Beer game
for Multi-agent system 2025 spring

CHENG Feilin [*1]

Abstract

In this assignment, research improvements focus primarily on the following aspects:

(1) Improve the DQN algorithm to improve the learning efficiency, performance, and stability of the DQN network.

(2) Provide multiple agents with different strategies for comparison: an agent using the most basic random strategy; a BS (Base Stock) agent that maintains inventory at a fixed base stock level; a DQN agent trained using DQN; and an MOE ((Mixture of Experts)) agent that combines the BS agent and DQN agent.

(3) Adjusting environmental parameters, such as the length of the supply chain, the initial inventory and the mean Poisson distribution, to examine their respective impacts on the overall supply chain system.

(4) Implement a reward reshaping mechanism to enable multi-agent systems to achieve global optimization rather than just local individual optimization.

1. Improve DQN Algorithm

To improve the capability of the original DQNAgent, we mainly made the following enhancements.

1.1. Implementation of Double DQN

In the baseline DQN, the agent selects actions and estimates target Q-values using the same network parameters, leading to:

$$Target = r + \gamma \max_{a'} Q_{target}(s', a')$$

where Q_{target} is the target network's Q-value for the next state.

[1]2300010850, Peking University. Correspondence to: CHENG Feilin <cflyuke@gmail.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Problem: Direct maximization causes overestimation bias, as errors in Q-value estimates are systematically amplified by the max operator. This results in inflated Q-values, propagating through Bellman updates and degrading policy convergence.

Double DQN Solution: Decouple action selection (using the online network) from action evaluation (using the target network):

$$Target_{DDQN} = r + \gamma Q_{target}(s', \arg\max_{a'} Q_{online}(s', a'))$$

In this way, it can reduce overestimation and stabilized training.

1.2. Implementing of Dueling DQN

In standard DQN, the neural network directly outputs Q-values for each action Q(s,a). This approach fails to explicitly separate the inherent value of the state from the relative importance of individual actions, which can lead to inefficient learning and increased variance in action-value estimates.

Dueling DQN Solution: addresses this by decoupling the Q-value computation into two streams. The state value stream V (s) measures the expected long-term return of being in state s. The advantage stream A (s, a) quantifies the relative benefit of taking action a compared to the average action in state s. The final Q-value is combined as:

$$Q(s, a) = V(s) + (A(s, a) - \frac{1}{|A|} \sum_{a'} A(s, a'))$$

1.3. State Parameter Modification

In the baseline version, the state s_t observed by the agent at time t consisted of:

- order$_{t-1}$: Order quantity placed by the agent at $t-1$
- satisfied_demand$_{t-1}$: Fulfilled demand from downstream partners
- inventory$_t$: Current inventory level

To enhance the agent's situational awareness, we augmented the state representation with two critical supply chain parameters:

- arrived_order$_{t-1}$: Received goods quantity from upstream suppliers at $t-1$
- demand$_{t-1}$: Received order quantity from downstream partners

Explicit inclusion of upstream arrivals (arrived_order) and raw downstream orders (demand) better reflects real-world supply chain dynamics.

1.4. Historical State Window Experiment

We investigated extending the state representation by concatenating consecutive historical states:

$$\hat{s}_t = (s_{t-k}, s_{t-k+1}, ..., s_t)$$

where k denotes the window size of historical states.

Contrary to initial expectations, the historical window approach demonstrated *inferior performance* compared to the standard single-state baseline. Key observations:

- **Dimensionality Challenge**: State dimensionality increased linearly with k (e.g., 5D $\rightarrow$ 15D for $k=3$). Require $3\times$ more network parameters to maintain similar feature resolution.
- **Markov Property Violation**:The supply chain environment inherently satisfies the Markov property. s_t already contains all necessary information (inventory + recent orders), Historical states introduced redundant/irrelevant information.

2. Introduce other intelligent agents

In the base version, apart from the DQNAgent, other agents employ random strategies. This approach prevents agents from effectively predicting future demand or utilizing historical demand information for forecasting, essentially introducing noise into the state representation. Therefore, it's necessary to adjust other agents' strategies to better simulate real-world enterprise behaviors.

2.1. BSAgent

In real-world scenarios, the Base Stock (BS) strategy represents a classical approach in supply chain management, following the principle:

$$\text{Order} = \text{Base Stock} - \text{Current Inventory} + \text{Demand}$$

This algorithm demonstrates stability when inventory levels are high (resulting in zero orders) and can consistently meet downstream demand when the base stock level is appropriately set. However, its limitations include limited flexibility in demand prediction and inevitable cost penalities for maintaining base stock levels.

2.2. MoEAgent

During actual operation of the DQN algorithm, we observe that the agent continues placing orders even when inventory levels are high. This phenomenon may occur because:

- The agent encounters high inventory situations infrequently during training due to rapid inventory depletion
- The network's strong linear relationships lack proper truncation characteristics, making it difficult to reduce orders to zero

To address these issues, we propose the Mixture of Experts (MoE) Agent, which combines the DQN Agent and BS Agent. This hybrid approach selects strategies based on inventory levels, achieving superior performance through:

- Dynamic strategy selection at different inventory stages
- Combined advantages of both learning-based and rule-based approaches

2.3. Performance Comparison

2.3.1. Experiment Setup

We select Enterprise 1 as our research subject, comparing four different strategies:

- Base Stock (BS) Agent
- Baseline DQN Agent
- Improved DQN Agent
- MoE Agent (combining BS and DQN strategies)

The other two enterprises employ either random strategies or BS strategies. Key configuration details include:

- The initial inventory of Enterprise 0 is set to 20 units. Because, under BS policy, Higher initial inventory prevents upstream orders, which is not suitable for reasonable comparison with stochastic strategies.
- All other environmental variables remain at default values

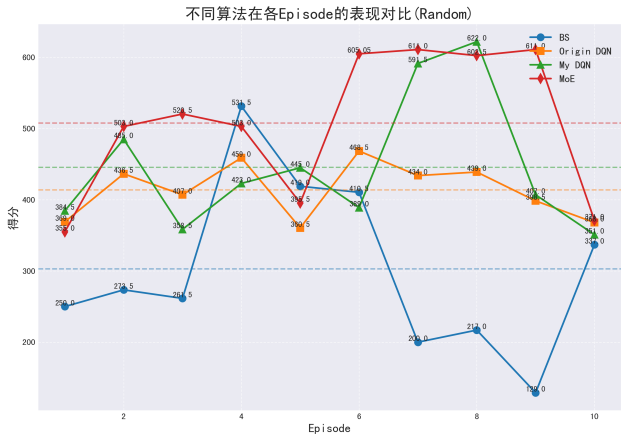

Figure 1. Performance comparison under random strategy environment

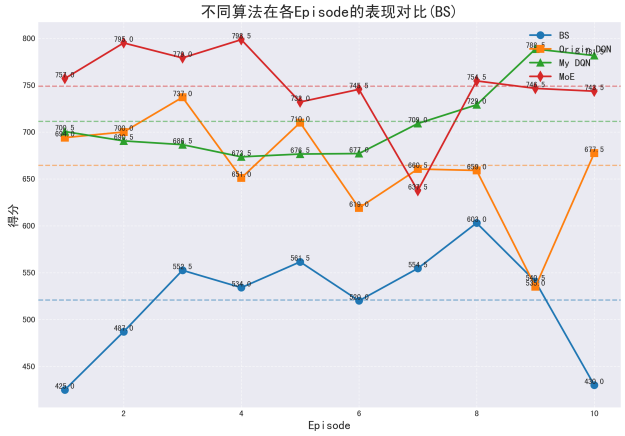

Figure 2. Performance comparison under BS strategy environment

2.3.2. Comparative Results

Figures 1 and 2 present Enterprise 1's performance under different strategy configurations.

The performance ranking demonstrates:

$$\text{MoEAgent} > \text{Improved DQN} > \text{Baseline DQN} > \text{BS}$$

Key observations about environmental strategies:

- **Random Strategy Environment**:
  - Lower mean profits with higher variance
  - Causes difficulty in both demand prediction and order pattern recognition
- **BS Strategy Environment**:
  - Higher overall profits with reduced variance
  - More predictable behavior patterns better reflect real-world operations

The MoE Agent demonstrates consistent advantages through:

- Stable and rational strategy at high inventory levels
- Effective combination of rule-based and learning-based approaches

3. Environmental Parameter Adjustment

3.1. Experimental Setup

In the following experiments, Enterprise 1 employs the MoE strategy while all other enterprises adopt the BS strategy. All parameters are adjusted based on their original values.

3.2. Parameter Analysis

3.2.1. Poisson Distribution Mean Adjustment

- **Adjustment**: Increased mean from 10 to 15
- **Effects**:
  - Market demand increases with higher order volumes
  - Significant profit improvement observed
- **Visualization**: See Figures 3 and 4

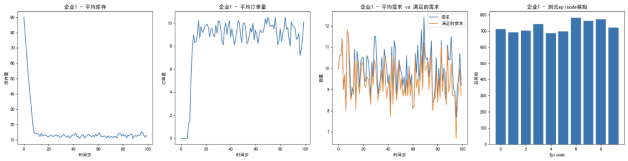

Figure 3. Market demand under poisson lambda equals 10

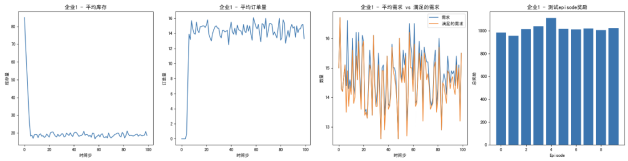

Figure 4. Market demand under poisson lambda equals 15

3.2.2. Supply Chain Length Variation

- **Adjustment**: Increased from 3 to 4 tiers
- **Key Findings**: The revenue situation of Enterprise 1 did not change significantly. This is largely because under the BS strategy, upstream enterprises can always meet the demand of downstream enterprises after maintaining a certain inventory level. Therefore, changes in supply chain length have little noticeable impact on Enterprise 1.

3.2.3. Initial Inventory Level Modification

- **Low Inventory Scenario**: Short-term cost reduction, Long-term profit decrease due to:

$$\text{Reorder Cost} > \text{Holding Cost} \tag{1}$$

- **High Inventory Scenario**: Profit growth can be observed, but diminishing returns occur when:

$$\text{Holding Cost} > \text{Reorder Cost} \tag{2}$$

4. Multi-Agent Global Optimization

4.1. Experimental Configuration

We conduct comprehensive optimization under the following conditions:

- Extended to 4 tiers.
- All enterprises employ the MoE strategy.
- Initial inventory of Enterprise 0 sets to 20 units (maximum order quantity).
- All other parameters maintain baseline values.

4.2. Global Optimization Mechanism

To transcend local optima and achieve global profit maximization, we implement a novel feedback mechanism(Oroojlooyjadid et al., 2017):

$$r'_{i,t} = r_{i,t} + \frac{\beta}{3}(\tau_t - r_{i,t}) \tag{3}$$

Where:

- $r'_{i,t}$: Reshaped reward
- $r_{i,t}$: Original individual reward
- $\tau_t = \frac{\sum_{i=0}^{N-1} r_{i,t}}{N}$: Average reward across all agents at each timestep
- β: Regularization coefficient (empirically set to 0.3)

The reshape reward is only used as a reward for learning during the training process. The accumulated score in the test is still calculated according to the original reward.

4.3. Experimental Results

The implementation demonstrates a certain overall improvement:

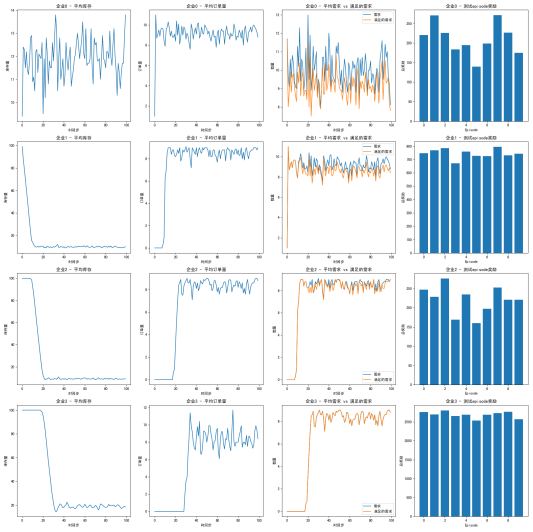

Figure 5. no reshape reward

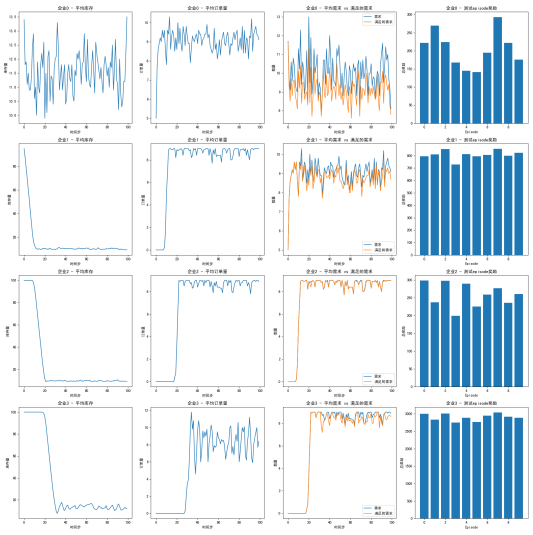

Figure 6. use reshape reward

References

Oroojlooyjadid, A., Nazari, M., Snyder, L. V., and Takác, M. A deep q-network for the beer game with partial information. *CoRR*, abs/1708.05924, 2017. URL http://arxiv.org/abs/1708.05924.