Engineering Journal issn: 0125-8281

Article

Employing Neuroevolution of Augmenting Topologies (NEAT) in Linear Multi-Echelon Inventory Systems

Yanvaroj Pongsethpaisal[a], Naragain Phumchusri[b,*], and Paveena Chaovalitwongse[c]

Department of Industrial Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
E-mail: [a]6071463221@student.chula.ac.th, [b,*]naragain.p@chula.ac.th (Corresponding author), [c]paveena.c@chula.ac.th

Abstract. Reinforcement learning has emerged as a leading algorithmic approach due to its successful applications across various domains. While many implementations favour the model-free approach for its aptitude for handling complex problems, its learning curve tends to be slower. Given the intricacies of the Linear Multi-Echelon Inventory System, a model-based approach might be more fitting, offering faster learning rates. This study seeks to integrate Neuroevolution of Augment Topologies (NEAT) – a hybrid of model-based reinforcement learning and evolutionary algorithms – into such an inventory system. Furthermore, the research delves into hyperparameter tuning, experimenting with seven specific hyperparameters to discern the most efficient combination and understand their interplay. Benchmarking against the model-free Proximal Policy Optimisation (PPO) serves as a measure of NEAT's effectiveness. Findings indicate that when optimally tuned, NEAT can slash total costs by 25.02% compared to PPO. Impressively, NEAT achieves this peak performance in a mere 1,000 generations, significantly outpacing PPO's learning trajectory.

Keywords: Inventory control, reinforcement learning, genetic algorithms, neuro evolutionary augment topology.

ENGINEERING JOURNAL Volume 29 Issue 3
Received 25 July 2024
Accepted 5 March 2025
Published 31 March 2025
Online at https://engj.org/
DOI:10.4186/ej.2025.29.3.11

1. Introduction

Reinforcement learning (RL), a pivotal segment of machine learning, has seen progress in areas like robotics and predictive analytics [1, 2, 3]. RL frequently applies Deep Learning (DL) techniques, known as Deep Reinforcement Learning (DRL), to manage complex issues and learn effectively from rewards. However, its significant data requirements pose a challenge. Simulation models can mitigate this data hunger by generating necessary training data. RL is categorised into model-based, efficient for simple tasks, and model-free, suitable for intricate situations but slower to adapt.

In the domain of inventory control, RL adapts to various complexities, influenced by system structure, costs, demand, and lead times. Studies highlight the application of both model-free [4, 5] and model-based algorithms [6] to tackle different levels of inventory complexity. [5] specifically delved into the model-free Proximal Policy Optimisation (PPO) algorithm, introduced by [7], for its capacity to handle complex challenges or scenarios where full simulation is impractical due to less reliance on the simulation model. Employing the Linear Multi-Echelon Inventory System (LMEIS) framework by [4, 5] applied PPO to a system encompassing five levels—supplier, manufacturer, distributor, retailer, and customer—each with a single stock point as illustrated in Fig. 1. Customers at level 0 initiate the replenishment process by requesting stock from the subsequent upper level. This operational model unfolds in discrete time steps, marked by four pivotal events: receiving the previous order from the upper level, obtaining a new order from the lower level, order fulfillment, and the agent's decision-making on whether a new order is warranted.

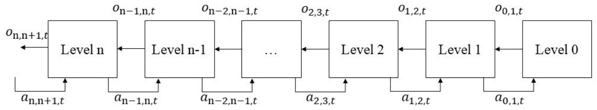


Fig. 1. Example of a linear multi-echelon inventory system.

On the other hand, model-based algorithms excel in efficiency, achieving results on par with model-free methods in fewer cycles, thanks to their adaptability. This proves especially advantageous for streamlined LMEIS inventory management. [8] showcased the success of Neuroevolution of Augmenting Topologies (NEAT) in situations requiring strategic decision-making, such as pole-balancing and Atari games, where it significantly improved performance and reliability. The significance of past actions in inventory systems highlights NEAT's

Nevertheless, hyperparameter tuning is critical since no single RL algorithm fits every situation. This study focuses on NEAT to investigate how model-based RL can tackle the complexities of LMEIS. With limited discussion in existing literature, examining NEAT's application, the effects of hyperparameter Optimisation, and its comparison with traditional methods opens up promising avenues for research.

The paper is structured as follows: Section 2 explores recent developments in RL across various fields. Section 3 details the inventory system, its definitions, assumptions, and methodology. Section 4 addresses tuning of the NEAT algorithm's hyperparameters. Section 5 assesses the algorithm's performance and benchmarks it. The conclusion in Section 6 summarises key findings and suggests directions for future research.

2. Literature Review

This section briefly reviews related research on RL, focusing on two key areas: 1. Applying model-based RL and integrating meta-heuristics into various fields. 2. Development and use of RL in inventory control research.

2.1. Model-Based Reinforcement Learning (MBRL)

Researchers widely recognised RL's ability to address complex challenges. [9] discussed the diverse applications of model-based and model-free RL. [10] presented the Cluster-Explore Classify-Exploit (CECE) framework for clustering, adaptable across domains. [11] used RL for context-based change detection, tested in ball catching and traffic scenarios. [12] employed a Markov decision process (MDP) for demand distribution changes in inventory. For decision-making, researchers like [13, 14, 15, 16, 17, 18, 19] applied MDP in route planning and robotic movements.

These studies span various domains, including robotics [20, 19, 16, 18, 15, 14], where they have been applied to tasks like balance control, motion planning, and robotic arm trajectories. RL algorithms have been used in logistics for route and path planning [21, 17, 22]. However, a key challenge in implementing RL lies in hyperparameter tuning. Several studies have focused on developing algorithms specifically for adjusting hyperparameters, such as those by [12, 23]. It is important to note that the choice of hyperparameters is data-dependent [13], requiring each research study implementing RL to conduct experiments to identify the most suitable hyperparameter configuration.

As [24] stated, the RL nature of data-hungry in the training phase might limit itself from a real-world problem and large state spaces. Therefore, one of the ways out is to hybridise it with other well-known concepts. These evolutionary algorithms help in their data-hungry nature, called Evolutionary Algorithms (EA) for RL (EARL). An evolutionary algorithm for learning algorithms can be implemented in the process [25, 26] or parameter tuning [27]. For instance, [24] reviewed the literature on EARL-type implementation in various fields such as business management, medical, industrial, or image processing. Conversely, some papers, such as [12], implemented EA for parameter tuning, and [27] applied their research to detect fraud within cryptocurrency networks.

NEAT was chosen for its ability to evolve both network topologies and weights, which allows it to dynamically adapt to the complexity of multi-echelon inventory systems. Unlike DQN, which relies on fixed network architectures that may not fully capture varying system dynamics, NEAT optimizes both structure and parameters simultaneously. Additionally, NEAT builds on the core principles of Genetic Algorithms (GA), such as mutation and crossover, but extends them by evolving architectures alongside weights. This combination makes NEAT particularly effective for handling the non-linear, multi-level interactions in dynamic, high-dimensional inventory problems [28].

One of the concepts is combining neural network development with the Genetic Algorithm (GA) named NEAT, introduced by [8]. This method continuously improves the neural network structure and weight through evolutionary processes. In addition, this method has shown promising results throughout various research fields, e.g., the medical field [29, 30], playing a game [31], or scheduling problems [32].

2.2. Inventory Control

In inventory control research, the complexity of the system can be gauged through various factors, as recommended by [33, 34]. These factors include the number of stages, time horizon, node connections in each stage, uncertainty, performance criteria, and other constraints, which help classify the system's complexity. Regarding solution approaches, techniques for inventory control problems, like many computational problems, have evolved. They initially used exact solutions like the EOQ model [35, 36] or mathematical models [37]. Researchers introduced approximation methods [38] as problems required longer computation times.

As the problem grew more computationally demanding, heuristic methods emerged as popular solutions, such as the greedy approach [39] and simulation-based methods [40, 41, 42]. These heuristics further evolved into well-known meta-heuristics, many drawing inspiration from natural processes. [43] conducted a review of these nature-inspired meta-heuristics, including GA [44, 33, 45, 46, 47], particle swarm optimisation (PSO) [48, 49, 50, 51], and EA [52].

Machine learning has gained popularity in addressing inventory challenges. [53] applied supervised machine learning for inventory classification, but most algorithms focus on cost minimisation in inventory control. [54, 55] explored the implementation of RL in inventory control, while [56] summarised literature using deep Q-learning for Multi-Echelon inventory control, citing [4, 5]. [57] employed Asynchronous Advantage Actor-Critic (A3C), and [5, 58] chose PPO for inventory systems. However, as highlighted by [54], most implemented learning is model-free, leading to slow learning convergence [54, 18]. Table 1 compares these studies in terms of sourcing, echelons, retail, inventory structure, cost considerations, uncertainty sources, constraints, and methodologies.

In this study, we employ NEAT algorithm, a hybrid MBRL and EA, which has demonstrated promise across different research areas but has not been applied to the LMEIS as proposed by [4]. The PPO algorithm, studied by [5], is used as a benchmark with an objective of minimising total costs. Additionally, how hyperparameter tuning influences NEAT's performance are investigated. This point has not previously been explored in the LMEIS literature.

3. Model Formulation

The inventory system focused in this paper is explained in this section, which is divided into two main parts: 1. inventory system and its simulation modelling, and 2. DRL formulation.

3.1. Inventory System and Simulation Modelling

The study adopts a LMEIS model from [4], characterised by five echelons and centralised information, as depicted in Fig. 2. Customer orders at the first echelon are generated each period, adhering to a uniform distribution between 0 and 15 units, with immediate fulfilment required to avoid backlogs. To meet this demand, each subsequent echelon, spanning a total of five levels, requests inventory from the level above it. Inventory transfers are governed by a lead time, which is determined randomly from a uniform distribution of 0 to 4 periods for each time period and applies to all stock levels except the first within that timeframe.

Table 1. Literature comparison.

Study	Source			Echelon			Retail		Structure				Performance (Cost)				Uncertainty			Constraint		Methodology				
	1	2	n	1	2	n	1	n	1 to 1	Many to 1	1 to Many	Many to Many	Ordering	Unit	Holding	Penalty	Demand	Lead time	Cost	Backlog	Lost sales	Exact method	Heuristic	Metaheuristic	Model-free RL	Model-based RL
Maity and Maiti (2005)	x			x			x		x					x	x				x				x			
Katsaliaki and Brailsford (2007)	x				x		x		x				x		x	x	x			x			x			
Yandra et al. (2007)			x			x		x				x	x		x		x							x		
Khanlarzade et al. (2012)					x		x		x				x		x	x			x	x				x		
Park and Kyung (2014)	x					x	x		x						x	x	x			x				x		
Chaharsooghi et al. (2008)	x					x	x		x						x	x	x	x		x					x	
Gijsbrechts et al. (2019)		x				x	x			x					x	x	x				x				x	
Gharaei et al. (2019)	x					x		x			x		x	x	x				x			x				
Geevers (2020)	x					x	x		x						x	x	x	x		x					x	
Kaynov (2020)	x				x			x			x				x	x	x			x	x				x	
This research	x					x	x		x						x	x	x	x		x						x

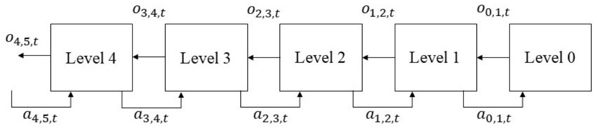


Fig. 2. Inventory system considered in this research.

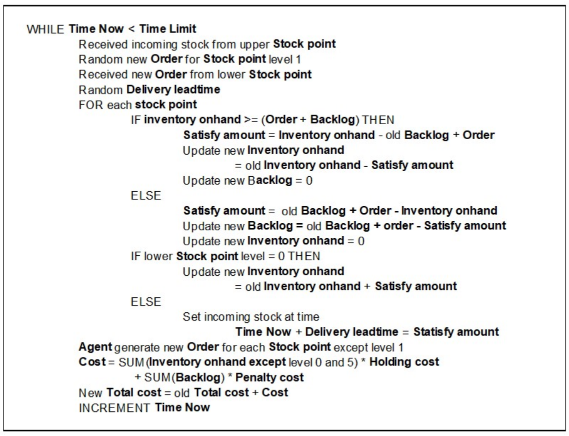
Table 2. Inventory system parameters.

Parameters	Level				
	0	1	2	3	4
Demand Distribution (Unit)	Uni[0,15]	-	-	-	-
Lead time (Period)	-	Uni[0,4]	Uni[0,4]	Uni[0,4]	Uni[0,4]
Holding cost ($ / unit / period)	0	1	1	1	1
Penalty cost ($ / unit / period)	2	2	2	2	2

This setup illustrates how, for example, a generated lead time of 1 period means any transport initiated across the supply chain concludes in the following period. Conversely, a lead time of 3 periods causes transfers to reach their destinations after three periods. At the end of each period, holding and penalty costs are calculated at $1 and $2 per unit, respectively, for stored and backlogged units. System parameters for each level are detailed in Table 2. This process repeats over 35 periods to complete a single experimental replication, with its operational logic succinctly illustrated through pseudocode in Fig. 3.

Following the operational details outlined, a mathematical model is introduced to succinctly represent this LMEIS. This model captures the essence of customer demand, inventory transfers, and cost calculations, as previously described. It offers a quantitative foundation for analysing the system's dynamics over the 35-period experimental replication.

$$minimise \sum_{t=1}^{n} \sum_{i=1}^{m} (h_i x_{i,t} + s_i c_{i,t}) \tag{1}$$



```
WHILE Time Now < Time Limit
    Received incoming stock from upper Stock point
    Random new Order for Stock point level 1
    Received new Order from lower Stock point
    Random Delivery leadtime
    FOR each stock point
        IF inventory onhand >= (Order + Backlog) THEN
            Satisfy amount = Inventory onhand - old Backlog + Order
            Update new Inventory onhand
                = old Inventory onhand - Satisfy amount
            Update new Backlog = 0
        ELSE
            Satisfy amount =  old Backlog + Order - Inventory onhand
            Update new Backlog = old Backlog + order - Satisfy amount
            Update new Inventory onhand = 0
        IF lower Stock point level = 0 THEN
            Update new Inventory onhand
                = old Inventory onhand + Satisfy amount
        ELSE
            Set incoming stock at time
                Time Now + Delivery leadtime = Statisfy amount
    Agent generate new Order for each Stock point except level 1
    Cost = SUM(Inventory onhand except level 0 and 5) * Holding cost
          + SUM(Backlog) * Penalty cost
    New Total cost = old Total cost + Cost
    INCREMENT Time Now
```

Fig. 3. Pseudocode for inventory system simulation.

Such that:

$$p_{i,t} = p_{i,t-1} + o_{i,j,t} - o_{i-1,i,t}; \quad \forall i \in m \tag{2}$$

$$\begin{aligned} x_{i,t} = \max \big(0, x_{i,t-1} + a_{i,i+1,t} \\ - o_{i-1,i,t} - c_{i-1,t}\big); \quad \forall i \in m \end{aligned} \tag{3}$$

$$\begin{aligned} c_{i,t} = \big|\min \big(0, x_{i,t-1} + a_{i,i+1,t} - o_{i-1,i,t} \\ - c_{i-1,t}\big)\big|; \quad \forall i \in m \end{aligned} \tag{4}$$

where:

- m - Level of supply chain member (1, 2, 3, 4)
- n - Time period for the simulation (1, 2, 3, ..., 35)
- $x_{i,t}$ - Inventory on-hand of level i at the end of time t
- $p_{i,t}$ - Inventory position of level i at the end of time t
- $c_{i,t}$ - Shortage amount of level i at the end of time t
- h_i - Holding cost per unit at level i ($/unit)
- s_i - Shortage cost per unit at level i ($/unit)
- $o_{i,j,t}$ - Order from level i to level j at time t and $j = i + 1$
- $a_{i,j,t}$ - Arriving stock of level i from level j arrives at time t and $j = i + 1$

In this model, Equation (1) represents the objective, which is set up to minimize the total holding and penalty costs at each echelon, except at the customer level, for every time period. Equations (2) and (3) are constraints to ensure that the inventory position and inventory on-hand at each level i are updated correctly. The last constraint, shown in Equation (4), captures any shortages in inventory at each level.

3.2. Inventory System and Simulation Modelling

To embed DRL within the simulation model, the ”OpenAI Gym” framework, a prevalent Python library for integrating RL into various environments, is utilised. This framework offers a systematic methodology for crafting RL agents and systems, making the integration process more manageable. However, it requires users to build RL agents and systems on their own. Also, ”NEATPython” is tailored to fit the simulation environment’s specific needs, including state inputs and outputs. Importantly, it is vital to initially outline the core components of RL, such as the state space, action space, and rewards. The detail on these crucial aspects is provided in the next section.

3.2.1. State space

In RL, the ’state’ represents system information at each point in time, like inventory position, on-hand inventory, and backlog. The ’state’ is crucial to the learning model as inadequate information limits the learning agent’s capabilities. Thus, it is essential to design a state encapsulating all vital information. The state space for this research is defined as follows:

$$S(t) = [t_{x,t}, t_{b,t}, x_{i,t}, c_{i,t}, o_{i,j,t-1}, \\ a_{i,j,t}, a_{i+1,j,t}, a_{i+2,j,t}, a_{i+3,j,t}, a_{i+4,j,t}] \quad (5)$$

State $S(t)$ comprises 10 primary dimensions, with txt denoting total inventory, $t_{b,t}$ representing total backlog, $x_{i,t}$ indicating inventory per level, $c_{i,t}$ signifying each level’s order backlog, and $o_{i,j,t-1}$ being the recent request per level. The parameters $a_{i,j,t}$, $a_{i+1,j,t}$, $a_{i+2,j,t}$, $a_{i+3,j,t}$, and $a_{i+4,j,t}$ correspond to incoming inventory over 0, 1, 2, 3, and 4 periods at each level, totalling 30 parameters. Each parameter has a minimum value of zero, while their maximum values vary.

3.2.2. Action space

At the period’s conclusion, the formulated learning model determines whether a request to the upper level is requisite. Given the environmental setup, each level can request a maximum of 30 units per period. While the learning model operates in continuous space for action output, it necessitates rounding before initiating requests.

3.2.3. Reward definition

Due to the GA-based concept in NEAT, the reward is redefined as fitness, tied to total costs of holding and shortages. Balancing these costs is critical, as excessive holding reduces inventory turnover. From Table 2, holding beyond two periods is less cost-effective, making penalties a wiser choice. The reward function is designed to be multi-objective, prioritizing penalties while maintaining reasonable inventory levels.

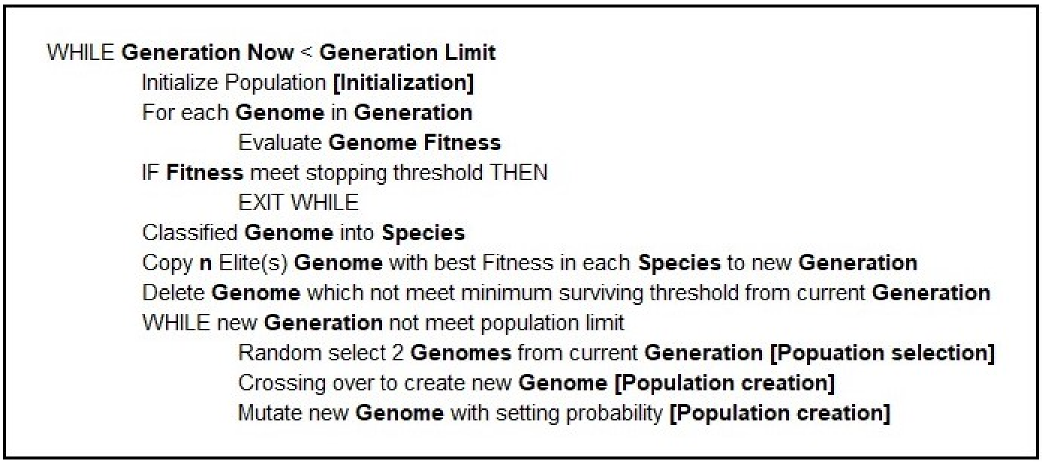
3.3. DRL Agent Training

3.3.1. NEAT explanation

The NEAT algorithm fundamentally melds GA with RL processes to formulate a neural network that serves as an agent in the RL system. GA is utilised for encoding neural networks, selecting parents, and creating populations, ensuring its effective operation. Moreover, the NEAT algorithm, grounded in the concept of species grouping, guarantees adequate population diversity in each generation while mitigating stagnation. Fig. 4 delineates the NEAT procedure through pseudocode.

3.3.2. Encoding

The encoding method encrypts node connections, comprising connection status and genes of two connect-



```
WHILE Generation Now < Generation Limit
        Initialize Population [Initialization]
        For each Genome in Generation
                Evaluate Genome Fitness
        IF Fitness meet stopping threshold THEN
                EXIT WHILE
        Classified Genome into Species
        Copy n Elite(s) Genome with best Fitness in each Species to new Generation
        Delete Genome which not meet minimum surviving threshold from current Generation
        WHILE new Generation not meet population limit
                Random select 2 Genomes from current Generation [Popuation selection]
                Crossing over to create new Genome [Population creation]
                Mutate new Genome with setting probability [Population creation]
```

Fig. 4. Pseudocode for NEAT

ing nodes, as shown in Fig. 5. This figure illustrates a neural network example encoded into the genome in Fig. 6, where each gene reveals connection number, node linkage, and status.

3.3.3. Initialisation

The neural network begins with only NEAT's initial input and output nodes. There are three initialisation types: full direct, partial direct, and combined. In full direct, every input and output are fully connected with randomised weights. Conversely, partial direct determines the existence of connections between inputs and outputs probabilistically; if a connection is established, the system then assigns its weight.

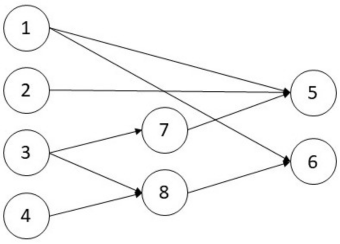
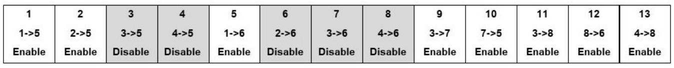


Fig. 5. Example of developing a neural network.



1	2	3	4	5	6	7	8	9	10	11	12	13
1->5	2->5	3->5	4->5	1->6	2->6	3->6	4->6	3->7	7->5	3->8	8->6	4->8
Enable	Enable	Disable	Disable	Enable	Disable	Disable	Disable	Enable	Enable	Enable	Enable	Enable

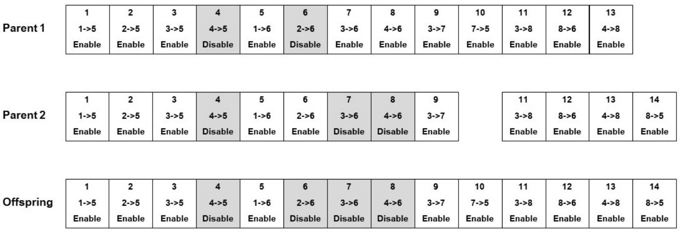
Fig. 6. Encoded genome.

3.3.4. Parental selection

In the GA, parent selection, necessary for generating the next generation population, can be accomplished using various methods at the end of each generation. These methods include basic n-best selection and tournament selection. Tournament selection is used in this study for its unique ability to randomly segment the population into multiple tournaments, preserving diversity and avoiding local optima. Unlike conventional methods that randomly select contenders in each tournament, this research categorises tournaments by species, choosing the n-best from each. These selected individuals are then sorted and subjected to n-best selection again.

3.3.5. Population creation

After selecting parents, the process sequentially matches and mutates the selected parents to create a population. During matching, genomes are combined. If a genome is present in both parents and is disabled in either or both, it will also be turned off in the offspring. Fig. 7 illustrates this: although connection six is enabled in parent two, it is disabled in the offspring since it is disabled in parent one. The offspring inherits connection ten from parent one and fourteen from parent two.



Parent 1	1 1->5 Enable	2 2->5 Enable	3 3->5 Enable	4 4->5 Disable	5 1->6 Enable	6 2->6 Disable	7 3->6 Enable	8 4->6 Enable	9 3->7 Enable	10 7->5 Enable	11 3->8 Enable	12 8->6 Enable	13 4->8 Enable	
Parent 2	1 1->5 Enable	2 2->5 Enable	3 3->5 Enable	4 4->5 Disable	5 1->6 Enable	6 2->6 Enable	7 3->6 Disable	8 4->6 Disable	9 3->7 Enable		11 3->8 Enable	12 8->6 Enable	13 4->8 Enable	14 8->5 Enable
Offspring	1 1->5 Enable	2 2->5 Enable	3 3->5 Enable	4 4->5 Disable	5 1->6 Enable	6 2->6 Disable	7 3->6 Disable	8 4->6 Disable	9 3->7 Enable	10 7->5 Enable	11 3->8 Enable	12 8->6 Enable	13 4->8 Enable	14 8->5 Enable

Fig. 7. Example of the matching process.

Upon completion of the matching phase, the offspring are subject to mutation. NEAT incorporates six mutation methodologies: addition and disabling of nodes, addition and disabling of connections, connection weight mutation, and connection weight replacement, each occurring with an independent probability.

4. Numerical Study

Having elucidated both principal components in the preceding section, ensuring their proper function requires additional steps. Although DRL appears poised to address the problem, algorithm tuning is imperative. Consequently, this section will elaborate on integrating NEAT into the inventory system and deriving appropriate hyperparameters.

4.1. Objective

Several crucial reasons necessitate the adjustment of NEAT hyperparameters for specific cases. Firstly, the unique nature of the problem influences the requisite changes to parameters. Secondly, the algorithm modifies its parameters to maximise expected rewards, but the default parameters might not always be optimal for specific tasks. Lastly, the circumstances of the problem also dictate the necessary parameter adjustments.

Therefore, it is often necessary to fine-tune the parameters to align with the specific environment to optimise the results derived from the RL algorithm. Hyperparameters are adjusted to identify a set that minimises inventory cost within the given environment.

4.2. Methodology

Typically, DRL necessitates a feedback loop integrating environment and agent, consisting of state, action, and reward, as illustrated in Fig. 8. Within the context of the LMEIS described in Section 3.1, the state encompasses 10 main dimensions, reflecting the intricacies of our inventory management process across the five echelons and centralised information system. The action corresponds to each level's request to the upper level for inventory replenishment, a critical component of our model that ensures efficient stock flow and minimises delays in order fulfillment. This study employs total holding cost and penalty as rewards to gauge agent performance, directly aligning with our inventory system's objectives to optimize stock levels and reduce associated costs. By applying these metrics, we can precisely assess the effectiveness of the RL agent in managing the complex dynamics of the described inventory system.

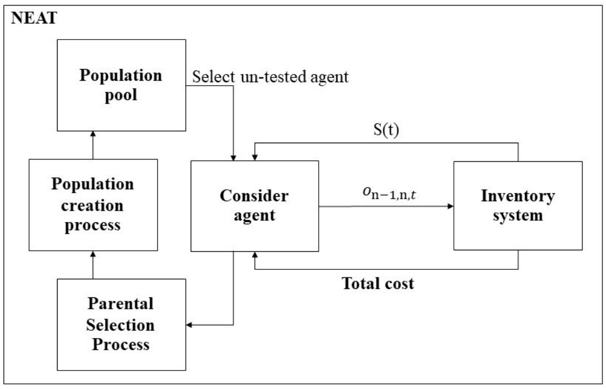


Fig. 8. Integrated mechanism.

Hyperparameters must first be scoped to identify the combination optimising the specific environment's solution. Given the NEAT algorithm's needs, about 20 hyperparameters require tuning. Since exhaustive experimentation on all hyperparameters (most continuous between 0 and 1) is impractical, they will be scoped into high-impact ones, categorised into levels, and examined through total enumeration testing. Relying on [8] to sift through low-impact hyperparameters, the focus will be on seven high-impact ones: population size per generation, neural network activation function, generation limit, probabilities of node insertion/deletion, link weight replacement, link weight mutation, and minimum species per generation.

4.3. Hyperparameter Evaluation and Adjustment

This section focuses on exploring effective combinations of hyperparameters and analysing their trends and potential impacts. The seven selected hyperparameters are categorised into distinct levels, reflecting their unique characteristics as outlined in Table 3.

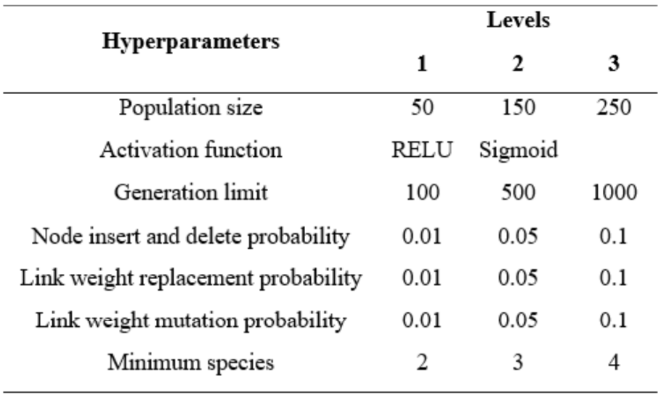
These hyperparameters and their respective levels have been extensively investigated and utilised in various studies [59, 60, 61, 62], highlighting their significance and effect on our analysis. Through this exploration, we aim to identify synergistic hyperparameter configurations that enhance performance, acknowledging the challenge of finding highly effective combinations.

The identified seven hyperparameters, creating 1,458 combinations, significantly influence the search performance of the algorithm. Population size and generation limit are GA concepts, while others like node insert/delete probability, link weight replacement and mutation probability, are crucial for neural development. Including the practical ReLU function as an activation option enhances the search, with various minimum species levels maintaining neural network diversity. Experimental results reveal objective values between $2,000 to approximately $40,000 per 35 periods, with around 10% (91 combinations) yielding acceptable results. However, half of these combinations result in values over $8,000—about four times the optimum found—with some significantly underperforming, as evidenced by the outliers shown in Fig. 9.

Additionally, the best objective, at 2,044, was yielded with a population size of 250, a sigmoid activation function, a generation limit of 1,000, and node insert/delete, weight replacement, and weight mutation probabilities of 0.01, 0.05, and 0.1 respectively, with a minimum of 2 species per generation, as detailed in Table 4.

After removing outliers, each hyperparameter's results are analysed by a violin plot to examine their statistical data distribution, with Fig. 9 showing the objective range distribution. The plot reveals a distinct separation into two major groups of lower and upper objective values, with the upper group ranging between 10,000 and 13,000 and the lower between 3,000 and 6,000. An investigation into each hyperparameter's distribution within these groups aimed to identify patterns, such as specific hyperparameter levels correlating with high or low objective functions. However, as Table 5 indicates, the distribution of hyperparameter levels within each objective function group does not display significant differences, necessitating further statistical testing to determine the significance of each hyperparameter.

Table 3. Hyperparameter levels.



Hyperparameters	Levels		
	1	2	3
Population size	50	150	250
Activation function	RELU	Sigmoid	
Generation limit	100	500	1000
Node insert and delete probability	0.01	0.05	0.1
Link weight replacement probability	0.01	0.05	0.1
Link weight mutation probability	0.01	0.05	0.1
Minimum species	2	3	4

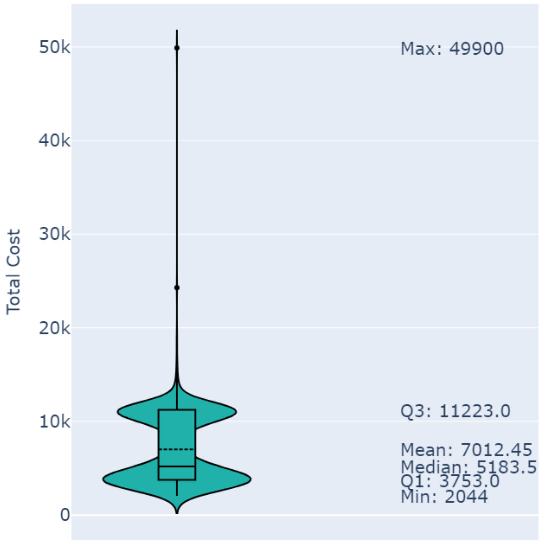
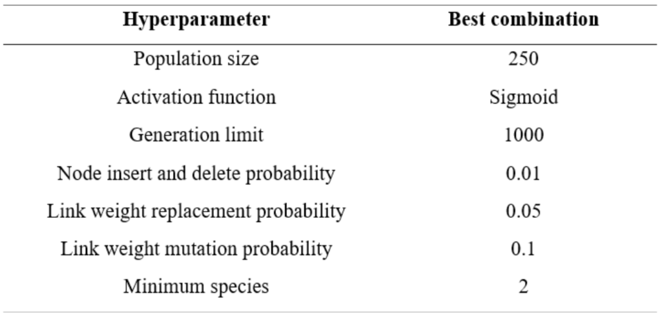


Fig. 9. Experimental result group by the objective range.

The "General full factorial design" is employed as a statistical testing tool for deeper investigation, executing tests in Minitab (version 18). All hyperparameters are considered continuous except for the categorical activation function. Table 6 displays results from the "Analyze factorial design" function, showing that no significant interactions between hyperparameters were observed. While some hyperparameters exhibit trends, as depicted in Fig. 10, the absence of significant interaction effects suggests that these trends are insufficient for drawing definitive conclusions.

Table 4. Best hyperparameter combination for this environment.



Hyperparameter	Best combination
Population size	250
Activation function	Sigmoid
Generation limit	1000
Node insert and delete probability	0.01
Link weight replacement probability	0.05
Link weight mutation probability	0.1
Minimum species	2

Table 5. Hyperparameter ratio at lower and higher objectives.

Hyperparameter	Lower (%)	Upper (%)	Hyperparameter	Lower (%)	Upper (%)
Population size:			**Link weight replacement probability:**		
50	39.15%	25.10%	0.01	36.31%	25.29%
150	23.97%	49.05%	0.05	32.20%	34.60%
250	36.88%	25.86%	0.1	31.49%	40.11%
Node insert and delete probability:			**Link weight mutation probability:**		
0.01	34.18%	31.37%	0.01	31.06%	36.12%
0.05	34.33%	32.51%	0.05	34.04%	33.46%
0.1	31.49%	36.12%	0.1	34.89%	30.42%
Minimum species:			**Generation limit:**		
2	31.21%	34.98%	100	34.18%	38.97%
3	34.47%	33.46%	500	33.90%	31.94%
4	34.33%	31.56%	1000	31.91%	29.09%
Activation function:					
Sigmoid	53.48%	44.30%			
ReLU	46.52%	55.70%			

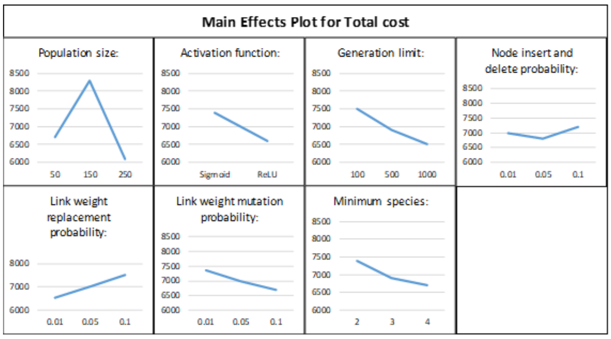


Fig. 10. Main effect mean plot.

5. Experiment Analysis

In this section, NEAT's performance and behaviours are analysed, focusing on two main topics: Algorithm Benchmarking and NEAT's response to system parameters.

5.1. Comparing between Algorithms

The DRL algorithms, PPO and Q-Learning, are used as benchmark comparisons to the proposed NEAT algorithm. The intention behind this comparison was to explore the potential of applying PPO within a more intricate inventory framework. [5] previously applied PPO as well as the Reinforcement Learning Ordering Mechanism (RLOM), which is based on Q-Learning, to a similar inventory setting. This prior study conducted 50 experiments that were identical in every aspect—ranging from characteristics and simulation parameters to mechanics and total cost calculation methodologies—to those in our research. Crucially, both our experiments and Geevers' experiments utilized datasets derived from the same data distribution, providing a direct and unambiguous basis for comparison. [5] achieved a best objective value of $2,726 with PPO, while RLOM achieved $3,259.

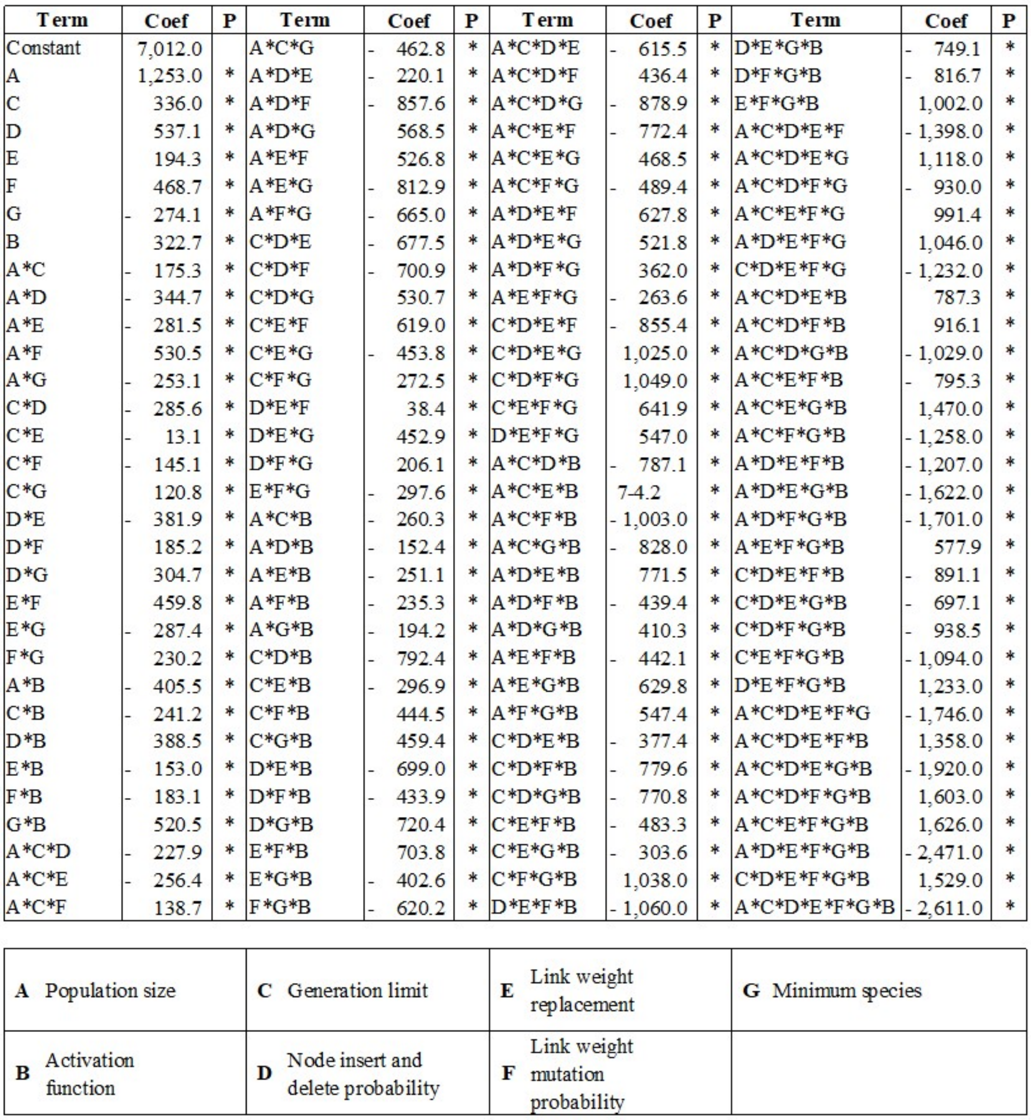
Through hyperparameter optimisation, we achieved a substantial reduction in average total inventory cost to $2,044 across 35 periods from 50 replications, as detailed in Table 7. The table compares the costs of three methods—NEAT, PPO, and RLOM—while also showing the percentage improvement of PPO and RLOM relative to NEAT. Although these outcomes are encouraging, a direct comparison with [5] is partially limited by our inability to duplicate their study exactly. Consequently, our comparative analysis is confined to the aspects of the inventory systems that remained constant across both studies.

Table 7. Comparison between NEAT and PPO algorithm.

Methodology	NEAT (This research)	PPO	RLOM
Cost ($)	2,044	2,726	3,259
Improvement (%)		25.02%	37.28%

Additionally, NEAT achieves its lowest cost of 2,044 within 1,000 generations, whereas PPO requires around 2,000 iterations to attain its best at 2,726, with no improvement observed until the 10,000th iteration, as depicted in Fig. 11. Notably, NEAT can match PPO's best performance within just 100 to 200 generations, which is tenfold fewer than the iterations PPO requires.

Table 6. Analysed factorial design results for each term.



Term	Coef	P	Term	Coef	P	Term	Coef	P	Term	Coef	P
Constant	7,012.0		A*C*G	- 462.8	*	A*C*D*E	- 615.5	*	D*E*G*B	- 749.1	*
A	1,253.0	*	A*D*E	- 220.1	*	A*C*D*F	436.4	*	D*F*G*B	- 816.7	*
C	336.0	*	A*D*F	- 857.6	*	A*C*D*G	- 878.9	*	E*F*G*B	1,002.0	*
D	537.1	*	A*D*G	568.5	*	A*C*E*F	- 772.4	*	A*C*D*E*F	- 1,398.0	*
E	194.3	*	A*E*F	526.8	*	A*C*E*G	468.5	*	A*C*D*E*G	1,118.0	*
F	468.7	*	A*E*G	- 812.9	*	A*C*F*G	- 489.4	*	A*C*D*F*G	- 930.0	*
G	- 274.1	*	A*F*G	- 665.0	*	A*D*E*F	627.8	*	A*C*E*F*G	991.4	*
B	322.7	*	C*D*E	- 677.5	*	A*D*E*G	521.8	*	A*D*E*F*G	1,046.0	*
A*C	- 175.3	*	C*D*F	- 700.9	*	A*D*F*G	362.0	*	C*D*E*F*G	- 1,232.0	*
A*D	- 344.7	*	C*D*G	530.7	*	A*E*F*G	- 263.6	*	A*C*D*E*B	787.3	*
A*E	- 281.5	*	C*E*F	619.0	*	C*D*E*F	- 855.4	*	A*C*D*F*B	916.1	*
A*F	530.5	*	C*E*G	- 453.8	*	C*D*E*G	1,025.0	*	A*C*D*G*B	- 1,029.0	*
A*G	- 253.1	*	C*F*G	272.5	*	C*D*F*G	1,049.0	*	A*C*E*F*B	- 795.3	*
C*D	- 285.6	*	D*E*F	38.4	*	C*E*F*G	641.9	*	A*C*E*G*B	1,470.0	*
C*E	- 13.1	*	D*E*G	452.9	*	D*E*F*G	547.0	*	A*C*F*G*B	- 1,258.0	*
C*F	- 145.1	*	D*F*G	206.1	*	A*C*D*B	- 787.1	*	A*D*E*F*B	- 1,207.0	*
C*G	120.8	*	E*F*G	- 297.6	*	A*C*E*B	7-4.2	*	A*D*E*G*B	- 1,622.0	*
D*E	- 381.9	*	A*C*B	- 260.3	*	A*C*F*B	- 1,003.0	*	A*D*F*G*B	- 1,701.0	*
D*F	185.2	*	A*D*B	- 152.4	*	A*C*G*B	- 828.0	*	A*E*F*G*B	577.9	*
D*G	304.7	*	A*E*B	- 251.1	*	A*D*E*B	771.5	*	C*D*E*F*B	- 891.1	*
E*F	459.8	*	A*F*B	- 235.3	*	A*D*F*B	- 439.4	*	C*D*E*G*B	- 697.1	*
E*G	- 287.4	*	A*G*B	- 194.2	*	A*D*G*B	410.3	*	C*D*F*G*B	- 938.5	*
F*G	230.2	*	C*D*B	- 792.4	*	A*E*F*B	- 442.1	*	C*E*F*G*B	- 1,094.0	*
A*B	- 405.5	*	C*E*B	- 296.9	*	A*E*G*B	629.8	*	D*E*F*G*B	1,233.0	*
C*B	- 241.2	*	C*F*B	444.5	*	A*F*G*B	547.4	*	A*C*D*E*F*G	- 1,746.0	*
D*B	388.5	*	C*G*B	459.4	*	C*D*E*B	- 377.4	*	A*C*D*E*F*B	1,358.0	*
E*B	- 153.0	*	D*E*B	- 699.0	*	C*D*F*B	- 779.6	*	A*C*D*E*G*B	- 1,920.0	*
F*B	- 183.1	*	D*F*B	- 433.9	*	C*D*G*B	- 770.8	*	A*C*D*F*G*B	1,603.0	*
G*B	520.5	*	D*G*B	720.4	*	C*E*F*B	- 483.3	*	A*C*E*F*G*B	1,626.0	*
A*C*D	- 227.9	*	E*F*B	703.8	*	C*E*G*B	- 303.6	*	A*D*E*F*G*B	- 2,471.0	*
A*C*E	- 256.4	*	E*G*B	- 402.6	*	C*F*G*B	1,038.0	*	C*D*E*F*G*B	1,529.0	*
A*C*F	138.7	*	F*G*B	- 620.2	*	D*E*F*B	- 1,060.0	*	A*C*D*E*F*G*B	- 2,611.0	*

A Population size	**C** Generation limit	**E** Link weight replacement	**G** Minimum species
B Activation function	**D** Node insert and delete probability	**F** Link weight mutation probability	

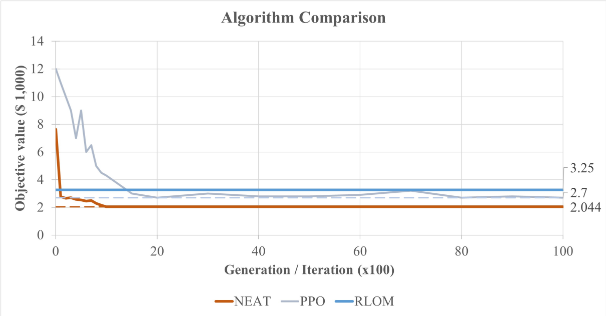


Fig. 11. Cost comparison between NEAT and PPO in each generation/iteration.

Stock duration was observed to assess NEAT's ability to navigate the trade-off between $1/unit holding cost and $2/unit penalty in the simulation. Per Fig. 12, inventory should not be held beyond two periods at any simulation point. Hence, the inventory turnover ratio, derived from dividing the average inventory on-hand by the average demand at each inventory level, is employed to gauge the average duration of inventory holding. As Table 8 indicates, inventory levels 1 to 4 have turnover ratios of 1.05, 2.64, 0.77, and 1.05 days, respectively.

Table 8. Comparison between NEAT and PPO algorithm.

Parameter	Inventory level			
	1	2	3	4
Average Inventory	8.77	23.77	7.74	11.57
Average Demand	8.34	9.00	10.00	11.00
Inventory turnover ratio	1.05	2.64	0.77	1.05

The inventory turnover ratio at level 2 is approximately 2.64, longer than the expected two periods. This scenario is common in uncertain inventory environments with variable demand and lead time. A typical response to uncertainty is maintaining safety stock,

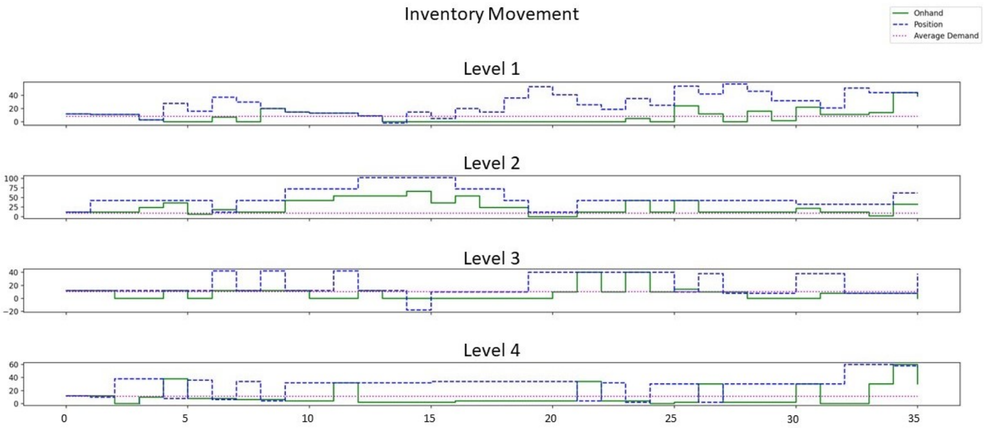


Fig. 12. Inventory movement at each level.

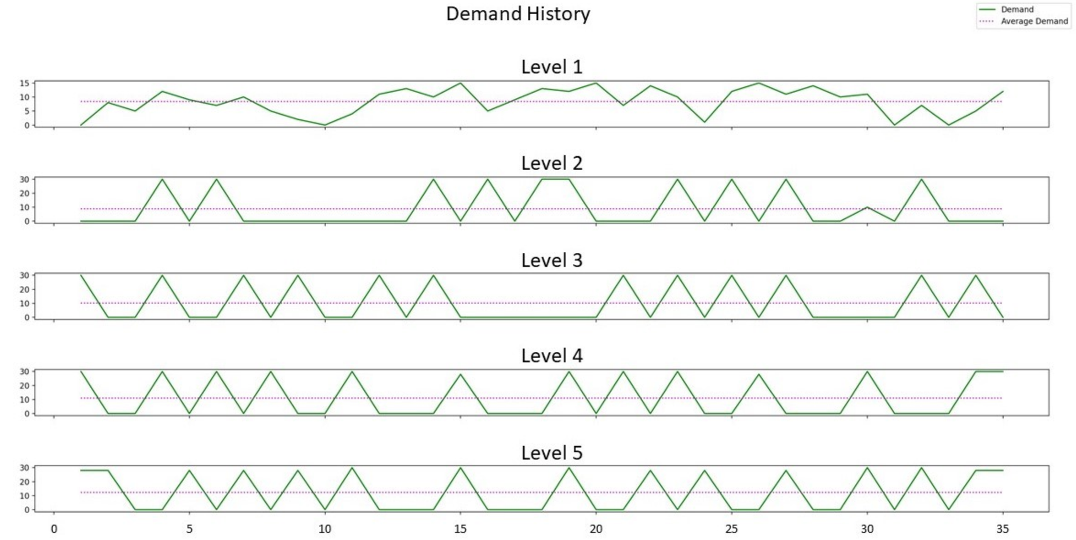


Fig. 13. Order history at each level.

leading to extended inventory turnover ratios. The turnover ratios close to 1 at other levels are surprising but can be explained by the absence of ordering costs in this environment. Without ordering costs, keeping enough inventory on-hand to cover demand is economical, avoiding multiple periods of holding costs. Hence, turnover ratios at levels 1, 3, and 4 are near 1.

Figure 12 reveals that while inventory on-hand often remains zero, inventory positions are always positive. This pattern, showcasing NEAT's ability to discern demand patterns reducing average inventory on-hand, also reflects the algorithm's recognition of linear relations between inventory levels. Further examination of the demand history in Fig. 13 shows synchronised ordering patterns across levels the agent decides. Nonetheless, orders might be insufficient or lead from the upper level due to the bullwhip effect common in Multi-Echelon Inventory System.

6. Conclusion and Future Sesearch

This investigation into the tuning of the NEAT algorithm uncovers its pronounced sensitivity to hyperparameters, with implications for costs ranging between \$2,000 and \$40,000. Compared to model-free RL methods such as PPO and Deep Q-Learning-based RLOM, NEAT demonstrates superior performance in managing complex inventory systems, achieving a significant cost reduction of 25.07% over PPO and 37.28% over RLOM. Its efficacy within the LMEIS suggests considerable advantages for sectors characterized by extensive supply chains. The 5-level structure used in this study can be generalized to an n-level structure, allowing for infinite scalability to increasingly complex systems. NEAT's scalability has also been validated in fields such as robotics and engineering optimization, where it has successfully addressed large, dynamic problems. Additionally, its modular and adaptive nature allows it to efficiently manage increasingly complex architectures [63]. Future research could explore increasing system complexity through divergent multi-echelon or networked configurations, incorporating stochasticity or discounted costs, testing in real-world supply chain environments, or combining NEAT with other RL methods to leverage their complementary strengths. Reward shaping, alongside Bayesian Optimization, could also help refine the reward function to better balance competing objectives and reduce sensitivity, enabling deeper trend analysis.

Acknowledgement

This research was supported by the 100th Anniversary Chulalongkorn University Fund for Doctoral Scholarship and the 90th Anniversary Chulalongkorn University Fund (Ratchadaphiseksomphot Endowment Fund) and supported by Integrated Management System and Smart Technology Research Group of Chulalongkorn University.

References

[1] C.-F. Chien and Y.-B. Lan, "Agent-based approach integrating deep reinforcement learning and hybrid genetic algorithm for dynamic scheduling for industry 3.5 smart production," *Computers & Industrial Engineering*, vol. 162, p. 107782, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835221006860

[2] B. M. Kayhan and G. Yildiz, "Reinforcement learning applications to machine scheduling problems: a comprehensive literature review," *Journal of Intelligent Manufacturing*, 2021. [Online]. Available: https://doi.org/10.1007/s10845-021-01847-3

[3] H. Yoo, H. E. Byun, D. Han, and J. H. Lee, "Reinforcement learning for batch process control: Review and perspectives," *Annual Reviews in Control*, vol. 52, pp. 108–119, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S136757882100081X

[4] S. K. Chaharsooghi, J. Heydari, and S. H. Zegordi, "A reinforcement learning model for supply chain ordering management: An application to the beer game," *Decision Support Systems*, vol. 45, no. 4, pp. 949–959, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923608000560

[5] K. Geevers, "Deep reinforcement learning in inventory management," PhD Thesis, University of Twente, 2020.

[6] C. Jiang and Z. Sheng, "Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system," *Expert Systems with Applications*, vol. 36, no. 3, Part 2, pp. 6520–6526, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417408005034

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

[8] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002. [Online]. Available: https://doi.org/10.1162/106365602320169811

[9] S. Padakandla, "A survey of reinforcement learning algorithms for dynamically varying environments," *ACM Comput. Surv.*, vol. 54, no. 6, p. Article 127, 2021. [Online]. Available: https://doi.org/10.1145/3459991

[10] A. Hallak, D. Di Castro, and S. Mannor, "Contextual markov decision processes," *arXiv preprint arXiv:1502.02259*, 2015.

[11] E. Hadoux, A. Beynier, and P. Weng, "Sequential Decision-Making under Non-stationary Environments via Sequential Change-point Detection," in *Learning over Multiple Contexts (LMCE)*, Nancy, France, Sep. 2014. [Online]. Available: https://hal.science/hal-01200817

[12] A. Sehgal, H. La, S. Louis, and H. Nguyen, "Deep reinforcement learning using genetic algorithm for parameter optimization," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 596–601.

[13] M. Feurer and F. Hutter, *Hyperparameter Optimization*. Cham: Springer International Publishing, 2019, pp. 3–33. [Online]. Available: https://doi.org/10.1007/978-3-030-05318-5_1

[14] A. Wilson, A. Fern, and P. Tadepalli, "Using trajectory data to improve bayesian optimization for reinforcement learning," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 253–282 , numpages = 30, 2014.

[15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," pp. 1889–1897, 2015. [Online]. Available: https://proceedings.mlr.press/v37/schulman15.html

[16] I. Mordatch, N. Mishra, C. Eppner, and P. Abbeel, "Combining model-based policy search with online model learning for control of physical humanoids," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, Conference Proceedings, pp. 242–248.

[17] M. Maleki, V. Hakami, and M. Dehghan, "A model-based reinforcement learning algorithm for routing in energy harvesting mobile ad-hoc networks," *Wireless Personal Communications*, vol. 95,

no. 3, pp. 3119–3139, 2017. [Online]. Available: https://doi.org/10.1007/s11277-017-3987-8

[18] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017. [Online]. Available: https://doi.org/10.1007/s10846-017-0468-y

[19] G. S. Martins, H. Al Tair, L. Santos, and J. Dias, "αpomdp: Pomdp-based user-adaptive decision-making for social robots," *Pattern Recognition Letters*, vol. 118, pp. 94–103, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865518300825

[20] M. Frank, J. Leitner, M. Stollenga, A. Förster, and J. Schmidhuber, "Curiosity driven reinforcement learning for motion planning on humanoids," *Frontiers in neurorobotics*, vol. 7, p. 25, 2014.

[21] R. Allamaraju, H. Kingravi, A. Axelrod, G. Chowdhary, R. Grande, J. P. How, C. Crick, and W. Sheng, "Human aware uas path planning in urban environments using nonstationary mdps," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, Conference Proceedings, pp. 1161–1167.

[22] S. Sitthiracha, C. Nantabut, S. Koetniyom, and G. Phanomchoeng, "Human-like trajectory planning for autonomous vehicles using flexible virtual reference points in car overtaking motorcycle scenarios," *Engineering Journal*, vol. 28, no. 10, pp. 77–91, 2024.

[23] M. Kiran and M. Ozyildirim, "Hyperparameter tuning for deep reinforcement learning applications," 2022, accession Number: edsarx.2201.11182; Publication Type: Working Paper; Publication Date: 20220126.

[24] J. Grefenstette, D. Moriarty, and A. Schultz, "Evolutionary algorithms for reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 11, 2011.

[25] M. M. Alipour, S. N. Razavi, M. R. Feizi Derakhshi, and M. Balafar, "A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem," *Neural Computing and Applications*, vol. 30, 2018.

[26] Y. D. Ko, "An efficient integration of the genetic algorithm and the reinforcement learning for optimal deployment of the wireless charging electric tram system," *Comput. Ind. Eng.*, vol. 128, pp. 851–860, 2019.

[27] R. M. Aziz, R. Mahto, K. Goel, A. Das, P. Kumar, and A. Saxena, "Modified genetic algorithm with deep learning for fraud transactions of ethereum smart contract," *Applied Sciences*, 2023.

[28] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002. [Online]. Available: http://nn.cs.utexas.edu/?stanley:ec02

[29] S. Belciug, "Learning deep neural networks' architectures using differential evolution. case study: Medical imaging processing," *Computers in Biology and Medicine*, vol. 146, p. 105623, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482522004152

[30] B. Liu, X. Nie, Z. Li, S. Yang, and Y.-z. Tian, "Evolving deep convolutional neural networks by ip-based marine predator algorithm for covid-19 diagnosis using chest ct scans," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1 – 14, 2022.

[31] T. Andersen, "Neuroevolution through augmenting topologies applied to evolving neural networks to play othello," Department of Computer Sciences, The University of Texas at Austin, Undergraduate Honors Thesis HR-02-01, 2002. [Online]. Available: http://nn.cs.utexas.edu/?andersen:ugthesis02

[32] S. Lang, T. Reggelin, J. Schmidt, M. Müller, and A. Nahhas, "Neuroevolution of augmenting topologies for solving a two-stage hybrid flow shop scheduling problem: A comparison of different solution strategies," *Expert Systems with Applications*, vol. 172, p. 114666, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742100107X

[33] N. Khanlarzade, B. Yousefi Yegane, and I. Nakhai Kamalabadi, "Genetic algorithm to optimize two-echelon inventory control system for perishable goods in terms of active packaging," *international journal of industrial engineering computations*, vol. 3, 2012.

[34] J. Svoboda, S. Minner, and M. Yao, "Typology and literature review on multiple supplier inventory control models," *European Journal of Operational Research*, vol. 293, no. 1, pp. 1–23, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221720309693

[35] A. Gharaei, M. Karimi, and S. Shekarabi, "Joint economic lot-sizing in multi-product multi-level integrated supply chains: Generalized benders decomposition," *International Journal of Systems Science*, pp. 1–17, 2019.

[36] R. Kapur and C. Moberg, "Evaluating inventory turns for a hospital environment," *Computers & Industrial Engineering*, vol. 13, no. 1, pp. 73–77, 1987.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/0360835287900544

[37] S. Benjaafar, M. Elhafsi, C.-Y. Lee, and W. Zhou, "Optimal control of assembly systems with multiple stages and multiple demand classes 1," *SSRN Electronic Journal*, 2010.

[38] G. Gallego, "New bounds and heuristics for (q, r) policies," *Management Science*, vol. 44, no. 2, pp. 219–233, 1998, doi: 10.1287/mnsc.44.2.219. [Online]. Available: https://doi.org/10.1287/mnsc.44.2.219

[39] K. Maity and M. Maiti, "Numerical approach of multi-objective optimal control problem in imprecise environment," *Fuzzy Optimization and Decision Making*, vol. 4, no. 4, pp. 313–330, 2005. [Online]. Available: https://doi.org/10.1007/s10700-005-3666-1

[40] K. Katsaliaki and S. C. Brailsford, "Using simulation to improve the blood supply chain," *Journal of the Operational Research Society*, vol. 58, no. 2, pp. 219–227, 2007. [Online]. Available: https://doi.org/10.1057/palgrave.jors.2602195

[41] P. Huynh and P. Yenradee, "Vendor managed inventory for multi-vendor single-manufacturer supply chain: A case study of instant noodle industry," *Engineering Journal*, vol. 24, no. 6, pp. 91–107, 2020.

[42] T. Kusomrosananan and N. Phumchusri, "Inventory policy improvement with periodic review for perishable goods: A case study of a retail coffee shop in thailand," *Engineering Journal*, vol. 28, no. 6, pp. 59–73, 2024.

[43] F. Sarwar, M. Ahmed, and M. Rahman, "Application of nature inspired algorithms for multi-objective inventory control scenarios," *International Journal of Industrial Engineering Computations*, vol. 12, pp. 91–114, 2021.

[44] A. K. Bhunia, S. Kundu, T. Sannigrahi, and S. K. Goyal, "An application of tournament genetic algorithm in a marketing oriented economic production lot-size model for deteriorating items," *International Journal of Production Economics*, vol. 119, no. 1, pp. 112–121, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925527309000395

[45] M. K. Maiti and M. Maiti, "Two-storage inventory model with lot-size dependent fuzzy lead-time under possibility constraints via genetic algorithm," *European Journal of Operational Research*, vol. 179, no. 2, pp. 352–371, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221706002165

[46] S. Mondal, J. K. Dey, and M. Maiti, "A single period inventory model of a deteriorating item sold from two shops with shortage via genetic algorithm," *Yugoslav Journal of Operations Research*, vol. 17, 2007.

[47] Y. Yandra, I. Jamaran, M. Marimin, E. Eriyatno, and H. Tamura, "An integration of genetic algorithm and fuzzy logic for optimization of agroindustrial supply chain design," *Proceedings of the 51st Annual Meeting of the ISSS - 2007, Tokyo, Japan*, vol. 51, no. 2, 2007. [Online]. Available: https://journals.isss.org/index.php/proceedings51st/article/view/474

[48] D. Chakraborty, D. Jana, and T. Roy, "Multi-item integrated supply chain model for deteriorating items with stock dependent demand under fuzzy random and bifuzzy environments," *Computers and Industrial Engineering*, vol. 88, pp. 166–180, 2015.

[49] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, Conference Proceedings, pp. 1942–1948 vol.4.

[50] K. Park and G. Kyung, "Optimization of total inventory cost and order fill rate in a supply chain using pso," *The International Journal of Advanced Manufacturing Technology*, vol. 70, no. 9, pp. 1533–1541, 2014. [Online]. Available: https://doi.org/10.1007/s00170-013-5399-6

[51] A. A. Taleizadeh, S. T. A. Niaki, N. Shafii, R. G. Meibodi, and A. Jabbarzadeh, "A particle swarm optimization approach for constraint joint single buyer-single vendor inventory problem with changeable lead time and (r,q) policy in supply chain," *The International Journal of Advanced Manufacturing Technology*, vol. 51, no. 9, pp. 1209–1223, 2010. [Online]. Available: https://doi.org/10.1007/s00170-010-2689-0

[52] I. Huseyinov and A. Bayrakdar, "Performance evaluation of nsga-iii and spea2 in solving a multi-objective single-period multi-item inventory problem," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, 2019, Conference Proceedings, pp. 531–535.

[53] H. Kartal, A. Oztekin, A. Gunasekaran, and F. Cebi, "An integrated decision analytic framework of machine learning with multi-criteria decision making for multi-attribute inventory classification," *Computers & Industrial Engineering*, vol. 101, pp. 599–613, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835216301991

[54] R. N. Boute, J. Gijsbrechts, W. van Jaarsveld, and N. Vanvuchelen, "Deep reinforcement learning for inventory control: A roadmap," *European Journal of Operational Research*, vol. 298,

no. 2, pp. 401–412, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221721006111

[55] N. Sutthibutr, N. Chiadamrong, K. Hiraishi, and S. Thajchayapong, "A unified optimization model with proportional fairness and robustness of fuzzy multi-objective aggregate production planning in supply chain under uncertain environments," *Engineering Journal*, vol. 28, no. 5, pp. 25–52, 2024.

[56] A. Oroojlooy Jadid, M. Nazari, L. Snyder, and M. Takáč, "A deep q-network for the beer game: A reinforcement learning algorithm to solve inventory optimization problems," *Neural Information Processing Systems (NIPS), Deep Reinforcement Learning Symposium 2017*, 2017.

[57] J. Gijsbrechts, R. Boute, D. Zhang, and J. Van Mieghem, "Can deep reinforcement learning improve inventory management? performance on dual sourcing, lost sales and multi-echelon problems," *SSRN Electronic Journal*, 2019.

[58] I. Kaynov, M. van Knippenberg, V. Menkovski, A. van Breemen, and W. van Jaarsveld, "Deep reinforcement learning for one-warehouse multi-retailer inventory management," *International Journal of Production Economics*, vol. 267, p. 109088, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925527323003201

[59] V. Asha, A. Prasad, C. R. Vishwanath, K. M. Raj, A. R. M. Kumar, and N. Leelavathi, "Designing a popular game framework using neat a genetic algorithms," in *2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1)*, 2023, Conference Proceedings, pp. 1–5.

[60] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, and V. B. S. Prasath, "Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach," 2019.

[61] S. Khamesian and H. Malek, "Hybrid self-attention neat: a novel evolutionary self-attention approach to improve the neat algorithm in high dimensional inputs," *Evolving Systems*, 2023. [Online]. Available: https://doi.org/10.1007/s12530-023-09510-3

[62] C. B. Pătraşcu and D. T. Iancu, "Neat algorithm for simple games," in *2023 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2023, Conference Proceedings, pp. 1–6.

[63] J. Clune, J.-B. Mouret, and H. Lipson, "The evolutionary origins of modularity," *Proceedings of the Royal Society B: Biological Sciences*, vol. 280, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:10570975

Yanvaroj Pongsethpaisal is a Ph.D. scholar in the Department of Industrial Engineering, Chulalongkorn University, Bangkok, Thailand. He obtained his B.Eng. and M.Eng. from the same Department of Industrial Engineering, Chulalongkorn University, Bangkok, Thailand in 2016 and 2017, respectively. His research interests include inventory management, production planning, Facility designing, and Machine learning.

Naragain Phumchusri hold her Ph.D. in Industrial Engineering from Georgia Institute of Technology, Atlanta, GA, USA since 2010. She is an Associate Professor at the Department of Industrial Engineering, Chulalongkorn University, Bangkok, Thailand. Her current research fields are stochastic models for revenue management, machine learning for demand forecasting, inventory optimisation, and pricing strategy for the retail industry.

Paveena Chaovalitwongse is an Associate Professor at the Department of Industrial Engineering, Chulalongkorn University, Bangkok, Thailand. She obtained a Ph.D. in Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA in 2000. Her research interests include inventory management and control, scheduling problem, supply chain management, and operation management.