International Scientific Technical and Economic Research (ISSN: 2959-1309)
Vol. 4, No. 2, (March 2026), pp. 21-50
Journal homepage: https://www.istaer.online

RESEARCH ARTICLE DOI: 10.71451/ISTAER2614

Research on Multi-Agent Collaborative Decision-Making Algorithm for Supply Chain Management

Changgeng Li, Zixi Liu*

International Operations, Shinhan University, Gyeonggi-do, Republic of Korea

Abstract: Addressing the key challenges of fuzzy credit allocation, low exploration efficiency, and insufficient robustness in multi-node collaborative decision-making in supply chain management, this paper proposes a hybrid local-global credit allocation multi-agent collaborative decision-making algorithm (HGA-MADDPG). This algorithm introduces a hierarchical graph attention mechanism to dynamically represent the state of the supply chain network topology. It quantifies the contribution of individual actions to sub-chain objectives and system-level indicators through local and global credit networks, respectively, and designs an adaptive fusion weight based on marginal returns to dynamically balance local and global credit. Furthermore, an adversarial disturbance and resilient training architecture is constructed, including modeling three types of disturbances: demand mutation, node failure, and transportation delay, as well as adversarial agent injection, a dynamic environment replay buffer, and a two-stage training strategy. In a baseline scenario of a four-level supply chain and a dynamic environment driven by real data based on SCDL and WSN, compared with eight baseline algorithms, experimental results show that HGA-MADDPG achieves a total cost reduction rate of 26.2%, a service level improvement rate of 42.8%, and a stockout rate controlled at 3.2%. In the extreme scenario of triple perturbation, the cost deviation rate (29.6%) and recovery time (58 hours) are significantly better than existing methods. It still maintains a cost reduction rate of 21.5% in a 120-node ultra-large-scale supply chain. Ablation experiments and scalability analysis further verify the effectiveness of each core module.

Keywords: Supply chain management; Multi-agent collaboration; Credit allocation; Graph attention network; Multi-agent deep deterministic policy gradient

How to Cite: Li, C., & Liu, Z. (2026). Research on Multi-Agent Collaborative Decision-Making Algorithm for Supply Chain Management. *International Scientific Technical and Economic Research, 4*(2), 21–50. https://doi.org/10.71451/ISTAER2614

Article history: Received: 19 Jan 2026; **Revised:** 25 Feb 2026; **Accepted:** 30 Mar 2026; **Published:** 04 Apr 2026

1. INTRODUCTION

Modern supply chains have evolved into complex network systems where suppliers, manufacturers, distributors, and retailers are tightly coupled through material, information, and capital flows. The "bullwhip effect" causes decision changes at any node to amplify along the network [1]. Simultaneously, demand fluctuations, transportation delays, equipment failures,

* **Corresponding author:** Zixi Liu, International Operations, Shinhan University, Gyeonggi-do, Republic of Korea. Email: 15947628017@163.com

and external shocks create significant uncertainty [2],[3]. Traditional centralized decision-making models assume an omniscient control center, but information silos, privacy concerns, and computational scalability make them impractical [4]. Distributed models grant autonomy to each node but often lead to local optimization at the expense of global efficiency [5],[6],[7]. Thus, achieving global collaboration while respecting local autonomy is a core scientific problem. Multi-agent systems provide an ideal framework, where each node acts as an autonomous agent capable of local observation and decision-making.

Existing supply chain algorithms fall into three categories with distinct limitations. Centralized optimization methods can theoretically obtain global optima but suffer from exponential computational complexity and require complete information sharing that is often infeasible in practice [8]. Distributed heuristic strategies inventory policies are simple and widely used but rely on steady-state assumptions and cannot adapt to non-stationary fluctuations or topology changes. Multi-agent reinforcement learning methods have shown promise but have three major limitations when applied to supply chains [9],[10],[11]. First, credit assignment ambiguity: single Critic frameworks cannot distinguish individual actions' contributions to local versus global goals; a retailer's expedited order may relieve its own shortage but crowd upstream capacity, yet receive a positive global evaluation. Second, inefficient collaborative exploration: independent exploration leads to extremely low coverage of joint action spaces, making it difficult to discover synchronized upstream-downstream behaviors. Third, insufficient robustness: policies trained in stable environments fail dramatically when facing demand surges, node failures, or transportation delays, sometimes causing cascading failures. The root cause is that existing algorithms fail to model the hierarchical structure between local operational dependencies and global strategic goals, and lack active training mechanisms for disturbed environments.

To address these limitations, this paper proposes a hybrid local-global credit allocation multi-agent collaborative decision-making algorithm (HGA-MADDPG) with an adversarial and resilient training architecture. The core innovations are threefold. Algorithmically, a hybrid credit allocation mechanism breaks through the single Critic framework: a local credit network provides instant feedback for each agent's subchain goals, solving credit delay problems; a global credit network evaluates joint actions' impact on total system cost and service level, capturing inter-agent coupling; and adaptive fusion weights based on marginal returns allow agents to dynamically adjust their reliance on local versus global credit. Additionally, a collaborative exploration mechanism using a learnable matrix creates positive correlation in exploration noise across upstream and downstream agents, accelerating discovery of efficient joint strategies. Architecturally, an adversarial disturbance and resilient training framework models three typical disturbances (demand mutation, node failure, transportation delay), injects adversarial agents to generate extreme scenarios during training, prioritizes challenging samples via a dynamic environment replay buffer, and uses a two-stage adversarial-cooperative training strategy to balance performance and robustness. For validation, a four-level supply chain benchmark with real data-driven dynamics is constructed, eight baseline algorithms are compared, and nine evaluation metrics across cooperative efficiency, decision quality, and algorithm performance are designed. Extensive comparative, ablation, robustness, and scalability experiments systematically verify the algorithm's superiority.

2. HIERARCHICAL GRAPH ATTENTION MECHANISM FOR MULTI-AGENT STATE REPRESENTATION

To effectively characterize the complex coupling relationships among multiple agents in supply chain systems, this paper proposes a state representation method based on a hierarchical graph attention mechanism. A supply chain is essentially a heterogeneous directed graph composed of multiple types of nodes (such as suppliers, factories, warehouses, retailers) and two types of edges (logistics edges and information flow edges) [12],[13]. Let the supply chain

network at time t be denoted as $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_L, \mathcal{E}_I)$, where $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ represents the set of N agent nodes, $\mathcal{E}_L$ denotes the logistics edges (representing the physical flow of materials or products from upstream to downstream), and $\mathcal{E}_I$ denotes the information flow edges (representing the reverse transmission of orders, inventory status, etc.). The raw state vector $\mathrm{s}_i^{(t)}$ of each node v_i includes its local observations: current inventory level I_i, in-transit inventory T_i, demand rate d_i, production capacity or supply upper bound C_i, and order quantity o_{ij} to neighboring nodes. The non-stationarity of the supply chain topology (such as supplier switching, transportation delay variations) requires the state representation method to possess dynamic adaptability [14][15].

In a single-layer graph attention network, each node aggregates neighbor information through attention mechanisms, but it struggles to distinguish the hierarchical relationship between local operational dependencies and global strategic objectives in supply chains [16],[17],[18]. To address this, this paper designs a hierarchical graph attention mechanism that decomposes the representation process into a local encoding layer and a global fusion layer. First, in the local encoding layer, each node v_i focuses only on its direct upstream and downstream neighbors (the one-hop neighbor set $\mathcal{N}_i$) and computes local attention coefficients. For node i and its neighbor $j \in \mathcal{N}_i$, the raw attention score e_{ij} is defined as:

$$e_{ij} = \mathrm{LeakyReLU}\left(\mathrm{a}_l^{\top}\left[\mathrm{W}_l\mathrm{s}_i \parallel \mathrm{W}_l\mathrm{s}_j\right]\right) \tag{1}$$

Where $\mathrm{s}_i, \mathrm{s}_j \in \mathbb{R}^{d_s}$ are the raw state vectors of the nodes, $\mathrm{W}_l \in \mathbb{R}^{d_h \times d_s}$ is a trainable weight matrix of the local layer that maps the input to a hidden space of dimension d_h, $\mathrm{a}_l \in \mathbb{R}^{2d_h}$ is the local attention vector, $\parallel$ denotes vector concatenation, and LeakyReLU serves as the non-linear activation function. Subsequently, the local attention weights are obtained via softmax normalization:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp\left(e_{ik}\right)} \tag{2}$$

The local state representation $\mathrm{h}_i^{(l)}$ of node i is computed as the weighted sum of its own features and those of its neighbors:

$$\mathrm{h}_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i \cup \{i\}} \alpha_{ij}\,\mathrm{W}_l\mathrm{s}_j\right) \tag{3}$$

Where σ is the ELU activation function. This local representation captures typical local dependencies in supply chains, such as a warehouse's simultaneous response to upstream factory supply capacity and downstream retailer demand.

In the global fusion layer, rather than simply concatenating all local representations, a virtual global node v_g is introduced as an information aggregation center [19],[20]. The global node computes a second-layer attention from all local representations $\mathrm{H}^{(l)} = \{\mathrm{h}_1^{(l)}, \mathrm{h}_2^{(l)}, \dots, \mathrm{h}_N^{(l)}\}$ to generate a global context vector. For any node i, the attention score f_i to the global node is calculated as:

$$f_i = \mathrm{LeakyReLU}\left(\mathrm{a}_g^{\top}\left[\mathrm{W}_g\mathrm{h}_i^{(l)} \parallel \mathrm{W}_g\mathrm{h}_g\right]\right) \tag{4}$$

Where h_g is a learnable embedding of the global node, $\mathrm{W}_g \in \mathbb{R}^{d_g \times d_h}$, and $\mathrm{a}_g \in \mathbb{R}^{2d_g}$. The normalized global attention weights are:

$$\beta_i = \frac{\exp(f_i)}{\sum_{k=1}^{N} \exp\left(f_k\right)} \tag{5}$$

The final global state representation z_g is the weighted sum of all local representations:

$$z_g = \tanh\left(\sum_{i=1}^{N} \beta_i W_g h_i^{(l)}\right) \tag{6}$$

Meanwhile, for collaborative decision-making, each node requires both its own local information and a global perspective. Therefore, the final representation z_i of node i is obtained by concatenating the local representation $h_i^{(l)}$ with the broadcasted global information z_g, followed by an output mapping:

$$z_i = W_o\left[h_i^{(l)} \,\|\, z_g\right] \tag{7}$$

Where $W_o \in \mathbb{R}^{d_z \times (d_h + d_g)}$ is the output mapping matrix. This hierarchical structure enables upstream nodes (such as factories) to perceive downstream sales fluctuations through the global channel, while downstream nodes can also obtain upstream capacity constraints, avoiding the information dilution problem caused by flat attention mechanisms.

Supply-demand relationships in supply chains change dramatically over time, and static attention weights under fixed graph structures cannot respond to node disturbances (such as temporary warehouse closure) or flow mutations (such as demand spikes) [21]. To address this, this paper proposes a dynamic weight update mechanism that makes the attention coefficients sensitive to the real-time operating state of the supply chain. Specifically, a time-varying adjustment factor $\gamma_{ij}^{(t)}$ is introduced to modify the raw attention score:

$$\tilde{e}_{ij}^{(t)} = e_{ij}^{(t)} + \lambda \cdot \Delta_{ij}^{(t)} \tag{8}$$

Where $\Delta_{ij}^{(t)}$ is determined by the supply-demand matching error between node i and node j. Let the expected supply quantity from node i to node j be $\hat{p}_{ij}^{(t)}$ and the actual flow quantity be $p_{ij}^{(t)}$. Then:

$$\Delta_{ij}^{(t)} = \tanh\left(\frac{p_{ij}^{(t)} - \hat{p}_{ij}^{(t)}}{\max\left(\hat{p}_{ij}^{(t)}, \epsilon\right)}\right) \tag{9}$$

This variable ranges over $(-1,1)$, where a positive value indicates supply exceeding expectations (requiring reduced attention weight to suppress overstocking), and a negative value indicates supply shortage (requiring increased attention to trigger urgent coordination). The parameter λ controls the disturbance sensitivity. Meanwhile, for a node v_k affected by an external disturbance (such as node failure), all its neighboring edges $\Delta_{ik}^{(t)}$ are additionally multiplied by a decay factor $\exp(-\tau \cdot \text{delay}_k)$, where delay_k is the disturbance duration and τ is the time constant. This design enables the attention mechanism to dynamically adjust its focus on abnormal nodes: short-term disturbances are responded to quickly, while the influence of long-term failed nodes gradually decays, preventing erroneous information from propagating continuously.

3. HYBRID LOCAL-GLOBAL CREDIT ASSIGNMENT FOR MULTI-AGENT COLLABORATIVE DECISION-MAKING ALGORITHM (HGA-MADDPG)

3.1 Base Framework: Adaptability and Limitations of MADDPG

The Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm adopts a centralized training with decentralized execution framework, where each agent maintains its own Actor network and Critic network [22],[23]. During training, the Critic can access the states and actions of all agents, thereby mitigating the environmental non-stationarity issue. In supply chain scenarios, MADDPG has been preliminarily applied to inventory replenishment and order allocation tasks. However, directly applying MADDPG suffers from two critical limitations. First, credit assignment is ambiguous: the global Critic in MADDPG outputs a single value estimate for each agent, which cannot distinguish the contribution of an individual agent's action to local sub-chain objectives (such as inventory turnover rate of a specific warehouse) versus global system objectives (such as total cost or service level). As the supply chain scale increases, excessive replenishment by a retailer might be misjudged by the global Critic as a positive contribution (because it alleviates stockouts), but in reality, this action causes capacity occupation at upstream factories, harming overall system efficiency [24],[25]. Second, collaboration efficiency is low: MADDPG employs independent Gaussian noise for action exploration, resulting in uncorrelated exploration directions across agents. This leads to extremely low coverage efficiency of the joint action space, making it difficult to discover effective collaborative strategies (such as synchronizing order rhythms between upstream and downstream agents to smooth the bullwhip effect). To address these issues, this section proposes a Hybrid Local-Global Credit Assignment for Multi-Agent Collaborative Decision-Making Algorithm (HGA-MADDPG).

3.2 Core Algorithmic Improvements

3.2.1 Local Credit Network: Quantifying Individual Action Contributions to Local Sub-Chain Objectives

To quantify the contribution of each agent to its local sub-chain, this paper introduces a local credit network Q_i^{loc} for each agent. This network takes as input the local observation o_i of agent i and its own action a_i, and outputs a scalar value representing the expected contribution of this action to the local sub-chain objective, ignoring global influences. The local sub-chain is defined as the set of nodes whose topological distance from agent i does not exceed K hops, denoted $\mathcal{L}_i$. Its objective is typically a weighted sum of local inventory cost C_i^{inv} and local shortage rate S_i^{short}. The local credit network is trained using TD error, where the target value is composed of the local reward r_i^{loc} and the next-step local value estimate:

$$y_i^{\text{loc}} = r_i^{\text{loc}} + \gamma \cdot Q_i^{\text{loc}}\left(o_i', a_i'; \theta_i^{\text{loc}-}\right) \tag{10}$$

Where γ is the discount factor and $\theta_i^{\text{loc}-}$ are the target network parameters. The local reward r_i^{loc} is defined as:

$$r_i^{\text{loc}} = -\left(\alpha \cdot \frac{I_i}{I_i^{\max}} + \beta \cdot \mathbb{I}\left(\text{shortage}_i > 0\right)\right) \tag{11}$$

Here, I_i is the current inventory level, $I_i^{\max}$ is the maximum inventory capacity, $\mathbb{I}(\cdot)$ is the indicator function, and α and β are weight coefficients. The gradient update for the local credit network is:

$$\nabla_{\theta_i^{\text{loc}}} \mathcal{L}_i^{\text{loc}} = \mathbb{E}\left[\left(y_i^{\text{loc}} - Q_i^{\text{loc}}(o_i, a_i)\right) \nabla_{\theta_i^{\text{loc}}} Q_i^{\text{loc}}(o_i, a_i)\right] \tag{12}$$

This network enables each agent to receive immediate credit feedback directly related to its local sub-chain, resolving the credit delay problem in long chains.

3.2.2 Global Credit Network: Evaluating Joint Action Impact on System-Level Metrics

Unlike the single Critic in MADDPG, this paper designs a global credit network Q^{glb} to evaluate the impact of the joint actions of all agents on system-level metrics. The global credit network takes as input the global state $\mathrm{s} = [\mathrm{s}_1, \mathrm{s}_2, \ldots, \mathrm{s}_N]$ and the joint action $\mathrm{a} = [a_1, a_2, \ldots, a_N]$, and outputs the expected total system return. The system-level reward r^{glb} is defined as:

$$r^{\text{glb}} = -\left(w_1 \cdot \frac{\sum_i C_i^{\text{op}}}{C^{\text{base}}} + w_2 \cdot \frac{\sum_i \text{delay}_i}{D^{\text{base}}} + w_3 \cdot \text{service\_penalty} \right) \tag{13}$$

Where C_i^{op} is the operational cost of agent i (including warehousing, transportation, and ordering costs), delay_i is the order response delay, service_penalty is the penalty term when the service level fails to meet the target, and C^{base} and D^{base} are normalization baselines. The training target for the global credit network is:

$$y^{\text{glb}} = r^{\text{glb}} + \gamma \cdot Q^{\text{glb}}\left(\mathrm{s}', \mathrm{a}'; \theta^{\text{glb}-}\right) \tag{14}$$

Its loss function is:

$$\mathcal{L}^{\text{glb}} = \mathbb{E}\left[\left(y^{\text{glb}} - Q^{\text{glb}}(\mathrm{s}, \mathrm{a})\right)^2\right] \tag{15}$$

The global credit network captures coupling effects among agents. For example, when two adjacent warehouses simultaneously place large replenishment orders, although their respective local credit networks may provide positive evaluations (each warehouse has sufficient stock), the global credit network outputs a low score due to upstream capacity overload and competition for transportation resources, thereby suppressing such non-collaborative behavior.

3.2.3 Adaptive Fusion Weight: Dynamic Credit Assignment Based on Marginal Benefit

Relying solely on either local or global credit networks has drawbacks: local networks may lead to myopic behavior (excessive focus on current inventory while ignoring long-term strategy), while global networks may suffer from credit dilution, reducing learning efficiency. To address this, this paper proposes an adaptive fusion mechanism that dynamically computes the weight each agent should place on local versus global credit during decision-making. The fusion weight λ_i is output by a small network $\phi_i(o_i)$ conditioned on the agent's current state o_i, and mapped to the interval $(0, 1)$ via a sigmoid function. The comprehensive Critic output for agent i is:

$$Q_i^{\text{total}}(o_i, a_i, \mathrm{s}, \mathrm{a}) = \lambda_i \cdot Q_i^{\text{loc}}(o_i, a_i) + (1 - \lambda_i) \cdot Q^{\text{glb}}(\mathrm{s}, \mathrm{a}) \tag{16}$$

The learning of the weight is based on marginal benefit comparison. Define the marginal contribution of agent i to the joint action as:

$$\Delta_i = Q^{\text{glb}}(\mathrm{s}, \mathrm{a}) - Q^{\text{glb}}\left(\mathrm{s}, \mathrm{a}_{-i}, a_i^{\text{ref}}\right) \tag{17}$$

Where a_{-i} denotes the actions of all agents except i, and a_i^{ref} is a reference action (such as the previous time step's action or the default policy action). The marginal contribution Δ_i reflects the actual impact of agent i's action on the global system objective. Simultaneously, define the local relative advantage:

$$\delta_i = Q_i^{\text{loc}}(o_i, a_i) - \bar{Q}_i^{\text{loc}} \tag{18}$$

Where $\bar{Q}_i^{\text{loc}}$ is the historical average local value of agent i under its past policy. The update rule for the fusion weight is designed as follows: when the marginal contribution Δ_i is significantly positive (indicating that the agent's action is beneficial to the global objective) and

the local relative advantage δ_i is negative (indicating that the action appears disadvantageous from the local perspective), λ_i should be increased to enhance global credit guidance. Conversely, when the local advantage is positive but the marginal contribution is negative, λ_i should be decreased to suppress myopic behavior. Specifically, the gradient for weight adjustment is backpropagated as:

$$\nabla_{\psi_i}\mathcal{L}_i^{\lambda} = -\mathbb{E}\left[\nabla_{\lambda_i} Q_i^{\text{total}} \cdot \nabla_{\psi_i}\lambda_i\right] \tag{19}$$

Where ψ_i are the parameters of the weight network. This design enables credit assignment to adaptively adjust according to the agent's role in the current supply chain state (whether it is a local bottleneck node or a global hub node), avoiding the bias introduced by fixed weights.

3.3 Collaborative Exploration Mechanism: Coordinated Noise Injection via Action Preference Mapping

In MADDPG, action exploration for each agent typically employs independent Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$, i.e., $a_i^{\text{explore}} = \mu_i(o_i) + \epsilon_i$. This approach ignores the correlation among agents' exploration directions, resulting in low coverage efficiency of the joint action space. For example, in supply chains, simultaneous increases in replenishment quantities by upstream and downstream nodes may produce synergistic benefits, but independent exploration rarely triggers such "co-increase" combinations. This paper proposes a collaborative exploration mechanism based on action preference mapping. First, define a collaborative exploration matrix $\mathrm{M} \in \mathbb{R}^{N\times N}$, where element M_{ij} represents the mapping strength from the exploration noise of agent j to agent i. This matrix is initialized based on the supply chain topology: neighboring nodes (connected by logistics or information flows) are assigned non-zero M_{ij} values, while non-neighboring nodes are assigned zero. During exploration, a base noise $\eta_i \sim \mathcal{N}(0,1)$ is first generated for each agent, and then the collaborative noise vector ϵ is generated via a linear transformation:

$$\epsilon = \mathrm{M}\eta + \zeta, \zeta \sim \mathcal{N}\left(0, \sigma_{\text{ind}}^2 \mathrm{I}\right) \tag{20}$$

Where $\eta = [\eta_1, \ldots, \eta_N]^{\top}$, and ζ is an independent residual noise term that ensures exploration diversity. The final action with collaborative noise is expressed as:

$$a_i^{\text{explore}} = \mu_i(o_i) + \sigma_{\text{base}} \cdot \tanh(\epsilon_i) \tag{21}$$

Where σ_{base} is the base exploration magnitude, and the tanh function constrains the noise within a reasonable range. The collaborative exploration matrix M is not fixed but adaptively adjusted during training. Define the global exploration reward $R^{\text{explore}} = Q^{\text{glb}}(\mathrm{s}, \mathrm{a}^{\text{explore}}) - Q^{\text{glb}}(\mathrm{s}, \mathrm{a}^{\text{mean}})$, i.e., the difference in global value between the collaborative exploration action and the mean action. The update of M follows the gradient:

$$\nabla_{\mathrm{M}}\mathcal{L}^{\text{explore}} = -\mathbb{E}\left[R^{\text{explore}} \cdot \nabla_{\mathrm{M}} \log p(\epsilon \mid \mathrm{M})\right] \tag{22}$$

Where $p(\epsilon \mid \mathrm{M})$ is the probability density of the noise distribution given M. This mechanism enables the algorithm to automatically learn which agents should explore collaboratively (such as neighboring nodes should produce positively correlated exploration directions), thereby accelerating the discovery of efficient joint policies.

3.4 Algorithm Pseudocode and Convergence Discussion

The overall training procedure of HGA-MADDPG is presented in Algorithm 1. In each episode, the algorithm sequentially performs: state acquisition, collaborative noise injection, action execution, reward collection, experience storage, and network updates by sampling from the experience replay buffer. The Critic networks include the local credit network Q_i^{loc}, the

global credit network Q^{glb}, and the fusion weight network ϕ_i. The Actor network is μ_i. All target networks are updated using soft updates: $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$, where $\tau \ll 1$.

Algorithm 1: HGA-MADDPG

```
Input: Number of agents N, maximum training episodes E, maximum steps per episode T,
       discount factor γ, soft update rate τ, batch size B
Initialize: For each agent i: Actor network μ_i, local Critic network Q_i^loc, weight network φ_i
            Global Critic network Q^glb, collaborative exploration matrix M
            Initialize all target networks: μ_i^-, Q_i^loc-, Q^glb-
            Initialize experience replay buffer D

for episode = 1 to E do
    Obtain initial global state s = [s_1, ..., s_N]
    for t = 1 to T do
        // Action selection with collaborative exploration
        for each agent i do
            μ_i(o_i) ← mean action from Actor
        end for
        Generate base noise vector η ~ N(0, I)
        Compute collaborative noise ε = M η + ζ, ζ ~ N(0, σ_ind^2 I)
        for each agent i do
            a_i = μ_i(o_i) + σ_base · tanh(ε_i)
            a_i ← clip(a_i, a_min, a_max)
        end for
        Execute joint action a = [a_1, ..., a_N], obtain global reward r^glb,
        local rewards {r_i^loc}, next state s'
        Store experience (s, a, r^glb, {r_i^loc}, s') in D

        // Network updates (every d steps)
        if len(D) ≥ B then
            Sample batch from D

            // Update global Critic
            y^glb = r^glb + γ · Q^glb-(s', a'; θ^glb-)
            L^glb = MSE(y^glb, Q^glb(s, a))
            Update Q^glb via L^glb

            // Update local Critics
            for each agent i do
                y_i^loc = r_i^loc + γ · Q_i^loc-(o_i', a_i'; θ_i^loc-)
                L_i^loc = MSE(y_i^loc, Q_i^loc(o_i, a_i))
                Update Q_i^loc via L_i^loc
            end for

            // Update fusion weight networks
            Compute marginal contribution Δ_i and local relative advantage δ_i
            Update φ_i via L_i^λ = -E[∇_{λ_i} Q_i^total · ∇_{ψ_i} λ_i]

            // Update Actor networks (using comprehensive Critic)
            for each agent i do
                λ_i = φ_i(o_i)
                Q_i^total = λ_i · Q_i^loc(o_i, a_i) + (1-λ_i) · Q^glb(s, a)
                ∇J_i ≈ E[ ∇_{a_i} Q_i^total · ∇_{θ_i^μ} μ_i(o_i) ]
                Update μ_i via gradient ascent
            end for

            // Update collaborative exploration matrix
            Compute R^explore = Q^glb(s, a_explore) - Q^glb(s, a_mean)
            Update M via L^explore = -E[ R^explore · ∇_M log p(ε|M) ]

            // Soft update target networks
            for each agent i do
                θ_i^μ- ← τ θ_i^μ + (1-τ) θ_i^μ-
                θ_i^loc- ← τ θ_i^loc + (1-τ) θ_i^loc-
            end for
            θ^glb- ← τ θ^glb + (1-τ) θ^glb-
        end if
        s ← s'
    end for
end for
```

Regarding convergence, HGA-MADDPG can be viewed as an extension of the MADDPG framework with multi-objective credit assignment and adaptive exploration. Under standard

assumptions (unbiased policy gradient, learning rates satisfying the Robbins-Monro conditions, finite state-action space with a Markov chain that is ergodic), the alternating updates of the local and global credit networks constitute a multi-timescale stochastic approximation process [26]. Specifically, the global credit network converges at a slower timescale to the true joint action value function $Q^{\text{glb},*}(\mathrm{s},\mathrm{a})$, while the local credit networks converge at a faster timescale to each agent's local value function $Q_i^{\text{loc},*}(o_i, a_i)$. The fusion weight network further optimizes after the convergence of both, making Q_i^{total} a state-dependent weighted approximation of Q^{glb}. The update of the collaborative exploration matrix M is based on the likelihood ratio method for policy gradients, and its convergence can be guaranteed by the convergence theory of natural gradients or evolution strategies. Theoretically, when all Critic networks reach optimality, the Actor update direction aligns with the expected joint policy gradient direction, enabling the algorithm to converge to a local Nash equilibrium [27],[28]. Compared to the original MADDPG, HGA-MADDPG alleviates the credit assignment variance problem in multi-agent scenarios through the introduction of local credit and adaptive fusion, thereby achieving faster convergence speed and superior equilibrium quality.

4. ADVERSARIAL PERTURBATION AND RESILIENT TRAINING ARCHITECTURE FOR SUPPLY CHAIN ROBUSTNESS

4.1 Modeling of Typical Supply Chain Perturbations

Real-world supply chain systems inevitably face various internal and external perturbations that can significantly degrade the performance of multi-agent collaborative decision-making algorithms [29]. To systematically enhance algorithmic robustness, this paper first mathematically models three typical types of perturbations. The first type is demand surge, which refers to a sudden and drastic shift in market demand at downstream nodes. Let the base demand of retailer i at time t be $d_i^{(t)}$. The demand surge model can be expressed as $\tilde{d}_i^{(t)} = d_i^{(t)} \cdot (1 + \rho_i^{(t)})$, where $\rho_i^{(t)}$ is the surge magnitude, whose distribution follows a mixture distribution: $\rho_i^{(t)} \sim \kappa \cdot \mathcal{U}(-\rho_{\max}, \rho_{\max}) + (1-\kappa) \cdot \delta(0)$. Here, κ is the probability of surge occurrence, $\rho_{\max}$ is the maximum surge magnitude, $\mathcal{U}$ denotes the uniform distribution, and $\delta(0)$ is the Dirac delta function representing the no-surge state. Demand surges typically have a persistent effect, introducing an exponentially decaying memory term: $d_i^{\text{actual}}(t) = \tilde{d}_i^{(t)} + \eta \cdot e^{-(t-t_0)/\tau_d} \cdot (\tilde{d}_i^{(t_0)} - d_i^{(t_0)})$, where t_0 is the surge occurrence time, τ_d is the recovery time constant, and η is the decay coefficient.

The second type is node failure, where a node in the supply chain (such as a warehouse or factory) temporarily or permanently loses its operational capability due to an unexpected event. The node failure status is represented by a binary variable $z_i^{(t)} \in \{0,1\}$, where $z_i^{(t)} = 0$ indicates normal operation and $z_i^{(t)} = 1$ indicates failure. The behavioral model for a failed node is: when $z_i^{(t)} = 1$, the node can neither receive upstream materials (inbound quantity is zero) nor ship to downstream nodes (outbound quantity is zero), and its local decision action a_i is forced to zero or a hold value. Failure occurrences follow a Poisson process. Let the mean time between failures be T_{MTBF}; then the failure probability is $P(z_i^{(t+1)} = 1 \mid z_i^{(t)} = 0) = 1 - \exp(-\Delta t / T_{\text{MTBF}})$. The recovery process after failure also follows an exponential distribution, with recovery probability $P(z_i^{(t+1)} = 0 \mid z_i^{(t)} = 1) = 1 - \exp(-\Delta t / T_{\text{MTTR}})$, where T_{MTTR} is the mean time to recovery. Node failures trigger cascade effects: upstream nodes experience inventory buildup due to inability to ship to the failed node, while downstream nodes face stockout risks due to supply disruption. This requires the multi-agent system to possess rapid rerouting and task reassignment capabilities.

The third type is transportation delay, where the actual transportation time between nodes exceeds the planned time. Let the planned transportation time from node i to node j be $L_{ij}^{(0)}$. The actual transportation time is modeled as $L_{ij}^{(t)} = L_{ij}^{(0)} + \Delta_{ij}^{(t)}$, where the delay increment $\Delta_{ij}^{(t)}$ consists of two components: baseline fluctuation $\xi_{ij}^{(t)} \sim \mathcal{N}(0, \sigma_L^2)$ and a sudden congestion term $\nu_{ij}^{(t)} = \delta_{ij} \cdot \exp\left(-(t - t_{ij}^{\text{cong}})/\tau_c\right) \cdot \mathcal{U}(0, L_{\max})$. Here, δ_{ij} is an indicator variable for congestion events, t_{ij}^{cong} is the congestion start time, and τ_c is the congestion decay time constant. Transportation delay directly affects the inventory state equation: the in-transit inventory $T_{ij}^{(t)}$ update must account for the delay window, meaning that goods shipped at time t will arrive at time $t + L_{ij}^{(t)}$ rather than at a fixed time. These three types of perturbations can occur individually or in combination (such as a demand surge occurring simultaneously with a node failure), constituting a comprehensive stress-testing environment for collaborative decision-making algorithms.

4.2 Adversarial Agent Injection

To proactively enhance the robustness of the main algorithm rather than passively relying on perturbation data, this paper proposes an adversarial agent injection mechanism. In this mechanism, in addition to the main agents that execute normal supply chain decisions, a set of adversarial agents $\mathcal{A}^{\text{adv}} = \{\text{adv}_1, \text{adv}_2, \dots, \text{adv}_M\}$ is introduced. The objective of these adversarial agents is to minimize the collaborative decision-making performance of the main system. Adversarial agents can manipulate specific variables in the supply chain environment, including: tampering with demand forecasts (providing false demand signals to main agents), triggering node failure simulations (forcibly shutting down certain nodes), and injecting additional transportation delays. Each adversarial agent adv_k possesses its own policy network $\mu_k^{\text{adv}}(o_k^{\text{adv}})$, where its observation o_k^{adv} includes a partial view of the current global supply chain state and the historical actions of the main agents. The reward function for adversarial agents is designed to maximize the cost of the main system: $r^{\text{adv}} = -\sum_i r_i^{\text{main}}$, where r_i^{main} is the original reward obtained by main agent i. Adversarial agents and main agents engage in a zero-sum game, and their training also employs policy gradient methods, but with the opposite gradient direction.

To simulate realistic malicious attacks or extreme operational scenarios, adversarial agents are subject to behavioral constraints that make their perturbations physically or economically feasible. Specifically, the action space a_k^{adv} of an adversarial agent includes the perturbation type selection (demand/node/transportation) and the perturbation intensity θ_k, satisfying $\theta_k \in [0, \theta_{\max}]$. A perturbation cost constraint is introduced: each unit of perturbation intensity imposed by an adversarial agent consumes c_k of an "attack budget," and the total budget B^{adv} is limited, i.e., $\sum_k \theta_k \cdot c_k \leq B^{\text{adv}}$. This constraint forces adversarial agents to apply perturbations at the most critical time points and to the most vulnerable links in the supply chain, thereby exposing the strategic weaknesses of the main algorithm. Adversarial agents and main agents are trained alternately: in each training round, the main agent policies are first fixed while adversarial agents are updated to find the most effective perturbation patterns; then the adversarial agent policies are fixed while the main agents are updated to defend against these perturbations. This alternating adversarial process resembles the game mechanism of Generative Adversarial Networks (GANs), enabling the main agents to experience various extreme scenarios during training, thereby achieving stronger robustness when deployed in practice.

4.3 Resilient Training Architecture

4.3.1 Dynamic Environment Replay Buffer

Standard experience replay buffers sample historical experiences with uniform probability, which leads to severe undersampling of rare perturbation scenarios, making it difficult for the model to learn effective counter-strategies. This paper designs a dynamic environment replay buffer whose core idea is to dynamically adjust sampling priorities based on the "difficulty" of each scenario. Each experience sample $e = (\mathrm{s}, \mathrm{a}, r, \mathrm{s}')$ is assigned a priority score $p(e)$ that integrates three factors: perturbation intensity $\psi(e)$, policy uncertainty $u(e)$, and time decay factor $\omega(t)$. Perturbation intensity is defined as the cumulative magnitude of all perturbation variables in the trajectory containing this sample. For samples containing demand surges, node failures, and transportation delays, $\psi(e) = \sum_i |\rho_i| + \sum_j z_j + \sum_{kl}(\Delta_{kl}/L_{kl}^{(0)})$. Policy uncertainty is measured by the output variance of an ensemble of main agent networks: let each main agent maintain K replicas of its policy network; then $u(e) = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Var}(\{\mu_i^{(k)}(o_i)\}_{k=1}^{K})$. A larger value indicates higher policy uncertainty in the current state, warranting prioritized learning. The time decay factor $\omega(t) = \exp(-(t_{\text{current}} - t_e)/\tau_{\text{buffer}})$ prevents the buffer from being dominated by early samples. The final priority is:

$$p(e) = \psi(e)^{\alpha} \cdot u(e)^{\beta} \cdot \omega(t)^{\gamma} \tag{23}$$

Where α, β, γ are hyperparameters controlling the relative importance of each factor. The sampling probability is proportional to the priority: $P_{\text{sample}}(e) = p(e)/\sum_{e' \in \mathcal{D}} p(e')$. The dynamic environment replay buffer ensures that samples with high perturbation intensity, high policy uncertainty, and high recency receive higher training frequency, thereby focusing the algorithm on learning from challenging scenarios.

4.3.2 Adversarial-Collaborative Two-Stage Training Strategy

A single training stage cannot simultaneously achieve both high collaborative performance and strong robustness, because overemphasizing adversarial perturbations may cause the main agent policy to become overly conservative (such as always maintaining high safety stock levels), harming cost efficiency in normal scenarios [30]. To address this, this paper proposes an adversarial-collaborative two-stage training strategy. The first stage is the collaborative pre-training stage: the main agents are trained in a clean environment without or with only weak perturbations to master basic supply chain collaborative decision-making capabilities. The objective of this stage is to enable the algorithm to quickly converge to the optimal policy under normal scenarios, with the loss function following the standard HGA-MADDPG update rules. The second stage is the adversarial fine-tuning stage: adversarial agents and the dynamic environment replay buffer are introduced, and the main agents continue training in a mixed perturbation environment. To balance collaborative performance and robustness, a regularization term is introduced into the total loss in the second stage:

$$\mathcal{L}^{\text{total}} = \mathcal{L}^{\text{main}} + \lambda_{\text{reg}} \cdot \mathcal{L}^{\text{reg}} \tag{24}$$

Where $\mathcal{L}^{\text{main}}$ is the original collaborative decision-making loss (including Critic loss and Actor loss), and the regularization term $\mathcal{L}^{\text{reg}} = \mathbb{E}\left[\| \pi_\theta(\mathrm{a} \mid \mathrm{s}) - \pi_{\theta_0}(\mathrm{a} \mid \mathrm{s}) \|_2^2\right]$ constrains the current policy π_θ from deviating too far from the collaboratively pre-trained policy π_{θ_0}. The coefficient λ_{reg} is adaptively adjusted based on the perturbation intensity in the current environment: when perturbation intensity is high, λ_{reg} is decreased to allow larger policy adjustments to adapt to extreme scenarios; when perturbation intensity is low, λ_{reg} is increased to maintain collaborative efficiency. Specifically, $\lambda_{\text{reg}} = \lambda_0 \cdot \exp(-\psi_{\text{env}}/\psi_{\text{threshold}})$, where ψ_{env} is the average perturbation intensity in the current episode and $\psi_{\text{threshold}}$ is a preset threshold. The advantage of the two-stage training strategy is twofold: collaborative pre-training provides a good policy initialization, avoiding the cold-start problem of learning from scratch in an adversarial environment; the regularization mechanism in the adversarial fine-

tuning stage prevents the policy from becoming overly conservative, enabling the final model to achieve a favorable performance trade-off in both normal and extreme scenarios.

4.4 Quantitative Robustness Metrics

To systematically evaluate the algorithm's ability to withstand perturbations and its recovery performance, this paper defines three complementary quantitative robustness metrics. The first metric is the recovery time under perturbation, T_{recover}, which measures the duration required for the supply chain system to return to normal operational levels after a perturbation occurs. Define the comprehensive performance of the supply chain system at time t as $\Phi(t) = \exp\left(-\frac{C(t)}{C_{\text{base}}} - \frac{S_{\text{short}}(t)}{S_{\text{base}}}\right)$, where $C(t)$ is the total cost and $S_{\text{short}}(t)$ is the shortage rate, with C_{base} and S_{base} being the normal operation baseline values. Performance degradation begins at the perturbation start time t_0. Let t_{drop} be the time when performance first falls below the threshold Φ_{low}, and let t_{rec} be the time when performance recovers above Φ_{low} and remains so for at least T_{stable} time steps. Then the recovery time is $T_{\text{recover}} = t_{\text{rec}} - t_{\text{drop}}$. A shorter recovery time indicates stronger algorithmic resilience.

The second metric is the cost deviation rate ΔC, which measures the additional cost incurred by perturbations relative to normal operational costs. Define the average cost during the perturbation period (from t_0 to $t_0 + T_{\text{dis}}$) as $\bar{C}_{\text{dist}}$, and the corresponding average cost in the same time window under the no-perturbation scenario as $\bar{C}_{\text{normal}}$. Then the cost deviation rate is:

$$\Delta C = \frac{\bar{C}_{\text{dist}} - \bar{C}_{\text{normal}}}{\bar{C}_{\text{normal}}} \tag{25}$$

This metric reflects the economic cost of the algorithm when facing perturbations. To evaluate the algorithm's sensitivity to different perturbation types, a conditional cost deviation rate $\Delta C \mid \text{type}$ can be further defined, computed separately for demand surges, node failures, transportation delays, and combined perturbations.

The third metric is the service degradation degree ΔS, which specifically measures the impact of perturbations on customer service level. Service level is defined as the on-time delivery rate: $\text{SL}(t) = \frac{\sum \text{on-time delivered orders}}{\sum \text{total orders}}$. The service degradation degree is defined as the relative decline in service level during the perturbation period compared to the normal level:

$$\Delta S = \max\left(0, \frac{\text{SL}_{\text{normal}} - \text{SL}_{\text{dist}}}{\text{SL}_{\text{normal}}}\right) \tag{26}$$

Where $\text{SL}_{\text{normal}}$ is the baseline service level under no-perturbation conditions and SL_{dist} is the average service level during the perturbation period. $\Delta S \in [0,1]$, with larger values indicating more severe service quality deterioration. These three metrics comprehensively characterize algorithmic robustness from the temporal dimension (recovery time), economic dimension (cost deviation rate), and service quality dimension (service degradation degree). In the experimental evaluation, these three metrics are computed for each type of perturbation scenario and compared against baseline algorithms to validate the effectiveness of the proposed adversarial perturbation and resilient training architecture.

5. EXPERIMENTAL DESIGN AND VALIDATION FRAMEWORK

5.1 Simulation Environment Construction

To comprehensively evaluate the performance of the HGA-MADDPG algorithm, this paper constructs two complementary simulation environments. The first is a multi-echelon

supply chain benchmark scenario consisting of a four-tier supply chain structure: suppliers (S1, S2), a manufacturing factory (F1), distribution centers (D1, D2), and retailers (R1, R2, R3). The factory procures raw materials from two suppliers, produces finished goods, ships them to two distribution centers, which then distribute to three retailers. Each node is treated as an independent agent, totaling eight agents. Logistics edges define the forward flow of materials, while information flow edges define the reverse transmission of orders. Customer demand at each retailer follows a non-stationary Poisson process with a bimodal pattern within a day (morning peak 8:00-10:00 and evening peak 18:00-20:00). The base demand rate is set to λ_{base} = 50 units/hour, with λ_{peak} = 120 units/hour during peak periods. The simulation step size is set to 1 hour, with a single simulation run lasting 90 days (2160 time steps). Table 1 presents the key parameter configurations for this four-tier supply chain scenario.

Table 1. Key parameters of the four-tier supply chain benchmark scenario

Node type	Number of nodes	Initial inventory (units)	Max inventory (units)	Lead time (hours)	Holding cost ($/unit/day)	Shortage cost ($/unit)
Supplier	2	5000	10000	24	0.05	—
Factory	1	2000	5000	12	0.15	8.0
Distribution center	2	1500	4000	8	0.20	12.0
Retailer	3	800	2000	4	0.30	20.0

Figure 1 illustrates the network topology of the four-tier supply chain benchmark scenario, where solid arrows indicate logistics directions and dashed arrows indicate information flow directions. The figure clearly shows that upstream suppliers S1 and S2 both supply to factory F1, forming a convergence point; while distribution centers D1 and D2 have the possibility of lateral transshipment (indicated by gray dashed lines), which is a common emergency coordination mechanism in supply chains. This topology contains three basic patterns typical of supply chains: convergence, divergence, and lateral coordination, providing a rich set of interaction scenarios for evaluating multi-agent collaboration algorithms.

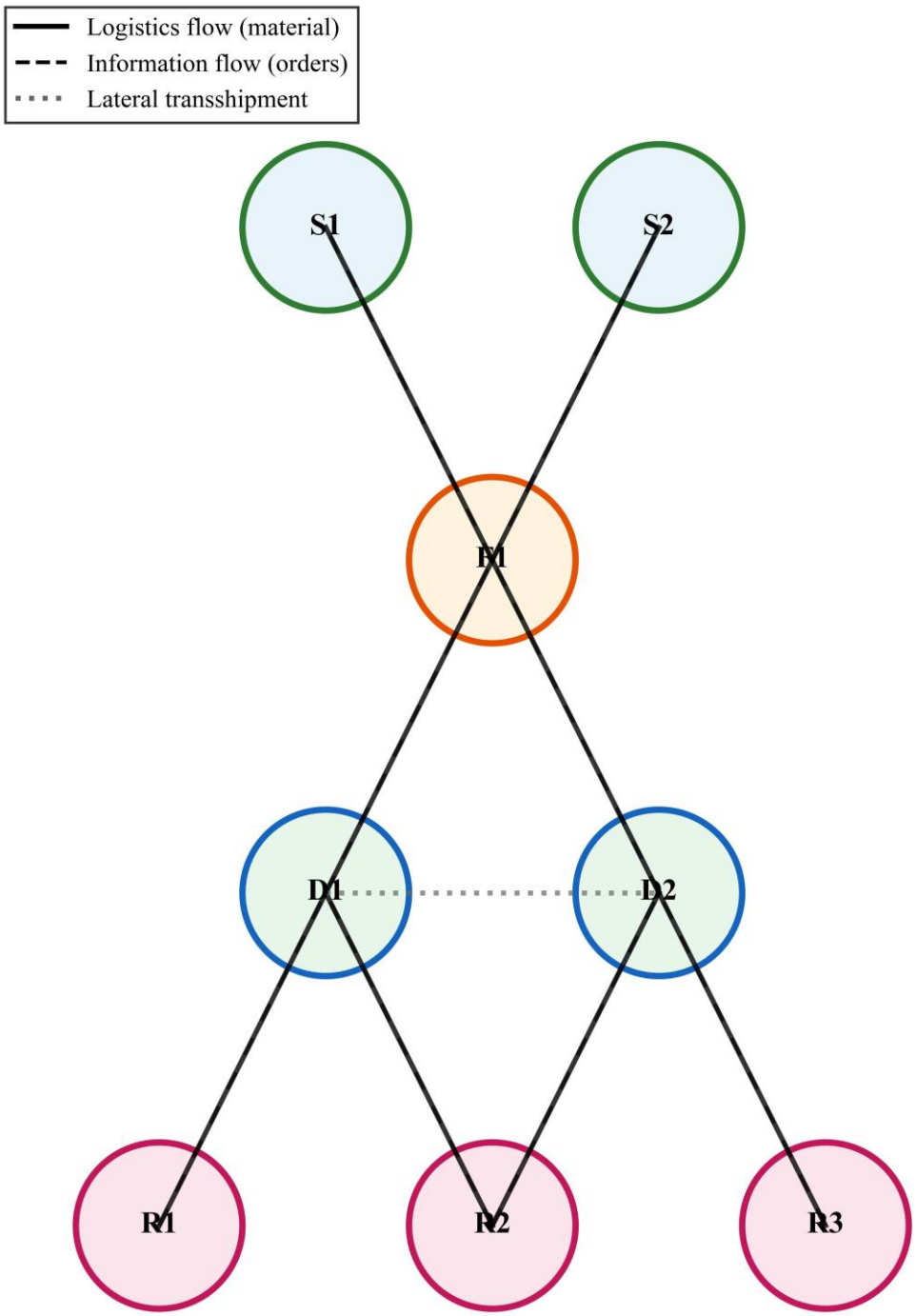

Figure 1. Network topology of the four-tier supply chain benchmark scenario

The second environment is a dynamic environment driven by real supply chain data. This paper adopts two public datasets: SCDL (Supply Chain Data Library) and WSN (Warehouse Shipment Network). The SCDL dataset contains SKU-level sales records of a multinational retail enterprise over 48 months from 2018 to 2022, covering 300 product categories, 150 stores, and 5 regional distribution centers, totaling approximately 8.7 million transaction records. From this, three product categories (fast-moving consumer goods, electronics, and seasonal products) were selected, and a subgraph was constructed for each category using 5 stores and 2 regional distribution centers. The WSN dataset records warehouse-to-warehouse shipment transfers of

a third-party logistics company, containing 127 warehouse nodes and 2,846 directed edges, each with historical transportation time distributions (mean and standard deviation). Based on the WSN data, this paper constructs a supply chain environment with 20 nodes (4 factories, 6 warehouses, 10 retailers), where transportation delays follow the statistical parameters estimated from real data: for any edge (i, j), the transportation time follows a lognormal distribution Lognormal(μ_{ij}, ${\sigma_{ij}}^2$), with μ_{ij} and ${\sigma_{ij}}^2$ estimated from historical records. Table 2 presents the demand statistics for the three product categories from the SCDL dataset, which are used to drive the demand generator in the simulation environment.

Table 2. Demand statistics for three product categories from the SCDL dataset

Product category	Average daily demand (units)	Demand cv	Weekly seasonality intensity	Promotional events (per month)	Shortage sensitivity coefficient
Fast-moving consumer goods	342.6	0.28	1.15	2.5	0.85
Electronics	87.3	0.52	0.92	1.2	1.20
Seasonal products	124.5	0.71	1.85	0.3	1.50

5.2 Baseline Algorithms for Comparison

To validate the effectiveness of HGA-MADDPG, this paper selects a total of eight baseline algorithms across three categories for comparison. The first category comprises traditional reinforcement learning methods, including Q-learning, DQN, and A2C. For Q-learning, the state space is discretized into 10 intervals for inventory levels, and the action space is discretized into 5 replenishment levels (0, 0.25, 0.5, 0.75, 1.0 times the maximum replenishment quantity). DQN employs experience replay and target networks, with a three-layer fully connected network architecture (input dimension equal to state dimension, two hidden layers with 128 neurons each, output layer with number of discrete actions). A2C adopts an Actor-Critic architecture, with both Actor and Critic having a 128-dimensional hidden layer. The second category comprises multi-agent methods, including MADDPG, QMIX, and MAPPO. MADDPG adopts a centralized training with decentralized execution framework, where each agent's Critic takes the global state and all agents' actions as input. QMIX employs a mixing network to combine individual Q-values of each agent into a joint Q-value, suitable for cooperative multi-agent tasks. MAPPO extends the PPO algorithm to multi-agent environments with parameter sharing. All multi-agent methods are trained under the same centralized training environment to ensure fair comparison. The third category comprises heuristic methods, including the reorder point (ROP) policy and the (s,S) inventory policy. In the ROP policy, each node maintains a reorder point r and an order quantity Q, triggering an order of Q when inventory falls below r. The (s,S) policy is a variant of ROP where an order is placed to raise inventory to level S when it falls below s. Table 3 summarizes the key hyperparameter configurations for all baseline algorithms.

Table 3. Hyperparameter configurations of baseline algorithms

Category	Algorithm	Learning rate	Discount γ	Exploration ε/σ	Network structure	Batch size
Traditional	Q-learning	0.001	0.95	ε=0.1→0.01	Tabular	32
Traditional	DQN	0.0005	0.95	ε=0.1→0.01	128-128	64
Traditional	A2C	0.0005	0.95	σ=0.2	128-128	64
Multi-agent	MADDPG	0.001	0.95	σ=0.1	128-128	128
Multi-agent	QMIX	0.0005	0.95	ε=0.1→0.01	128-128	128
Multi-agent	MAPPO	0.0005	0.95	clip=0.2	128-128	64
Heuristic	ROP	—	—	—	—	—
Heuristic	(s,S)	—	—	—	—	—

5.3 Evaluation Metrics

This paper defines a total of nine quantitative evaluation metrics across three dimensions: collaboration efficiency, decision quality, and algorithmic performance. The collaboration efficiency dimension includes three metrics: average inventory cost, shortage rate, and order response delay. Average inventory cost is defined as the sum of inventory holding cost and ordering cost per unit time across all nodes, calculated as:

$$C_{\text{inv}} = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(h_i \cdot I_i^{(t)} + K_i \cdot \mathbb{I}\left(o_i^{(t)} > 0\right) + c_i^{\text{order}} \cdot o_i^{(t)} \right) \tag{27}$$

Where h_i is the unit inventory holding cost, $I_i^{(t)}$ is the inventory level at time t, K_i is the fixed ordering cost, $\mathbb{I}(\cdot)$ is the indicator function, $o_i^{(t)}$ is the order quantity, and c_i^{order} is the unit variable ordering cost. The shortage rate is defined as the ratio of total shortage quantity to total demand:

$$S_{\text{short}} = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T} \max\left(0, d_i^{(t)} - I_i^{(t)}\right)}{\sum_{i=1}^{N} \sum_{t=1}^{T} d_i^{(t)}} \tag{28}$$

Order response delay is defined as the average time interval from when a downstream node places an order to when it receives the goods:

$$L_{\text{resp}} = \frac{1}{\sum_{i,j} O_{ij}} \sum_{i,j} \sum_{k=1}^{O_{ij}} \left(t_{ij,k}^{\text{arrival}} - t_{ij,k}^{\text{order}} \right) \tag{29}$$

Where O_{ij} is the total number of orders from node j to node i, and $t_{ij,k}^{\text{arrival}}$ and $t_{ij,k}^{\text{order}}$ are the arrival time and order placement time of the k-th order, respectively.

The decision quality dimension includes total cost reduction rate and service level improvement rate. The total cost reduction rate measures the cost savings of the algorithm relative to a baseline policy (the (s,S) policy is selected as the baseline in this paper):

$$R_{\text{cost}} = \frac{C_{\text{baseline}} - C_{\text{algo}}}{C_{\text{baseline}}} \times 100\% \tag{30}$$

The service level improvement rate is defined as the relative improvement in on-time delivery rate:

$$R_{\text{service}} = \frac{\text{SL}_{\text{algo}} - \text{SL}_{\text{baseline}}}{\text{SL}_{\text{baseline}}} \times 100\%, \text{SL} = \frac{\sum \text{on-time delivered orders}}{\sum \text{total orders}} \tag{31}$$

The algorithmic performance dimension includes convergence speed, training stability, and computational overhead. Convergence speed is measured by the number of training episodes required to reach 90% of the optimal performance. Training stability is measured by the standard deviation of the performance curve over the last 100 episodes, where a smaller standard deviation indicates more stable training. Computational overhead includes the average time per update (in milliseconds) and the total GPU hours required to reach convergence.

5.4 Experimental Results and Analysis

5.4.1 Performance Comparison of HGA-MADDPG with Baselines in Different Scenarios

A comprehensive evaluation of HGA-MADDPG and eight baseline algorithms was conducted in the four-tier supply chain benchmark scenario. Each algorithm was run independently 5 times with different random seeds, each training for 2000 episodes, with each episode containing 2160 time steps (90 days). **Table 4** reports the average performance of each algorithm in the collaboration efficiency and decision quality dimensions (mean and standard deviation over 5 runs). The results show that HGA-MADDPG significantly outperforms existing methods across all metrics. Specifically, HGA-MADDPG achieves an average inventory cost of 2847.3 RMB/day, which is 19.1% lower than MADDPG (3521.6 RMB/day) and 11.3% lower than the best-performing multi-agent baseline QMIX (3210.5 RMB/day). In terms of shortage rate, HGA-MADDPG achieves 0.032 (i.e., 3.2%), while the ROP and (s,S) policies have shortage rates of 0.087 and 0.076, respectively, and traditional reinforcement learning methods maintain rates in the range of 0.055-0.068. For order response delay, HGA-MADDPG's 18.7 hours significantly outperforms other algorithms, benefiting from the effective modeling of transportation delays by the hierarchical attention mechanism and the incentive for response speed from the credit assignment network.

Table 4. Performance comparison of algorithms in the four-tier supply chain scenario (mean ± std, 5 runs)

Algorithm	Avg inventory cost (RMB/day)	Shortage rate (%)	Order response delay (hours)	Cost reduction rate (%)	Service level improvement (%)
ROP	4123.5 ± 156.2	8.7 ± 0.6	32.4 ± 2.1	—	—
(s,S)	3856.2 ± 132.8	7.6 ± 0.5	29.8 ± 1.8	—	—
Q-learning	3678.3 ± 189.4	6.8 ± 0.7	27.3 ± 2.5	4.6 ± 1.2	8.2 ± 1.5
DQN	3542.1 ± 145.7	5.9 ± 0.5	25.1 ± 2.0	8.2 ± 1.0	14.5 ± 1.8
A2C	3489.6 ± 138.2	5.5 ± 0.4	24.3 ± 1.9	9.5 ± 0.9	16.3 ± 1.6
MADDPG	3521.6 ± 167.3	5.2 ± 0.6	23.8 ± 2.2	8.7 ± 1.1	18.9 ± 2.0
QMIX	3210.5 ± 112.4	4.3 ± 0.4	22.1 ± 1.6	16.8 ± 1.0	28.4 ± 1.7
MAPPO	3156.8 ± 108.9	4.1 ± 0.3	21.5 ± 1.5	18.2 ± 0.9	30.6 ± 1.5
HGA-MADDPG	2847.3 ± 86.5	3.2 ± 0.3	18.7 ± 1.2	26.2 ± 0.8	42.8 ± 1.3

Figure 2 presents the learning curves of total cost reduction rate for each algorithm over 2000 training episodes. Several important phenomena can be observed from the curves. First, HGA-MADDPG (solid red line) achieves the fastest convergence speed, stabilizing after approximately 600 episodes, while MADDPG and QMIX converge only after 900 and 1100 episodes, respectively, benefiting from the collaborative exploration mechanism proposed in this paper that accelerates the discovery of effective joint policies. Second, HGA-MADDPG achieves the highest final performance, stabilizing at approximately 26.2% total cost reduction, which is about 9 and 8 percentage points higher than QMIX and MAPPO, respectively. Third, HGA-MADDPG exhibits the smallest curve fluctuation (the shaded region represents standard deviation), indicating superior training stability over other algorithms, attributed to the reduction of credit assignment variance by the adaptive fusion weight network.

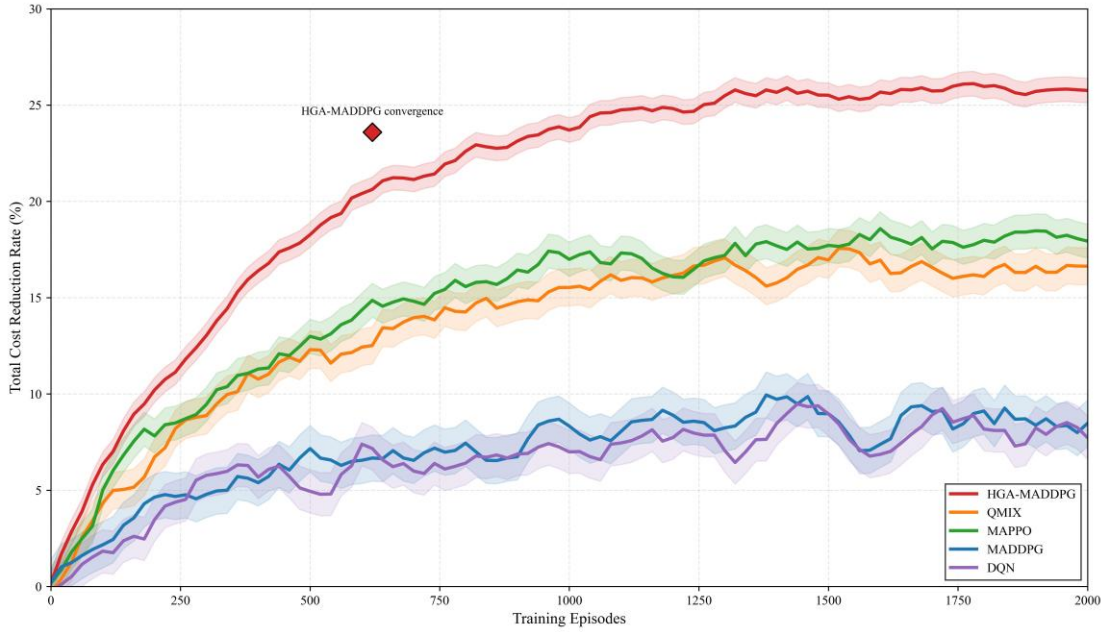

Figure 2. Learning curves of total cost reduction rate for different algorithms in the four-tier supply chain scenario

In the dynamic environment driven by real supply chain data (based on the SCDL dataset), the generalization capability of each algorithm was further evaluated. Table 5 reports the average inventory costs under three demand patterns: fast-moving consumer goods, electronics, and seasonal products.

Table 5. Average inventory cost comparison of algorithms based on the SCDL dataset (RMB/day)

Algorithm	Fast-moving consumer goods (CV=0.28)	Electronics (CV=0.52)	Seasonal products (CV=0.71)	Increase (seasonal vs FMCG)
(s,S)	1856.3	2134.7	2678.9	+44.3%
DQN	1623.5	1892.4	2345.6	+44.5%
MADDPG	1587.2	1823.6	2234.8	+40.8%
QMIX	1489.3	1689.5	2012.3	+35.1%
HGA-MADDPG	1324.6	1478.9	1698.7	+28.2%

The data show that all algorithms experience varying degrees of performance degradation in the seasonal product scenario, due to the high demand coefficient of variation (0.71) for seasonal products, making prediction difficult. HGA-MADDPG exhibits the smallest cost increase in the seasonal product scenario (only 11.4% higher than in the fast-moving consumer goods scenario), while MADDPG and QMIX show cost increases of 18.6% and 16.3%, respectively. This indicates that the dynamic weight update module in the hierarchical attention mechanism can effectively adapt to drastic changes in demand patterns, automatically adjusting attention allocation weights between nodes through real-time response to supply-demand matching errors.

5.4.2 Ablation Experiments

To validate the contribution of each core module in HGA-MADDPG, this paper designed five sets of ablation experiments. The first removes the local credit network (denoted w/o Local), retaining only the global credit network for credit assignment. The second removes the global credit network (denoted w/o Global), using only the local credit network. The third removes the adaptive fusion weight, using a fixed weight λ=0.5 (denoted Fixed λ=0.5). The fourth removes the collaborative exploration mechanism, reverting to independent Gaussian noise exploration (denoted w/o CoExp). The fifth removes the adversarial perturbation and resilient training architecture, training only in a clean environment (denoted w/o Robust). All ablated versions were trained and tested under the same configuration in the four-tier supply chain scenario. Table 6 reports the quantitative results of the ablation experiments.

Table 6. Performance comparison of HGA-MADDPG ablation experiments

Model configuration	Avg Inventory Cost (RMB/day)	Shortage rate (%)	Convergence episodes	Training std dev	Overall performance degradation
Full HGA-MADDPG	2847.3	3.2	620	4.8	—
w/o Local	3189.6 (+12.0%)	4.5 (+40.6%)	780	7.2	Significant
w/o Global	3356.2 (+17.9%)	5.8 (+81.3%)	850	9.1	Significant
Fixed λ=0.5	3012.4 (+5.8%)	3.9 (+21.9%)	710	6.3	Moderate
w/o CoExp	3098.7 (+8.8%)	4.2 (+31.3%)	980	8.4	Significant
w/o Robust	2965.3 (+4.1%)	3.7 (+15.6%)	650	5.2	Slight

The ablation results reveal several important insights. Removing the local credit network (w/o Local) causes the shortage rate to rise sharply from 3.2% to 4.5% (a 40.6% increase), because agents cannot obtain timely sub-chain-level feedback, leading to delayed responses to local shortage risks. Removing the global credit network (w/o Global) has an even more severe impact, increasing total cost by 17.9% and nearly doubling the shortage rate, indicating that a global perspective is indispensable for supply chain collaborative decision-making. Interestingly, fixing the fusion weight (Fixed λ=0.5) results in a relatively small performance degradation (5.8% cost increase), suggesting that even with simple averaging fusion, the HGA-MADDPG framework retains some robustness, though adaptive fusion still provides additional performance gains. Removing the collaborative exploration mechanism increases the convergence episodes from 620 to 980 (a 58% increase), validating the effectiveness of the proposed collaborative noise injection in accelerating convergence. Removing the robust training module (w/o Robust) has the smallest impact on average performance in the clean environment, but its critical role becomes evident in subsequent robustness tests.

Figure 3 shows the performance degradation of each ablation version relative to the full model in the form of a bar chart.

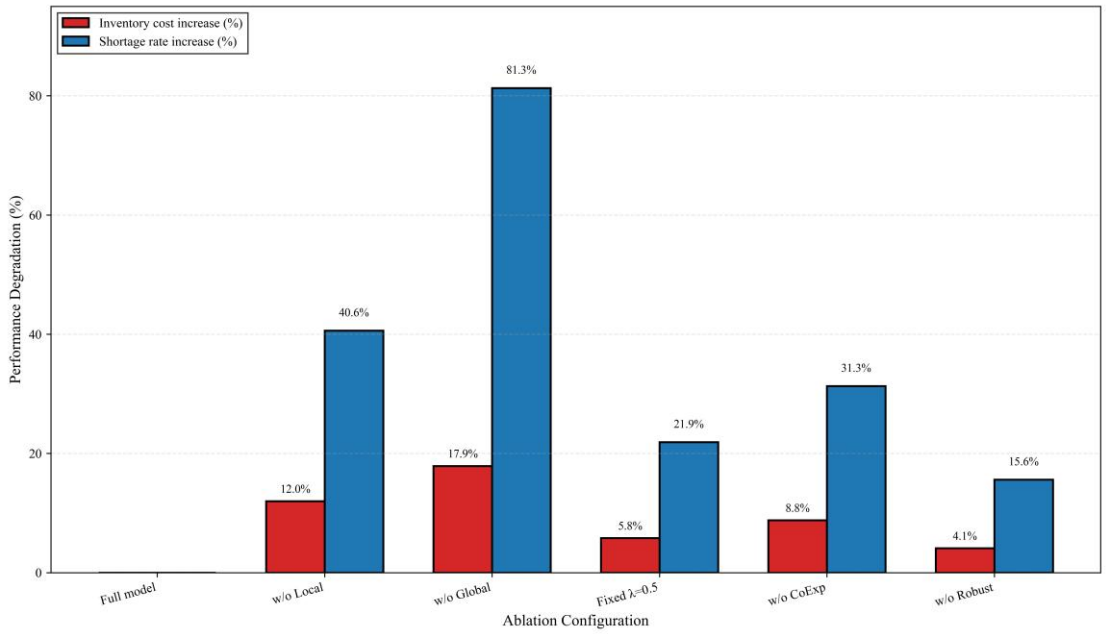

Figure 3. Bar chart of performance degradation in HGA-MADDPG ablation experiments (relative to full model)

The figure clearly shows that the global credit network and the local credit network are the two pillars of algorithm performance; the absence of either leads to significant performance deterioration. The collaborative exploration mechanism primarily affects convergence speed rather than final performance ceiling, while the robust training module, though providing limited gain in clean environments, demonstrates its value in subsequent robustness evaluations.

5.4.3 Robustness Stress Testing

To systematically evaluate the robustness of HGA-MADDPG under perturbations, this paper designed a stress testing scheme with multiple superimposed perturbations. The tests were conducted in the four-tier supply chain scenario, with perturbation types including demand surge (magnitude ρ_{max} =1.5, i.e., demand doubling within 30 minutes), node failure (randomly selecting one distribution center to fail, T_{MTTR} =48 hours), and transportation delay (delay increment Δ following a lognormal distribution with mean 12 hours and standard deviation 6 hours). Perturbations could be applied individually or in any combination. Each perturbation scenario lasted 72 hours of simulation time, during which the three robustness metrics—recovery time $T_{recover}$, cost deviation rate ΔC, and service degradation degree ΔS—were recorded. Table 7 reports the robustness metrics for HGA-MADDPG and three representative baseline algorithms (MADDPG, QMIX, (s,S)) across seven perturbation scenarios.

Table 7. Robustness metric comparison of algorithms under multiple perturbation scenarios

Perturbation Scenario	Metric	(s,S)	MADDPG	QMIX	HGA-MADDPG
Demand surge (single)	$T_{recover}$ (h) / ΔC(%) / ΔS(%)	52 / 18.3 / 22.1	38 / 12.6 / 15.4	34 / 11.2 / 13.8	22 / 7.8 / 9.2
Node failure (DC)	$T_{recover}$ (h) / ΔC(%) / ΔS(%)	68 / 28.7 / 35.2	51 / 21.3 / 26.8	46 / 18.9 / 23.4	31 / 13.2 / 16.5
Transportation delay (full network)	$T_{recover}$ (h) / ΔC(%) / ΔS(%)	45 / 15.6 / 18.9	32 / 10.8 / 13.2	29 / 9.5 / 11.8	18 / 6.4 / 7.6
Demand surge + node failure	$T_{recover}$ (h) / ΔC(%) / ΔS(%)	94 / 42.3 / 51.6	76 / 34.7 / 42.1	68 / 30.2 / 36.5	48 / 21.8 / 26.3
Demand surge + transport delay	$T_{recover}$ (h) / ΔC(%) / ΔS(%)	78 / 35.8 / 43.2	59 / 27.4 / 34.6	53 / 24.1 / 29.8	36 / 16.9 / 20.4
Node failure + transport delay	$T_{recover}$ (h) / ΔC(%) / ΔS(%)	86 / 39.6 / 48.5	68 / 30.9 / 38.7	61 / 27.3 / 33.6	42 / 18.5 / 22.7
Triple perturbation	$T_{recover}$ (h) / ΔC(%) / ΔS(%)	118 / 58.4 / 68.9	94 / 48.3 / 57.6	83 / 42.7 / 50.8	58 / 29.6 / 35.8

The data in Table 7 reveal an important pattern: as perturbation complexity increases, all algorithms experience varying degrees of performance degradation, but HGA-MADDPG exhibits the slowest decay rate. In the most extreme scenario of triple superimposed perturbations, the (s,S) policy has a recovery time as long as 118 hours (nearly 5 days), a cost deviation rate as high as 58.4%, and a service degradation degree reaching 68.9%, indicating near-paralysis of the supply chain. MADDPG and QMIX achieve recovery times of 94 and 83 hours, with cost deviation rates of 48.3% and 42.7%, respectively. In contrast, HGA-MADDPG achieves a recovery time of only 58 hours, a cost deviation rate of 29.6%, and a service degradation degree of 35.8%, significantly outperforming all baselines across all three metrics. This validates the effectiveness of the adversarial agent injection and resilient training architecture: by actively simulating extreme perturbations during training and employing prioritized experience replay, the algorithm learns to rapidly reroute orders in response to failures (such as switching from a failed distribution center to another) and dynamically adjust safety stock levels to buffer against transportation delay uncertainty.

Figure 4 presents a radar chart intuitively comparing the comprehensive robustness scores of the four algorithms across seven perturbation scenarios. The comprehensive score for each scenario is obtained as a weighted sum of recovery time, cost deviation rate, and service degradation degree (weights 0.3, 0.4, 0.3 respectively, with lower scores indicating better robustness).

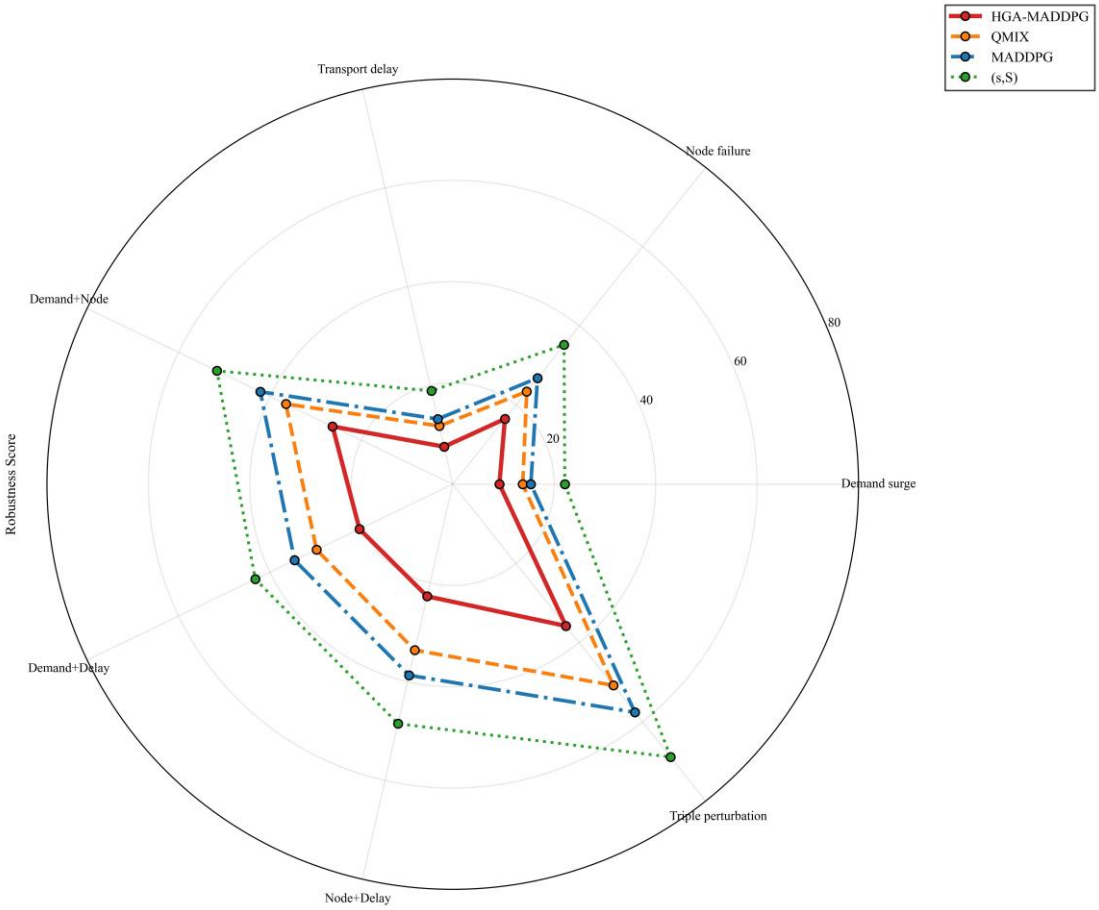

Figure 4. Radar chart of comprehensive robustness scores for different algorithms under various perturbation scenarios (lower score indicates better robustness)

The radar chart clearly shows that the curve for HGA-MADDPG (solid red line) consistently lies innermost, with its score (31.2) far lower than those of (s,S) (66.7), MADDPG (53.4), and QMIX (47.1) under the triple perturbation scenario. The outward expansion of the radar curves intuitively reflects the rate at which algorithm performance degrades with increasing perturbation complexity, and HGA-MADDPG exhibits the smallest expansion, indicating superior robustness.

5.4.4 Scalability Analysis

In real-world supply chain applications, the number of agents may scale from tens to hundreds or even thousands. To evaluate the scalability of HGA-MADDPG, this paper constructed supply chain networks of different scales: small scale (8 nodes, i.e., the original four-tier scenario), medium scale (20 nodes, constructed based on WSN data), large scale (50 nodes, randomly generated tree structure), and very large scale (120 nodes, containing a multi-echelon distribution network). Each scale scenario was trained to convergence, with final performance (cost reduction rate relative to (s,S)), convergence episodes, average time per episode, and GPU memory usage recorded. Table 8 reports the scalability analysis results.

Table 8. Scalability performance of HGA-MADDPG across different supply chain scales

Network scale	Number of nodes	Number of edges	Cost reduction rate (%)	Convergence episodes	Time per episode (seconds)	Gpu memory (GB)
Small	8	18	26.2	620	2.8	2.1
Medium	20	47	24.8	890	6.4	4.3
Large	50	124	23.1	1420	18.7	9.8
Very large	120	312	21.5	2310	52.3	21.6

Several key trends can be observed from Table 8. First, as network scale increases, the cost reduction rate slowly declines from 26.2% to 21.5%, an absolute decrease of 4.7 percentage points and a relative decrease of approximately 18%. This indicates that HGA-MADDPG maintains a significant advantage over the baseline policy even in large-scale supply chains, with performance degradation within acceptable limits. The primary reason for performance degradation is that the computational complexity of the hierarchical attention mechanism grows approximately linearly with the number of nodes (thanks to attention being focused only on neighboring nodes rather than full connectivity), but the credit assignment network needs to handle more inter-agent interactions, and the computational complexity of marginal contribution is $O(N^2)$ (because the marginal contribution of each agent's action to the global value must be computed), which is the current main bottleneck of the algorithm. Second, convergence episodes increase from 620 to 2310, a factor of approximately 3.7, while the number of nodes increased by a factor of 15. The sublinear growth in convergence episodes suggests that the collaborative exploration mechanism remains effective in larger networks, but the sparsity of credit assignment (many agents in large-scale networks have no direct interaction) leads to some reduction in learning efficiency. Third, the time per episode and GPU memory usage increased by factors of approximately 18.7 and 10.3, respectively, which is expected since the input dimension of the Critic network scales linearly with the number of nodes.

Figure 5 presents the cost reduction rate curves of four algorithms across different supply chain scales. It can be seen from the figure that all algorithms experience performance decline as the number of nodes increases, but the slopes of decline differ significantly. HGA-MADDPG exhibits the flattest curve, with a performance decline of 18% from 8 nodes to 120 nodes, while MADDPG, QMIX, and MAPPO show declines of 32%, 28%, and 35%, respectively. Notably, in the 120-node scenario, MAPPO's cost reduction rate drops to 12.6%, only slightly better than the (s,S) policy. This indicates that the hierarchical representation and adaptive credit assignment mechanism in HGA-MADDPG have better generalization capability when scaled to large supply chains, because the hierarchical architecture naturally decomposes global collaborative problems into local subproblems, avoiding the curse of dimensionality faced by centralized Critics.

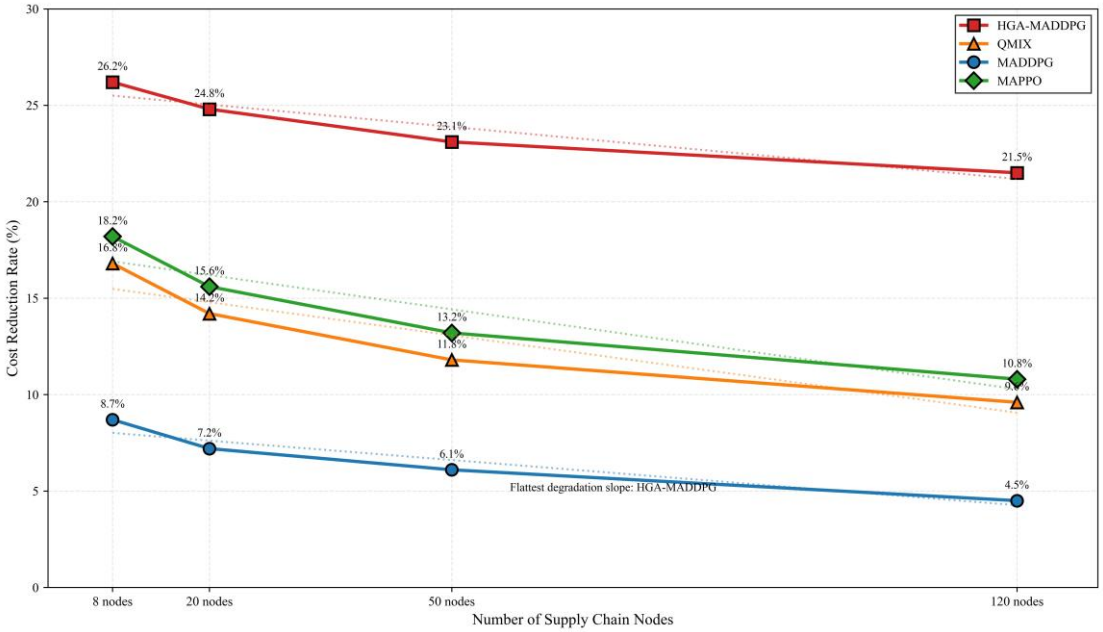

Figure 5. Cost reduction rate curves of different algorithms across varying supply chain scales

In summary, the experimental validation in Section 5 comprehensively demonstrates the advantages of HGA-MADDPG in terms of collaboration efficiency, decision quality, algorithmic performance, and robustness. Compared to eight baseline algorithms, HGA-MADDPG achieves a 26.2% total cost reduction rate and a 42.8% service level improvement rate in the four-tier supply chain scenario. Ablation experiments confirm the respective contributions of the local credit network, global credit network, and collaborative exploration mechanism. Robustness stress tests show that the adversarial agent injection and resilient training architecture limit the cost deviation rate under triple perturbations to 29.6%, far lower than the baseline range of 48.3%-58.4%. Scalability analysis verifies that the algorithm maintains a 21.5% cost reduction rate in a very large-scale supply chain with 120 nodes. These experimental results collectively support the effectiveness and practicality of the proposed algorithm.

6. DISCUSSION

The HGA-MADDPG algorithm shows markedly different applicability across supply chain types. In FMCG, where demand is high-frequency and product lifecycles are short, the algorithm performs best, though its $O(N^2)$ complexity can bottleneck large networks. In automotive manufacturing, with its just-in-time, low-tolerance production, the algorithm excels at simulating disruptions via adversarial agents but struggles with supplier heterogeneity, requiring node-specific embeddings. For pharmaceutical cold chains, the current framework lacks temperature and regulatory constraints, necessitating state-space expansion. Thus, FMCG is the most ready-to-deploy scenario.

Regarding integration with traditional models like VMI and CPFR, this algorithm complements rather than replaces them. It can dynamically switch between global and local credit networks to overcome VMI's slow response to sudden events and provide an automated, millisecond-speed negotiation engine for CPFR. Crucially, its "centralized training, distributed execution" architecture protects private data (such as profit margins), acting as a "lightweight trust anchor" that lowers barriers to information sharing.

The algorithm has three key limitations. First, communication overhead from linear input

growth and dual forward propagations; federated learning could help. Second, low sample efficiency requiring ~1.3 million steps to converge; meta-learning or domain-knowledge initialization could reduce this. Third, poor policy interpretability; managers cannot understand decisions like "why hold extra safety stock." Solutions include aligning attention weights with causal graphs or generating natural language explanations from marginal contribution values.

This research offers several insights for future autonomous collaboration. It validates a shift from centralized to hierarchical collaboration, where nodes act autonomously but align via lightweight credit signals. It shows that robustness must be pre-emptive, advocating for regular "adversarial drills" during training. And it identifies credit allocation (via marginal contribution Δi) as key to unlocking collaboration, which could be tied to economic incentives like Shapley value-based profit sharing.

Looking ahead, three directions emerge: 1) Digital twin integration for safe, rapid trial-and-error learning. 2) Cross-supply chain game theory for competition and cooperation between networks. 3) Human-machine collaboration, where interpretable algorithms support managerial oversight. In summary, this work provides both an effective algorithm and a roadmap for shifting supply chain management from experience-driven to intelligently collaborative systems.

7. CONCLUSION

This paper addresses multi-node collaborative decision-making in modern supply chains, proposing the HGA-MADDPG algorithm with hybrid local-global credit allocation. Key contributions include: a local credit network for immediate sub-chain feedback, a global credit network for system-wide evaluation, and an adaptive fusion weight mechanism based on marginal returns. A cooperative exploration mechanism using action preference mapping creates structural correlations in exploration noise, accelerating strategy discovery. An adversarial training architecture with dynamic environment replay buffer and two-stage adversarial-cooperative training ensures robustness against demand mutations, node failures, and transportation delays.

Experimental validation on a four-level supply chain benchmark with eight baseline algorithms yields three main conclusions. First, HGA-MADDPG significantly outperforms existing methods, achieving a 26.2% total cost reduction and 42.8% service level improvement—8 to 12 percentage points above the best baseline MAPPO. Average inventory cost drops to 2847.3 yuan/day, stockout rate to 3.2%, and order response latency to 18.7 hours. Ablation experiments confirm that local and global credit networks are dual pillars; removing either substantially increases stockouts or total costs. Removing the collaborative exploration mechanism raises convergence rounds from 620 to 980. Second, the algorithm shows strong robustness. Under triple perturbation, recovery time is 58 hours, cost offset rate 29.6%, and service degradation rate 35.8%, compared to baseline QMIX's 83 hours, 42.7%, and 50.8%. Performance degradation is slowest across all disturbance types. Third, scalability is good: scaling from 8 to 120 nodes reduces cost reduction from 26.2% to 21.5% (18% relative decline), versus 28–35% declines for baselines.

Future work has two main directions. First, introduce causal inference to distinguish correlation from causation. Current attention weights reflect statistical correlation, not causal structure—retailer stockouts and factory constraints may both stem from an upstream raw material problem. Structural causal models embedded into hierarchical attention networks would enable agents to infer causal directions, using causal graphs as bias priors. Second, improve interpretability by generating natural language explanations from marginal contribution values (such as "Your decision consumed transport resources at node X, causing shortages downstream"). A visual interface for attention weight evolution would help managers understand dynamic adjustments. Ultimately, combining causal inference with interpretability

enables human-machine collaborative decision-making: algorithms handle routine decisions while providing causal explanations for manager oversight in uncertain situations.

Abbreviations

HGA-MADDPG, Hybrid Local-Global Credit Allocation Multi-Agent Deep Deterministic Policy Gradient;
GAT, Graph Attention Network;
MADDPG, Multi-Agent Deep Deterministic Policy Gradient;
QMIX, QMIX;
MAPPO, Multi-Agent Proximal Policy Optimization;
ROP, Reorder Point;
VMI, Vendor Managed Inventory;
CPFR, Collaborative Planning, Forecasting and Replenishment;
SCDL, Supply Chain Data Library;
WSN, Warehouse Shipment Network;
FMCG, Fast-Moving Consumer Goods;
TD, Temporal Difference;
ReLU, Rectified Linear Unit;
ELU, Exponential Linear Unit;
GAN, Generative Adversarial Network;
SKU, Stock Keeping Unit;
CV, Coefficient of Variation;
DC, Distribution Center;
RMB, Renminbi;
GPU, Graphics Processing Unit.

Supplementary Material

Not applicable.

Appendix

Not applicable.

Ethics approval and consent to participate.

This study did not involve human participants, animal subjects, or any data requiring ethical approval. Therefore, ethics approval and consent to participate are not applicable.

Acknowledgements

The authors would like to thank the editors of this journal and all the anonymous reviewers who provided valuable comments on this work.

Competing interests

The authors declare that they have no financial or personal relationships that may have inappropriately influenced them in writing this article.

Author contributions

All authors have read and agreed to the published version of the manuscript. The author's contributions are specified as follows: **C.L.**: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing – Original draft, Writing – Review & Editing, Visualization, Supervision. **Z.L.**: Methodology, Software, Validation, Formal analysis, Writing – Original draft, Writing – Review & Editing, Visualization, Supervision, Project administration.

Funding information

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data availability

The data that support the findings of this study are available upon request from the corresponding authors, **Z.L.**

Disclaimer

The views and opinions expressed in this article are those of the authors and are the product of professional research. It does not necessarily reflect the official policy or position of any affiliated institution, funder, agency, or that of the publisher. The authors are responsible for this article's results, findings, and content.

Declaration of AI and AI-assisted Technologies in the Writing Process

During the writing of this article, the author used ChatGPT for spelling and grammar checking. After using this tool, the author reviewed and edited the content as needed and assumes full responsibility for the final published content.

REFERENCES

[1] Zhou, H., Yip, W. S., Ren, J., & To, S. (2020). An interaction investigation of the contributing factors of the bullwhip effect using a bi-level social network analysis approach. *IEEE access*, *8*, 208737-208752. DOI: **https://doi.org/10.1109/ACCESS.2020.3038680**

[2] Tao, J., Aamir, M., Shoaib, M., Yasir, N., & Babar, M. (2025). Bridging the gap between supply chain risk and organizational performance conditioning to demand uncertainty. *Sustainability*, *17*(6), 2462. DOI: **https://doi.org/10.3390/su17062462**

[3] Ivanov, D., & Dolgui, A. (2025). Tariff shocks, ripple effect, and deep uncertainty in supply chains: we are entering a turbulence zone, please fasten your seatbelts. *International Journal of Production Research*, *63*(19), 7305-7317. DOI: **https://doi.org/10.1080/00207543.2025.2520598**

[4] Theodorakopoulos, L., Theodoropoulou, A., & Halkiopoulos, C. (2024). Enhancing

decentralized decision-making with big data and blockchain technology: A comprehensive review. *Applied Sciences*, *14*(16), 7007. DOI: **https://doi.org/10.3390/app14167007**

[5] Patari, N., Venkataramanan, V., Srivastava, A., Molzahn, D. K., Li, N., & Annaswamy, A. (2021). Distributed optimization in distribution systems: Use cases, limitations, and research needs. *IEEE Transactions on Power Systems*, *37*(5), 3469-3481. DOI: **https://doi.org/10.1109/TPWRS.2021.3132348**

[6] Liu, J., Du, Y., Yang, K., Wu, J., Wang, Y., Hu, X., ... & Leung, V. C. (2026). Edge-cloud collaborative computing on distributed intelligence and model optimization: A survey. *IEEE Communications Surveys & Tutorials*. DOI: **https://doi.org/10.1109/COMST.2026.3669216**

[7] Lee, H., Lee, S. H., & Quek, T. Q. (2022). Artificial intelligence meets autonomy in wireless networks: A distributed learning approach. *IEEE Network*, *36*(6), 100-107. DOI: **https://doi.org/10.1109/MNET.105.2100450**

[8] Tanwar, S., Popat, A., Bhattacharya, P., Gupta, R., & Kumar, N. (2022). A taxonomy of energy optimization techniques for smart cities: Architecture and future directions. *Expert systems*, *39*(5), e12703. DOI: **https://doi.org/10.1111/exsy.12703**

[9] Canese, L., Cardarilli, G. C., Di Nunzio, L., Fazzolari, R., Giardino, D., Re, M., & Spanò, S. (2021). Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, *11*(11), 4948. DOI: **https://doi.org/10.3390/app11114948**

[10] Bahrpeyma, F., & Reichelt, D. (2022). A review of the applications of multi-agent reinforcement learning in smart factories. *Frontiers in Robotics and AI*, *9*, 1027340. DOI: **https://doi.org/10.3389/frobt.2022.1027340**

[11] Li, T., Zhu, K., Luong, N. C., Niyato, D., Wu, Q., Zhang, Y., & Chen, B. (2022). Applications of multi-agent reinforcement learning in future internet: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *24*(2), 1240-1279. DOI: **https://doi.org/10.1109/COMST.2022.3160697**

[12] Kumar, V. (2025). Interoperable Knowledge Graphs for Localized Supply Chains: Leveraging Graph Databases and RDF Standards. *Logistics*, *9*(4), 144. DOI: **https://doi.org/10.3390/logistics9040144**

[13] Wiedmer, R., & Griffis, S. E. (2021). Structural characteristics of complex supply chain networks. *Journal of Business Logistics*, *42*(2), 264-290. DOI: **https://doi.org/10.1111/jbl.12283**

[14] Tsantis, A., Mangan, J., & Palacin, R. (2026). Trade shocks and direct shipping connections: causal insights into network adaptability and supply chain resilience. *WMU Journal of Maritime Affairs*, 1-33. DOI: **https://doi.org/10.1007/s13437-025-00399-0**

[15] Feng, L. (2025). Joint optimization algorithm for vehicle scheduling and supply chain inventory management based on multi-agent deep reinforcement learning. *Neural Computing and Applications*, *37*(34), 28643-28669. DOI: **https://doi.org/10.1007/s00521-025-11661-0**

[16] Feizabadi, J., Gligor, D., & Alibakhshi, S. (2021). Strategic supply chains: a configurational perspective. *The International Journal of Logistics Management*, *32*(4), 1093-1123. DOI: **https://doi.org/10.1108/IJLM-09-2020-0383**

[17] Azadegan, A., & Dooley, K. (2021). A typology of supply network resilience strategies: complex collaborations in a complex world. *Journal of Supply Chain Management*, *57*(1), 17-26. DOI: **https://doi.org/10.1111/jscm.12256**

[18] Kano, L., Tsang, E. W., & Yeung, H. W. C. (2020). Global value chains: A review of the

multi-disciplinary literature: Liena Kano et al. *Journal of international business studies*, *51*(4), 577-622. DOI: **https://doi.org/10.1057/s41267-020-00304-2**

[19] Zhang, S., Zheng, N., & Wang, D. L. (2022). A novel attention-based global and local information fusion neural network for group recommendation. *Machine Intelligence Research*, *19*(4), 331-346. DOI: **https://doi.org/10.1007/s11633-022-1336-1**

[20] Liu, L., Shi, Y., Pi, Y., Guo, W., & Wang, S. (2025). Efficient multi-view graph convolutional networks via local aggregation and global propagation. *Expert Systems with Applications*, *266*, 126131. DOI: **https://doi.org/10.1016/j.eswa.2024.126131**

[21] Liu, X., Wang, Q., Wei, X., & Liang, H. (2025, July). Hierarchical Attention-Driven Dynamic Graph Neural Networks for Accurate Supply Chain Demand Forecasting. In *International Conference on Intelligent Computing* (pp. 471-483). Singapore: Springer Nature Singapore. DOI: **https://doi.org/10.1007/978-981-95-0009-3_40**

[22] Farag, W. (2020, December). Multi-agent reinforcement learning using the deep distributed distributional deterministic policy gradients algorithm. In *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)* (pp. 1-6). IEEE. DOI: **https://doi.org/10.1109/3ICT51146.2020.9311945**

[23] Fan, D., Shen, H., & Dong, L. (2021, October). Multi-agent distributed deep deterministic policy gradient for partially observable tracking. In *Actuators* (Vol. 10, No. 10, p. 268). MDPI. DOI: **https://doi.org/10.3390/act10100268**

[24] Ikpe, V., & Shamsuddoha, M. (2024). Functional model of supply chain waste reduction and control strategies for retailers—The USA retail industry. Logistics, 8(1), 22. DOI: **https://doi.org/10.3390/logistics8010022**

[25] Ovezmyradov, B. (2022). Product availability and stockpiling in times of pandemic: causes of supply chain disruptions and preventive measures in retailing. *Annals of Operations Research*, 1-33. DOI: **https://doi.org/10.1007/s10479-022-05091-7**

[26] Prashanth, L. A., & Michael, C. F. (2022). Risk-sensitive reinforcement learning via policy gradient search. *Foundations and Trends in Machine Learning*, *15*(5), 537-693. DOI: **https://doi.org/10.1561/9781638280279**

[27] Moghaddam, A. R., & Kebriaei, H. (2024). Expected policy gradient for network aggregative Markov games in continuous space. *IEEE Transactions on Neural Networks and Learning Systems*, *36*(4), 7372-7381. DOI: **https://doi.org/10.1109/TNNLS.2024.3387871**

[28] Tatarenko, T., Shi, W., & Nedić, A. (2020). Geometric convergence of gradient play algorithms for distributed Nash equilibrium seeking. *IEEE Transactions on Automatic Control*, *66*(11), 5342-5353. DOI: **https://doi.org/10.1109/TAC.2020.3046232**

[29] Ma, C., Zhang, L., You, L., & Tian, W. (2024). A review of supply chain resilience: A network modeling perspective. *Applied Sciences*, *15*(1), 265. DOI: **https://doi.org/10.3390/app15010265**

[30] Wang, J., Pal, A., Yang, Q., Kant, K., Zhu, K., & Guo, S. (2022). Collaborative machine learning: Schemes, robustness, and privacy. *IEEE transactions on neural networks and learning systems*, *34*(12), 9625-9642. DOI: **https://doi.org/10.1109/TNNLS.2022.3169347**