Proceedings of the 3rd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/2025.30681

Optimizing Supply Chain Management with Approximate SARSA and Reinforcement Learning

Zhaoxin Li[1*], Sabrina Poon[2]

[1]*Communication University of China (CUC), Beijing, China*
[2]*Dulwich College Beijing (DCB), Beijing, China*
**Corresponding Author. Email: 2546607634@qq.com*

Abstract. Supply chain optimization and inventory management have become increasingly critical issues in the era of globalization and the post-pandemic world. These topics have garnered significant attention across various sectors. However, due to the high-dimensional state and action spaces, the diverse and dynamic environmental variables, the need for continuous decision-making over time, and the data-dependence and lack of flexibility of traditional methods, we model the problem as a Markov Decision Process (MDP). To address it, we employ Approximate SARSA, a method using Q-value approximation, alongside RL methods based on different feature processing techniques. Our experiments evaluate and extend the applicability of the model, while also attempting to provide interpretability.

Keywords: supply chain, RL, SARSA, Approximate SARSA.

1. Introduction

In the context of globalization, many companies face the challenge of having raw material production, manufacturing, and sales markets located in different regions. The entire process, from raw material procurement to final product delivery to consumers, involves long-distance and multi-step logistics arrangements. To address this challenge, companies need to establish efficient logistics systems. Effective logistics not only reduce costs and enhance a company's competitiveness but also strengthen the stability of the supply chain, ensuring that businesses and the global economy continue to operate smoothly in the face of major events such as global transportation disruptions, natural disasters, or black swan events like the COVID-19 pandemic and the Russia-Ukraine war. Furthermore, manual inventory management often suffers from inefficiencies and high costs, with the rise of emerging technologies such as Artificial Intelligence (AI), cloud computing, blockchain, and the Internet of Things (IoT), new solutions for optimizing supply chain management have been proposed. Including ML and RL approach.

The optimization of supply chain management primarily includes supplier selection, warehouse location, inventory management, and decision implementation. The main objectives focus on two key aspects: enhancing supply agility and improving supply chain stability. This often involves evaluating and reducing supply chain costs. Reducing network complexity is a common optimization approach, as it effectively simplifies manual management, thereby increasing the efficiency of subsequent monitoring and distribution.

DOI: 10.54254/2755-2721/2025.30681

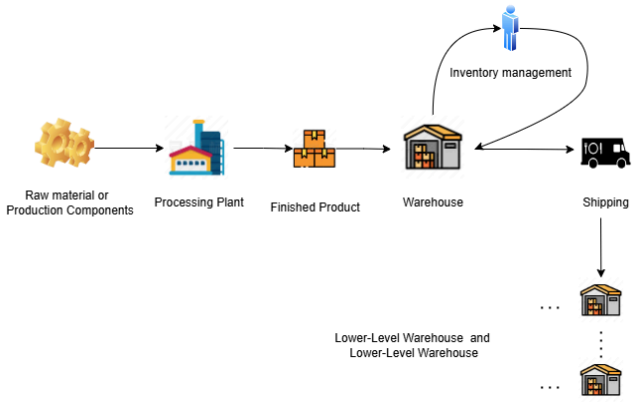

Figure 1. Schematic diagram of supply chain process

Inventory management optimization is a core issue in supply chain management. Financially, inventory represents a significant asset for a company, constituting 50% of its total assets [1]. From a production and sales perspective, inventory management is a critical factor in ensuring the smooth operation of a company. Inventory can be specifically categorized into safety stock, emergency stock, cycle stock, and anticipatory stock, each serving different supply and demand scenarios.

To elaborate, if the supply chain is slow to respond and inventory is insufficient, the company may lose potential customers or miss business opportunities. However, holding excessive inventory carries its own costs, including storage and insurance fees, along with the risks of spoilage, theft, and damage. These issues can strain the company's cash flow and may even lead to breaks in the supply and financial chains. Thus, the main goal of inventory management is to meet the dynamic demands of the market while minimizing inventory levels, thereby attracting and maintaining customer satisfaction [2].

This challenge can be addressed through data analysis, enabling manufacturers to accurately predict market needs and adjust their strategies accordingly. However, seasonal fluctuations, such as increased demand for certain products during holidays, present additional challenges for analysis. These require careful planning and early stockpiling to manage effectively.

2. Problem describe

2.1. Problem definition

The purpose of this study is to develop a comprehensive approach to optimize integrated inventory management within supply chain networks and to conduct experiments across different supply chain application scenarios to evaluate and promote the performance of the model. Each supply chain is a complex network involving multiple stages, including producers, suppliers, manufacturers, transporters, and retailers. To retain some of the complexity of real-world systems while providing the model with a certain degree of generality and flexibility, we have employed a simulated environment with multiple features and analyzed and trained the supply chain within this environment as a whole.

In inventory management problems, the primary decisions revolve around when to purchase and how much to purchase [3], based on changes in demand. Our model utilizes two types of nodes: factory nodes and store nodes, each assigned specific characteristics. At the store nodes, there are dynamic demand changes that include both seasonal and random fluctuations. To ensure the smooth operation of the entire supply chain—from planning and procuring raw materials to manufacturing, delivery, and returns—the Agent needs to learn the relationship between demand fluctuations over

DOI: 10.54254/2755-2721/2025.30681

continuous time and inventory decisions based on the characteristics of each node. This enables the Agent to determine the appropriate inventory levels for each node over time.Considering physical constraints, each storage facility has a maximum capacity that limits its operations. Over time, storage costs are incurred to reflect constraints on product availability dates and the calculation of space costs. Production facilities are similarly constrained by maximum production quantities and production costs.

Additionally, for the transportation process, we introduce transportation costs that include monetary costs (such as fuel costs, packaging costs, and refrigeration costs) and time costs. The load capacity of individual transport vehicles is also considered to describe and adjust transportation strategies. As the supply source for all store nodes, the factory must comprehensively consider the cumulative actions of all warehouses to determine the optimal quantities for production, storage, and distribution [4]. To enhance network flexibility while taking into account the consumer characteristics of the internet era, demand is allowed to exceed storage and production capacity. However, the model imposes backorder costs when inventory is insufficient, thereby encouraging the Agent to accurately predict future demand trends and adjust its strategies accordingly.

2.2. MDP progress

Based on the problem definition, the problem can be modeled as a time-continuous Markov Decision Process (MDP). The environment includes j factories and K stores. where $j \in \{0, \ldots, K\}$. Additionally, the state space S , action space A , a set of possible actions, reward function R , and discount factor γ are introduced to describe and represent the model. The entire model can be represented as $\langle S, A, P, R, \gamma \rangle$.

With the change of the time noted as $t = 1 \ldots N$, the state at each period is defined as $s_t = [s_0, \ldots, s_K, d_{t-1}, d_{t-2}]$ .For each $s_j \in [0, c_j]$, c_j is the upper bound of the storage in factory (s_0) and each store($s_1 \ldots s_K$). Based on the Markov property, the probability of transitioning to a future state s_{t+1} depends solely on the current state s_t and the chosen action a . Therefore, for the requirement vector in the observation space of the agent, we restrict the historical knowledge that the agent can use as (d_{t-1}, d_{t-2}) to reduce space usage while enabling the agent to learn the basic trend of supply and demand changes over time under the Markov condition.

We simulate the demand using a function that includes random and seasonal variables, which is mapped as the vector $d_t = [d_1, \ldots, d_K, t]$.

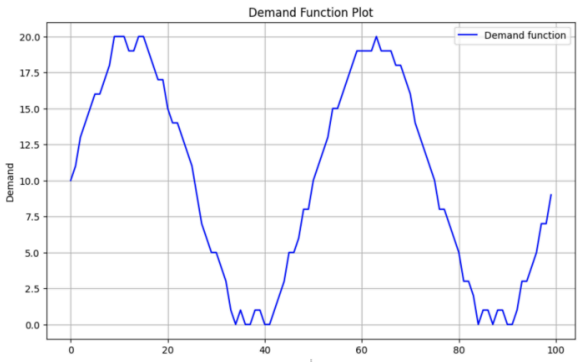

Figure 2. Example curve of demand variation

For transitioning from the action space $a_t = [a_0, \ldots, a_K]$ to the possible action space, $a_0 \in \{0, \ldots, \rho_{max}\}$, where ρ_{max} is the maximum production capacity. Although a_i are allowed to

DOI: 10.54254/2755-2721/2025.30681

exceed s_j and c_j , there still exists a natural constraint: the total sum of the demand actions for each store must be less than or equal to the current storage in the factory, i.e., $\sum_1^K a_j$, while a_0 plus the current inventory s_0 must be less than c_0 . By adhering to this constraint, we can derive a set of feasible actions for each state, which also introduces complexity due to the interdependence of actions over time.

Based on the definition above, the definition of the transition function $T(s_t, d_t, a)$ use to describe the storage changing over time can be written as:

$$T\left(s_t, d_t, a\right) := \left(min\{s_0 + a_0 - \sum_{i=1}^{K} a_j, c_0\right\}, min\{s_1 + a_1 - d_1, c_1\right\}, \ldots, min\{s_K + a_K - d_K, c_K\right\}, d_t, d_{t-1}\Big) \quad (1)$$

A one-step reward function evaluates the agent's actions at each step , while being continuously summed over time to give the final reward for a single episode. This representation is consistent with the real-world depiction of a company's cash flows, which can be represent as [2]:

$$r\left(s_t, d, a\right) := p\sum_{j=1}^{K} d_j - \kappa_{pr} a_0 - \sum_{j=0}^{K} \kappa_{st,j} max\{s_j, 0\right\} + \kappa_{pc} \sum_{j=1}^{K \sum_j \sum_{j=1}^{K} \kappa_{tr,j} \left\lceil \frac{a_j}{\zeta_j} \right\rceil} min \quad (2)$$

Table 1. One-step reward function

Component	Formula	Variable
Total profit	$p\sum_{j=1}^{K} d_j$	p : Price per unit d_j : Demand for product
Production cost	$\kappa_{pr} \cdot a_0$	κ_{pr} : Unit production cost a_0 : Total production volume of the factory
Storage cost	$\sum_{j=0}^{K} \kappa_{st,j} \cdot \max\{s_j, 0\}$	$\kappa_{st,j}$: Storage cost coefficient for stock level s_j (Higher for perishable and valuable products) s_j : Stock level for product j , with upper bound c_j .
Transportation cost	$\sum_{j=1}^{K} k_{tr,j} \cdot \frac{a_j}{\zeta_j}$	$\kappa_{tr,j}$: Transportation cost per truck for product j . ζ_j : Truck capacity for product j . a_j : Transportation volume for product j .
Penalty cost	$\kappa_{pc} \sum_{j=1}^{K} \min\{s_j, 0\}$	$\kappa_{tr,j}$: Penalty cost coefficient s_j : Stock level for product j , with upper bound c_j .

While during the process, the agent choose the action based on dynamic ε – greedy stretagy ,where ε would be gradually decrease linearly, thereby allowing the strategy to reduce exploration and converge to a stable value over time.

Where $\lceil \bullet \rceil$ to be the ceiling function. It is the total calculation of the overall profit, production and storage costs, and stock-out penalties. The terminal reward after the last period is set to zero, the storage won't be count.

Furthermore, the discounting factor (denoted as $\gamma \in \mathbb{R}^+$) is crucial for balancing short-term and long-term rewards. It determines the present value of future rewards, with a value between 0 and 1. In economic contexts, the discounting factor can also be interpreted in relation to inflation. Inflation erodes the purchasing power of future profits, effectively reducing their real value over time. By

DOI: 10.54254/2755-2721/2025.30681

using a discounting factor, the model accounts for this reduction in value, allowing for a more accurate assessment of long-term profitability.

3. Challenge

Dynamic and Stochastic Environment: Warehouse demand is random, seasonal, and affected by statistical delays over time, meaning that the demand statistics for the current period can only be determined at the beginning of the next period. Therefore, agents need to adapt to these fluctuations and predict future demand changes based on observed data to avoid penalty costs incurred from unmet demand. Additionally, given the broad range of supply chain applications and the diverse characteristics of goods (e.g., seasonal products, new products, perishable goods [5]), we hope the method is universal and flexible enough to handle more diverse features while offering better resistance to disruptions.

Complex Decision-Making: The supply chain problem is modeled as a Markov Decision Process (MDP), which requires making continuous decisions over time, where current actions can have ongoing effects on future decisions. The agent's goal in an MDP is typically to maximize long-term cumulative rewards. This involves balancing immediate returns with long-term impacts, which may include trade-offs between short-term costs (such as excess inventory costs) and long-term benefits (such as effectively meeting customer demand), as well as balancing exploration and exploitation.

Curse of Dimensionality: As noted by Powell (2007) [6], due to the multi-step nature of decision-making, the state and action spaces in supply chain networks grow exponentially, leading to high storage costs and making it challenging to find optimal solutions (e.g., state space size growing to 10^{15} and action space size expanding to over 2.4 billion possibilities).

4. Literature review

We reviewed a substantial number of articles related to this research and found that since 2020, influenced by global fluctuations and against the backdrop of globalization and global warming, more studies have focused on supply chain resilience under disruption. Additionally, the demand for optimization of inventory management across various fields such as manufacturing, medicine, humanitarian aid, environmental science, engineering, agriculture, and even energy has increased, leading to complex and diverse characteristics [7]. Research on competitive supply chains [8] and complex networks [9] is also advancing. Most of the methods currently used are still based on original data ML (machine learning) methods.

Additionally, papers such as [10] have employed topological methods, dividing the supply chain into nodes and edges to gradually derive critical nodes and layer divisions, thereby enhancing overall stability and efficiency.

For our problem and challenge, the considerations include:

LSTM Network (Long Short-Term Memory Network): Primarily used to capture and predict long-term dependencies and seasonal patterns in time series data. This method processes information over extended time intervals to address issues with delayed error decay through recurrent backpropagation learning. However, this algorithm heavily relies on the quality of historical data [11].

ADP (Approximate Dynamic Programming): ADP uses tabular methods to represent approximate value functions (such as linear functions, neural networks, etc.) to estimate value functions or policies. It is a classical method for addressing the curse of dimensionality and reduces computational complexity to some extent.

DOI: 10.54254/2755-2721/2025.30681

5. Methodology

5.1. The (ς, Q)-Policy

The (ς, Q)-Policy is a heuristic static strategy described by Tempelmeier (2011) and adopted in this experiment as a baseline for performance evaluation of the other agents. It involves sequentially comparing the stock level si of each warehouse with the threshold ς_j . If the current stock is below ς_j , the warehouse is replenished with Q_j units. For the factory, Q_0 units are produced in the next cycle. The threshold $\varsigma = [\varsigma_0, \ldots, \varsigma_K]$ and the replenishment quantities $Q = [Q_0, \ldots, Q_K]$ are pre-set hyperparameters.

By manually selecting appropriate values for ς and Q , the storage level can be maintained fluctuating around an average value.

5.2. Approximate SARSA

Published by Rummery and Niranjan [10], Approximate SARSA is an online, temporal-difference reinforcement learning algorithm that builds on the standard SARSA approach. The goal of the function is to estimate the Q-value function ($Q(s,a)$),which is an action-value function representing the expected cumulative reward of taking a specific action in a particular state. However, instead of explicitly storing Q-values in a table, which would cause exponentially growing state and action spaces, Approximate SARSA adopts a linear approximation function $Q_{w(s,a)} = w^T\phi(s,a)$ to estimate the parameterized function $Q(s,a)$. By which, the agent saves memory and achieves training objectives by storing and adjusting only the vector w (which consists of weights corresponding to different features $\phi(s,a)$ is the feature-mapping vector derived from the state s and action a, which is also a key aspect of Approximate SARSA designing. This feature vector allows the use of domain-specific knowledge to establish the transition relationships between states and actions.In our implementation, we utilize over 30 distinct features to enhance the accuracy of the Q-function approximation.

As a temporal-difference algorithm, the state function in Approximate SARSA includes the current state and a predicted state for the next stage. For a given demand d_t at time t in Approximate SARSA, the formula uses the demands from the previous two-time steps, d_{t-1} and d_{t-2} , to predict the future demand $\widehat{d}_{st}$ according to the following formula:

$$\hat{d}_{(st)} = d_{t-1} + \left(d_{t-1} - d_{t-2}\right) = 2d_{t-1} - d_{t-2} \tag{3}$$

Note that this function does not imply that the agent fully understands the state transition dynamics within the function; it simply assumes that the demand changes can be predicted using a simple linear relationship.

The transition function T is used to predict the next state $\hat{S}_{t+1}$ based on the current state s_t , the estimated demand $\widehat{d}_{st}$ and the action a .

$$\hat{S}_{t+1}\left(s,a\right) = T\left(s_t, d\left(s_t\right), a\right) \tag{4}$$

DOI: 10.54254/2755-2721/2025.30681

By doing so, the underlying MDP structure of the environment is involved in the learning process, helping to predict future outcomes that reflect long-term reward information, thereby improving the quality of Q-value estimates and the efficiency of learning.

Subsequently, the agent calculates the TD error based on equation (3) and updates the weights w, where Υ is the learning rate:

$$\delta = r + \gamma \widehat{Q}\left(s_1, a_1; w\right) - \widehat{Q}\left(s_0, a_0; w\right) \tag{5}$$

$$w \leftarrow w + \alpha \delta \varphi(s_0, a_0) \tag{6}$$

5.3. Reinforcement algorithm

Defined by Williams in 1992, the REINFORCE algorithm is a type of policy gradient method that discretely updates the parameterized policy to maximize the expected reward $J(\pi_\Theta)$:

$$J\left(\pi_\theta\right) = E_{\pi_\theta}\left[\sum_{t=1}^{N} r_t\right] \tag{7}$$

To maintain flexibility in action selection while effectively managing the high dimensionality of the problem, we discretize the action space and set the number of actions for each warehouse to 5. If the action space is too small, experimental results may converge or overlap, making it difficult to observe significant differences in the outcomes of different approaches. Therefore, maintaining a certain level of action complexity is necessary. However, the total action space becomes $n_a = 5(K+1)$, where $a^{(i)}$ represents the i^{th} action, and K is the number of warehouses. We evenly divide the maximum production capacity into 5 levels for each factory and set a fixed transportation capacity for each truck, allowing delivery quantities to be integer multiples of the truck's capacity in one period.

We used three methods for feature processing $\phi(s)$. Each of these methods manipulates the features differently to capture various aspects of the data's structure, with the choice of method depending on the nature of the problem being addressed. The first method involves using the original function or adding a constant bias to transform it into a linear function. The second method applies a square operation to the feature values, effectively introducing non-linearity by magnifying larger values more than smaller ones. The third method uses Gaussian Radial Basis Functions (RBFs) with the width of the Gaussian controlled by σ (set to 50 in our experiments):

$$\varphi\left(\| x\text{-}c \|\right) = exp\left(-\frac{\| x\text{-}c \|^2}{2\sigma^2}\right) \tag{8}$$

By converting the action space weights into a normalized probability distribution, actions are selected based on this probability distribution using the softmax function, ensuring exploratory updates. The possible action space for each state can be constrained by certain physical limitations, where $f = 1$ if the action is allowed and $f = 0$ otherwise. The policy function can then be represented as a probability function:

DOI: 10.54254/2755-2721/2025.30681

$$p\left(a=a^{(i)}\middle|s\right)=\pi_\theta\left(a=a^{(i)}\middle|s\right)=\frac{e^{\varphi\left(s\right)^T\omega_{a^{(i)}}}\cdot f\left(a^{(i)}\middle|s\right)}{\sum_{j=1}^{n_a}e^{\varphi\left(s\right)^T\omega_{a^{(j)}}}\cdot f\left(a^{(j)}\middle|s\right)}:=\sigma_i\left(s\right) \tag{9}$$

According to the policy gradient theorem, the gradient of the objective function with respect to the weights is:

$$\nabla_{w_j}\mathscr{J}\left(\pi_\Theta\right)=\mathbb{E}_{\pi_\theta}\left[\phi\left(s\right)\cdot D_t\cdot\begin{Bmatrix}(1-\sigma_j(s)) & \text{if select } a_j\\ -\sigma_j(s) & \text{else}\end{Bmatrix}\right] \tag{10}$$

The gradient is influenced by whether action a_j is chosen in state s. If action a_j is selected, the gradient increases, making the probability of a_j higher (hence the positive term $1-\sigma_j(s)$. If action a_j is not selected, the gradient decreases, reducing the probability of a_j (hence the negative term). The term scales the gradient according to the cumulative future rewards, encouraging actions that lead to high rewards.

6. Experiments and results

We commit two tests, each of 24 steps to simulate a full demand cycle, where demand increases along the length of the episode. During the tests we simulate 4000-10000 episodes for the agents,according to the convergence rate of the model,to learn and to track their performance which we will later display on a sliding window. Building on the original experiment, we increased the number of stores and tested more environmental parameters to observe the model's learning ability and performance.

As for the Approximate SARSA Agent:

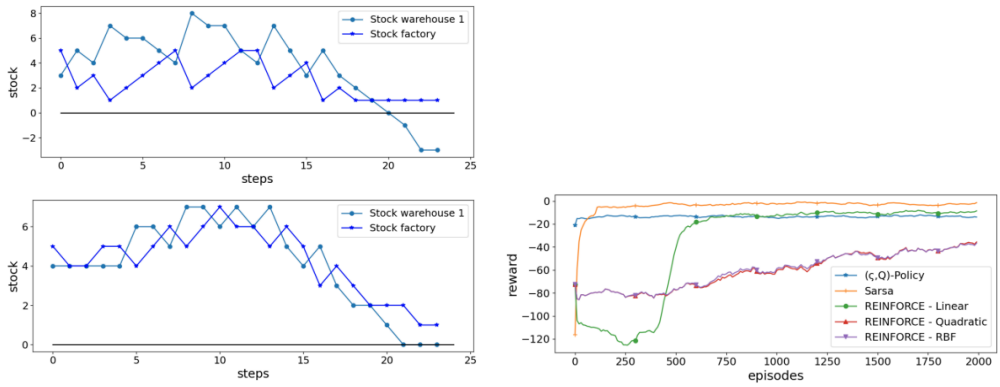

Figure 3. In a simple environment with 1 factor and 1 store, where$c_j$is much larger than max-demand}, a comparison between the (ς, Q)-Policy (top right) and Approximate SARSA (bottom right) strategies reveals that SARSA provides faster training feedback and higher rewards compared to other agents

DOI: 10.54254/2755-2721/2025.30681

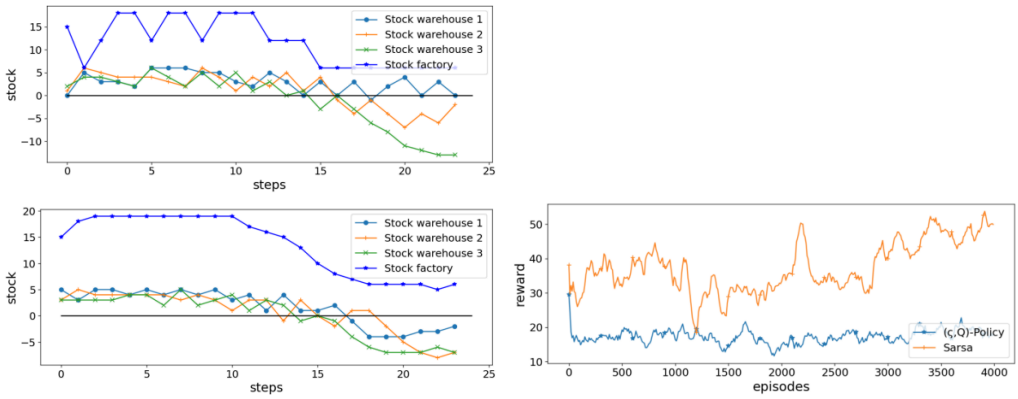

Figure 4. In a medium environment with 1 factor and 3 stores, where$c_j$is is roughly equal to max-demand a comparison between the (ς, Q)-Policy (top) and Approximate SARSA (bottom) strategies shows that Approximate SARSA typically achieves faster convergence and higher rewards compared to the (ς, Q)-Policy

For the RL agent, we further conducted experiments in a one factory, three stores environment as follows:

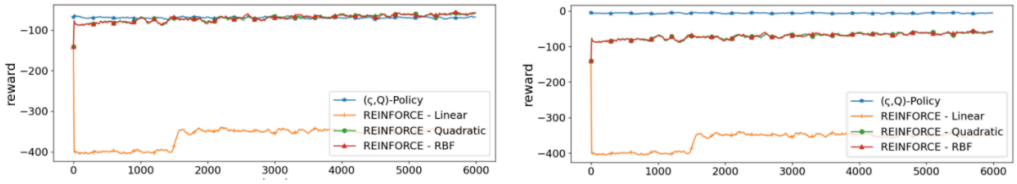

Figure 5. In scenarios where$c_j$is much greater than max-demand(left) and where the two are roughly equal (right), it can be observed that the (ς, Q)-Policy produces better results when provided with appropriate parameters

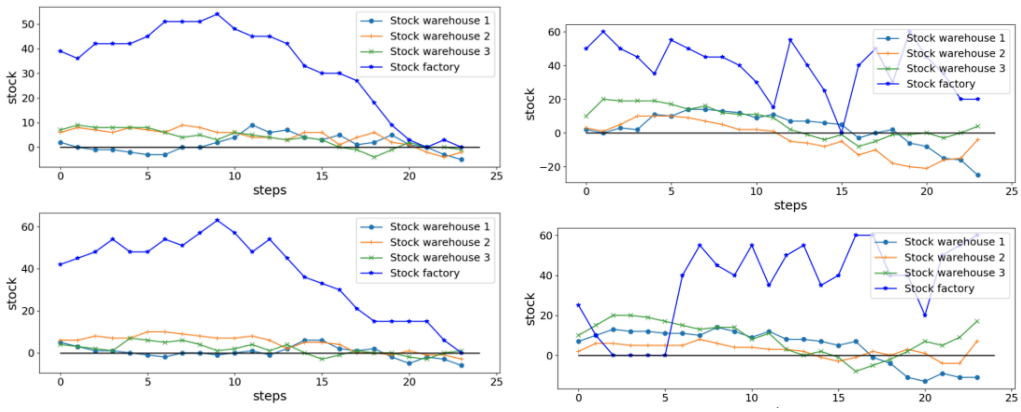

Figure 6. The optimal strategies provided by RL-quadratic (top left) and RL-RBF (bottom left) in the case of low-demand products where$c_j$is much greater than max-demand}. After increasing storage costs (simulating perishable products), the optimal strategies provided by RL-quadratic (top right) and RL-RBF (bottom right) are shown

DOI: 10.54254/2755-2721/2025.30681

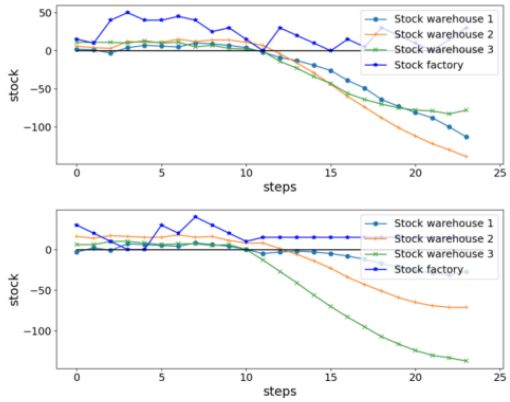

Figure 7. In the scenario where c_j is less than or equal to max-demand (with warehouses 2 and 3 having smaller capacities compared to warehouse 1) and storage costs have been moderately increased, the optimal strategies provided by RL-quadratic (top) and RL-RBF (bottom) are displayed

We noted that the agent learned to increase the factory's storage levels in advance to meet the anticipated rise in demand after a certain period. Additionally, due to the adjustments, the overall storage capacity of the factory was reduced, aligning more closely with the actual sales requirements.

7. Analyse and conclusion

Based on the above results, we further analyzed and explained the learning capabilities of the models, demonstrating that the models can effectively adjust their strategies when faced with different characteristic values. Since SARSA directly adopts a feature value updating strategy, random variables and outliers in the environment are immediately reflected in the strategy. This results in SARSA showing lower resistance to interference, making it more likely to converge to extreme values under the influence of random factors. Therefore, in our experiments, the learning rate for SARSA was set to 10^{-10}, and the exploration rate was kept low (0.1 in the research), based on [2].

Additionally, due to the exponential growth of the state space in Approximate SARSA as the action space increases, and the need to evaluate each possible action during selection, the settings for storage and maximum production were significantly constrained. In practical applications, it may be necessary to convert information into corresponding numerical values before it can be effectively utilized.

On the other hand, RL (Reinforcement Learning) demonstrated stronger resistance to interference. The use of quadratic and RBF (Radial Basis Function) feature processing effectively reduced the impact of outliers on the training model, resulting in more generalized methods. However, the decision-making process is highly sensitive to parameters. Therefore, we also experimented with changing parameters and complicating the environment. To expand the boundaries of numerical settings and enable the use of real data, it was concluded that discretizing actions is both effective and necessary.

We observed that the RL agent was able to maintain an upward trend in rewards during most of the training process, indicating significant room for improvement through training. SARSA, on the other hand, was able to converge within a limited number of steps, showing good training results in simpler environments and responsiveness to environmental changes.

DOI: 10.54254/2755-2721/2025.30681

8. Future work

In the training process, the Agent still exhibits a drop in reward under certain parameter conditions, particularly with Approximate agents. This suggests that further improvements to the algorithm may be needed or different feature extraction methods should be explored.

Additionally, while the proposed model demonstrates some learning capability, it is still in the training phase. The data used is lacking in complex correlations and exhibits prominent feature-oriented characteristics. There is a lack of validation against real-world data for the current training results. Moreover, the supply network is often not a central radiative network but rather a complex, multi-layer nested network. Different periods have varying reward measures, and the training mainly focuses on the necessary costs of production and sales. The model's functionality can be expanded in terms of time and scenarios by adjusting other parameters and actions.

References

[1] Render, B. Stair, R.M. and Hanna, M.E. 2016. Quantitative analysis for management, 9th ed. Pearson Prentice Hall. https: //www.pearson.com/us/higher-education/program/Render-Quantitative-Analysis-for-Manag

[2] Kemmer L, von Kleist H, de Rochebouët D, et al. Reinforcement learning for supply chain optimization [C]//European Workshop on Reinforcement Learning. 2018, 14(10).

[3] Munyaka J B, Yadavalli V S S. Inventory management concepts and implementations: a systematic review [J]. South African Journal of Industrial Engineering, 2022, 33(2): 15-36.

[4] Vrat, P. 2014. Basic Concepts in Inventory Management. Springer Texts in Business and Economics, in: Materials Management, edition 127, chapter 2, pp. 21-36, Springer.

[5] Shih, W. 1979. A general decision model for cost-volume-profit analysis under uncertainty. Accounting Review, 54(4), pp. 687-706.

[6] Naddor, E. 1966. Dimensions in operations research. Operations Research, 14(3), pp. 508-514.

[7] Rannane Y, Mharzi H, El Oualidi M A. A systematic review of the supply chain resilience measurement literature [C]//2022 14th International Colloquium of Logistics and Supply Chain Management (LOGISTIQUA). IEEE, 2022: 1-6.

[8] Farahani R Z, Rezapour S, Drezner T, et al. Competitive supply chain network design: An overview of classifications, models, solution techniques and applications [J]. Omega, 2014, 45: 92-118.

[9] Hearnshaw E J S, Wilson M M J. A complex network approach to supply chain network theory [J]. International Journal of Operations & Production Management, 2013, 33(4): 442-469.

[10] Nagurney A, Dong J, Zhang D. A supply chain network equilibrium model [J]. Transportation Research Part E: Logistics and Transportation Review, 2002, 38(5): 281-303.

[11] Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural computation, 1997, 9(8): 1735-1780.