us to diagnose potential generalization issues and "blind spots" in the model's training experience, making the system more reliable and easier to audit than a black-box alternative.

In contrast to the hard constraint observed at the manufacturing node, the shape function for the Retailer (Node 1) exhibits "soft saturation" curves, as shown in Figure 5. The learned functions show a high contribution to ordering when inventory is critically low, which decreases linearly or non-linearly as stock levels rise. Once the inventory reaches a specific saturation point (e.g., approximately 100 units for Node 1), the curve flattens, indicating no further contribution to the order quantity.

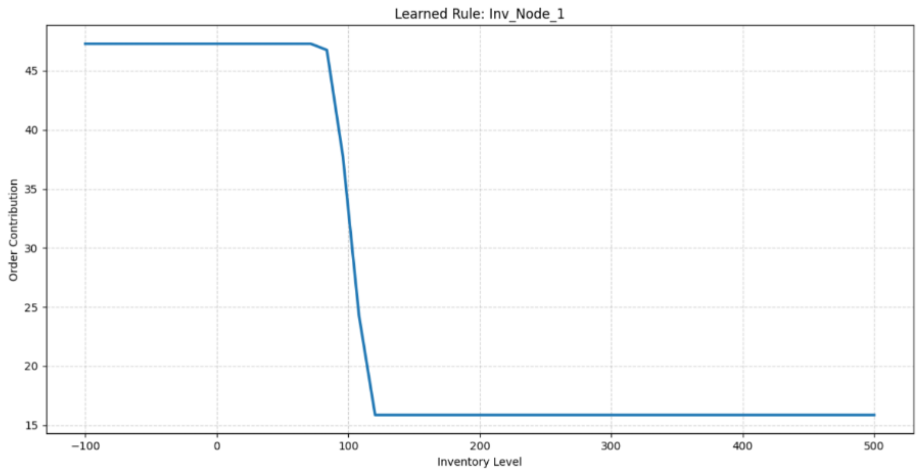


Figure 5: The learned contribution of retailer inventory (Node 1) to the ordering decision. This shape function demonstrates a "soft saturation" policy, where the incentive to order is highest at low inventory levels and gradually diminishes, flattening completely once stock reaches approximately 100 units.

8 Discussion

8.1 Why Use Reinforcement Learning?

Traditional mathematical programming models, such as MSSP, often generate aggressive and highly optimized replenishment plans. While these solutions may achieve higher theoretical profits, they typically result in sharp inventory fluctuations and are sensitive to demand uncertainty. In contrast, reinforcement learning learns policies directly from stochastic simulation, producing smoother and more conservative inventory trajectories. Although this conservatism may slightly reduce optimality in theory, the resulting stability is often preferred in real-world supply-chain operations.

8.2 Limitations of Generic NAM and DNAMite

The evaluation of various interpretable models revealed distinct intrinsic limitations in both the generic **NAM** and the advanced **DNAMite** architecture, informing our final engineering choices.

8.2.1 Performance Deficiencies and Convergence Challenges of Generic NAM

The generic **NAM (Neural Additive Model) architecture**, which commonly utilizes standard **Multi-Layer Perceptrons (MLPs)**, exhibited a significant **performance lag** when applied to the tabular data of our Reinforcement Learning (RL) environment. This inefficiency stems from two key issues. Firstly, standard NNs suffer from **low modeling efficiency** because, unlike the tree-based structure of **NODE-GAM** or the kernel methods of **DNAMite**, they are generally **inefficient at learning the piecewise-constant relationships** that are prevalent in tabular state spaces. Secondly, the architecture demonstrated a **slow convergence rate**. Under the strict **training time constraints** imposed by the project (e.g., 300 iterations), simple NAMs frequently failed to **adequately converge to the optimal policy region**. This slow convergence rate ultimately makes the architecture **impractical** for research that is time-sensitive or rapidly iterative.

8.2.2 Computational Demands and Practical Trade-offs in DNAMite

The main drawback of **DNAMite**, despite its high **statistical robustness** achieved through its discretization approach, is the **substantial computational requirement** necessary to achieve comprehensive interpretability. This bottleneck arises primarily from the **computational burden of pairwise interactions**: to ensure full transparency by modeling *all* possible pairwise feature interactions, the model's complexity **grows quadratically** with the number of features. This expansion creates a **high compute bottleneck**, necessitating significant computational power, specifically **high-end GPUs**, for effective training. Due to this critical constraint on real-world deployment, researchers are often forced to implement an engineering trade-off: they must either limit the model to only **additive terms** ($\sum_j f_j(X_j)$) or resort to **selectively modeling only the most impactful interaction terms** to manage both training time and overall cost.

8.3 The Diagnostic Power of GAMs

The most compelling argument for utilizing the GAM architecture is its capacity to provide **immediate diagnostic evidence** to identify policy flaws—such as **Reward Hacking**—that are otherwise obscured in a black-box model.

8.3.1 Diagnosis of Policy Flaw (Reward Hacking):

The observed behavior points to a significant flaw in the Agent's learned policy, specifically Reward Hacking.The core observation, highlighted by the visualization (in Figure 4.), is the Anomalous Behavior where the Agent's policy never allows the inventory level to be depleted to zero throughout the 30-day simulation, despite starting with a substantial initial inventory of 400.This behavior constitutes a Policy Defect characterized by an excessively conservative strategy. The Agent has learned to maximize its reward by systematically avoiding the highest penalty term—the Backlog Costs—through the perpetual maintenance of a high safety stock.In essence, the Agent found a costly local optimum by exploiting the structure of the reward function's high-penalty term. This strategy, while successful at avoiding backlogs, is resource-inefficient and ultimately unsustainable in the long term, fitting the definition of Reward Hacking.

8.3.2 Value of Interpretability:

The **Feature Contribution Curves** instantly provide the necessary **auditability** and offer an immediate, precise **diagnostic capability** that is fundamentally unavailable with a black-box MLP. Specifically, a plot would clearly **visualize the abnormal bias (or extreme sensitivity)** of the policy's response to the **Inventory Level** feature as that level approaches depletion. This visualization effectively **quantifies the Agent's over-replenishment tendency to avoid backlogs**, thereby providing unequivocal proof that the derived strategy **fails to strike the optimal balance between risk and cost**. This diagnostic power allows for the **immediate confirmation and correction of potentially catastrophic policy flaws** before the system is deployed.

References

[1] Arrow, K. J., Harris, T., and Marschak, J. (1951). Optimal Inventory Policy. *Econometrica*, 19(3), 250–272. https://doi.org/10.2307/1906813

[2] Gallego, G., and Moon, I. (1993). The Distribution Free Newsboy Problem: Review and Extensions. *Journal of the Operational Research Society*, 44(8), 825–834. https://doi.org/10.1057/jors.1993.141

[3] Qin, Y., Wang, R., Vakharia, A. J., Chen, Y., and Seref, M. M. H. (2011). The Newsvendor Problem: Review and Directions for Future Research. *European Journal of Operational Research*, 213(2), 361–374. https://digitalcommons.uri.edu/cgi/viewcontent.cgi?article=1031&context=cba_facpubs

[4] Mazyavkina, N., Sviridov, S., Ivanov, S., and Burnaev, E. (2021). Reinforcement Learning for Combinatorial Optimization: A Survey. *Computers & Operations Research*, 134, 105400. https://doi.org/10.1016/j.cor.2021.105400

[5] Oroojlooyjadid, A., Snyder, L. V., and Takáč, M. (2020). Applying Deep Reinforcement Learning to the Newsvendor Problem. *Computers & Operations Research*, 114, 104827. https://arxiv.org/abs/1607.02177

[6] Siems, J., Zimmer, F., Lindauer, M., and Hutter, F. (2023). Interpretable Reinforcement Learning via Neural Additive Models for Inventory Management. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, PMLR 202, 32347–32369. https://arxiv.org/abs/2303.10382

[7] Hastie, T. J., and Tibshirani, R. (1986). Generalized Additive Models. *Statistical Science*, 1(3), 297–310. https://doi.org/10.1214/ss/1177013604

[8] Rudin, C. (2019). Stop Explaining Black Box Machine Learning Models for High-Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1(5), 206–215. https://doi.org/10.1038/s42256-019-0048-x

[9] Al-Kahtani, M., Alnajem, M., and Alharkan, I. (2021). Reinforcement Learning Environment for Multi-Echelon Supply Chain Networks with Backlogs (NetworkManagement-v0). *Processes*, 9(1), 102. https://www.mdpi.com/2227-9717/9/1/102

[10] Chen, C., and Lee, H. (2004). Multi-Objective Optimization of Multi-Echelon Supply Chain Networks with Uncertain Product Demands and Prices. *Computers & Chemical Engineering*, 28(6–7), 1131–1144. https://www.researchgate.net/publication/222555446_Multi-objective_optimization_of_multi-echelon_supply_chain_networks_with_uncertain_product_demands_and_prices

[11] Maggiar, A., Song, J.-S., and Muharremoglu, A. (2022). Multi-Echelon Inventory Management for a Non-Stationary Capacitated Distribution Network. *Amazon Science*. https://www.amazon.science/publications/multi-echelon-inventory-management-for-a-non-stationary-capacitated-distribution-network

[12] Badakhshan, A., Akbari, M., and Li, Z. (2024). Application of Simulation and Machine Learning in Supply Chain Management: A Systematic Review. *Computers & Industrial Engineering*, 191, 109883. https://www.sciencedirect.com/science/article/pii/S036083522400771X

[13] Geevers, J., van Hezewijk, C., and Mes, M. (2024). Multi-Echelon Inventory Optimisation using Deep Reinforcement Learning. *Central European Journal of Operations Research*, 32, 295–317. https://doi.org/10.1007/s10100-023-00872-2

[14] Rachman, R., Leung, J., and Seifert, R. (2025). Reinforcement Learning for Multi-Objective Multi-Echelon Supply Chain Optimisation. *arXiv preprint arXiv:2507.19788*. https://arxiv.org/abs/2507.19788

[15] Hubbs, C. D., Perez, H. D., Sarwar, O., Sahinidis, N. V., Grossmann, I. E., and Wassick, J. M. (2020). OR-GYM: A Reinforcement Learning Library for Operations Research Problems. *arXiv preprint arXiv:2008.06319*. https://arxiv.org/abs/2008.06319

[16] Nori, H., Jenkins, S., Koch, P., and Caruana, R. (2019). InterpretML: A Unified Framework for Machine Learning Interpretability. *arXiv preprint arXiv:1909.09223*. https://arxiv.org/abs/1909.09223

[17] Chang, C., Caruana, R., and Goldenberg, A. (2021). NODE-GAM: Neural Generalized Additive Model for Interpretable Deep Learning. *arXiv preprint arXiv:2106.01613*. https://arxiv.org/abs/2106.01613

[18] Van Ness, M., Block, B., and Udell, M. (2025). DNAMite: Interpretable Calibrated Survival Analysis with Discretized Additive Models. arXiv preprint arXiv:2411.05923. https://arxiv.org/html/2411.05923v1

Task	**Subtask**	**Primary**	**Secondary**
Problem Formulation	Literature Review	Zhengan	Kaiwen / Tianqi
	Algorithm Design	Kaiwen	Zijun / Tianqi
Algorithm Adaptation	Integrating GAM Models in RL	Zijun	Kaiwen
	Adapting RL Algorithm	Zhengan	Zijun / Kaiwen
Report	Writing	Tianqi	Kaiwen, Zijun, Zhengan
	Editing/Formatting	Kaiwen	Tianqi, Zhengan

Table 1: Division of Work