Check for updates

ASME Journal of Computing and Information Science in Engineering
Online journal at:
https://asmedigitalcollection.asme.org/computingengineering

# Adaptive Network Intervention for Complex Systems: A Hierarchical Graph Reinforcement Learning Approach

**Qiliang Chen**
Department of Mechanical and Industrial Engineering,
Institute of Experiential AI,
and Network Science Institute,
Northeastern University,
Boston, MA 02115
e-mail: chen.qil@northeastern.edu

**Babak Heydari**[1]
Department of Mechanical and Industrial Engineering,
Institute of Experiential AI,
and Network Science Institute,
Northeastern University,
Boston, MA 02115
e-mail: b.heydari@northeastern.edu

*Effective governance and steering of behavior in complex multiagent systems (MAS) are essential for managing system-wide outcomes, particularly in environments where interactions are structured by dynamic networks. In many applications, the goal is to promote prosocial behavior among agents, where network structure plays a pivotal role in shaping these interactions. This article introduces a hierarchical graph reinforcement learning (HGRL) framework that governs such systems through targeted interventions in the network structure. Operating within the constraints of limited managerial authority, the HGRL framework demonstrates superior performance across a range of environmental conditions, outperforming established baseline methods. Our findings highlight the critical influence of agent-to-agent learning (social learning) on system behavior: under low social learning, the HGRL manager preserves cooperation, forming robust core-periphery networks dominated by cooperators. In contrast, high social learning accelerates defection, leading to sparser, chain-like networks. Additionally, the study underscores the importance of the system manager's authority level in preventing system-wide failures, such as agent rebellion or collapse, positioning HGRL as a powerful tool for dynamic network-based governance.*
[DOI: 10.1115/1.4068483]

*Keywords: network intervention, multiagent system, graph neural networks, deep reinforcement learning, hierarchical structure, artificial intelligence, engineering informatics, multiscale modeling and simulation*

## 1 Introduction

Many contemporary engineering systems—from smart grids and adaptive supply chains to large-scale socio-technical networks—operate through repeated interactions among autonomous agents. In most of these systems, each agent aims to optimize its local utility, consequently, the emergent collective behavior can either strengthen the entire system or drive it toward undesirable outcomes. For instance, collaborative sourcing in supply chains can reduce overall costs, but pervasive free-riding can create bottlenecks [1,2]. Similarly, individuals in digital communication networks may share resources equitably or hoard bandwidth for personal gain [3], with network-wide repercussions for reliability and efficiency.

Ensuring that agents retain their autonomy, yet avoid behaviors that jeopardize system-level goals for the sake of short-sighted individualistic gains is thus a central challenge in engineering design and governance [4,5]. Traditional approaches—such as centralized controls or algorithmic contract enforcement—often prove costly, require extensive compliance monitoring, or fail to address complex agent interdependencies. *Network-based governance* offers a compelling alternative by strategically structuring interaction patterns to naturally foster beneficial behaviors within the system. This approach proves particularly valuable when direct control over individual decisions is impractical or undesirable, and when agents' strategic choices are influenced by their neighbors—either through uncertainty reduction via information sharing (e.g., Bayesian updating) or through peer-based learning of superior strategies (e.g., social learning).

*Network-based governance*, however promising in theory, holds three interconnected challenges that complicate its effective implementation.

First, multiagent systems typically involve interactions among individuals operating across varying levels of cooperation, coordination, and competition [6,7]. Furthermore, the network topologies of these systems (which agent interacts with whom) are not static but evolve through both external interventions and agent-to-agent interactions, in which agents are likely to adopt superior interaction strategies from other agents that they can observe. This *social learning* process, limited until recently to human agents [8], is now embedded (often implicitly) in AI agents such as those based on large language models [9,10]. As a result, these systems exhibit nonstationary dynamics, constantly changing and requiring highly flexible and adaptive governance strategies.

Second, in real-world scenarios, the system manager's authority is usually constrained. Excessive or overly frequent rewiring of

[1]Corresponding author.
Manuscript received October 25, 2024; final manuscript received April 7, 2025; published online April 30, 2025. Assoc. Editor: Astrid Layton.


Copyright © 2025 by ASME; reuse license CC-BY 4.0

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

network links can consume significant resources, leading to cost overruns or instability [11,12]. Crafting governance strategies that operate within these practical limits is crucial to gaining agent compliance and ensuring long-term viability [13].

Third, both the state space (network configurations) and action space (possible interventions) grow exponentially with the number of agents. Monitoring, analyzing, and influencing a complex network of size $N$ becomes intractable as the number of potential topologies surges on the order of $2^{N(N-1)/2}$ [14]. Likewise, even a simple action space—expressed as $\mathcal{O}(N(N-1)/2)$ possible link modifications—poses a bottleneck for decision-making. Faced with limited training opportunities, the network governing mechanism must distill critical information from a vast range of possibilities, often under time or budget constraints, which makes finding an efficient intervention policy a daunting task.

To address the rapidly expanding state and action spaces in network-based governance, we propose a hierarchical graph reinforcement learning (HGRL) framework that integrates graph neural networks' (GNNs) expressive capabilities [15,16] with hierarchical reinforcement learning's (HRL) modularity [17,18], applied to the action space of the reinforcement learning (RL) agent in charge of adaptive network governance. GNNs embed the evolving network into a compact representation, enabling the manager to monitor system-wide interactions in real time, while HRL structures the decision-making process into multiple layers, breaking complex tasks into manageable subproblems. This layered approach is ideally suited for large-scale multiagent systems where agents' strategies adapt continuously, creating nonstationary environments [19].

While previous research has combined HRL with GNNs [20,21] applied hierarchical schemes to specific graph tasks (see Sec. 2), our HGRL framework addresses the unique challenges of dynamic network intervention by implementing several novel techniques and model considerations. First, we implement topological rather than temporal abstraction—departing from conventional HRL approaches. Our manager agent selects a target node for intervention and then strategically determines which edges to add or remove around that node, effectively reducing the action space from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. This design choice represents more than a methodological innovation serving model efficiency; it builds upon established separability approximation principles in many complex networks [22,23], allowing us to decompose link functions into individual endpoint contributions and evaluate interventions independently. Second, while most existing GNN-based reinforcement learning methods operate on relatively static network structures, our framework is explicitly built for dynamic topological evolution. This capability enables strategic link creation or removal based on both local agent interactions and global performance objectives—a critical feature for effective network governance. Third, our bottom-up network evolution framework incorporates a peer-to-peer influence mechanism that enables agents to adopt strategies from more successful neighbors. This social learning component is essential for demonstrating the effectiveness of network-based governance, as it creates the pathway through which structural interventions propagate behavioral changes throughout the system.

These adaptations maintain computational tractability as networks scale while capturing essential economic and structural trade-offs in real-world systems. Our HGRL implementation organizes network interventions into two-stage decision rounds: first selecting a target node and then specifying which connections around that node to modify. Between intervention rounds, agents interact with network neighbors and update behaviors through probabilistic peer-to-peer imitation, adopting strategies from higher-performing neighbors.

We evaluate HGRL across a diverse set of multiagent environments, scaling up to networks of 50 nodes. To isolate the fundamental governance challenges from domain-specific complexities, we model agent interactions through canonical *social dilemma* games that precisely calibrate the misalignment between individual incentives and system-level objectives [24,25]. Specifically, we implement pairwise interactions using three classic game-theoretic frameworks: prisoner's dilemma (PD), stag hunt, and snowdrift. We strategically select PD as our primary testbed—representing the most challenging cooperation scenario—while validating our approach across the full spectrum of games to demonstrate its generalizability. Agents employ social learning mechanisms [26], adopting strategies from higher-performing neighbors at variable imitation rates, allowing us to examine how different social influence intensities propagate through the network and shape collective outcomes.

To ensure robust evaluation across diverse network structures, we initialize environments using multiple topological models: random graphs (as our baseline), scale-free Barabási–Albert networks, and small-world Watts–Strogatz configurations. For benchmarking, we first compare HGRL against conventional "flat" deep reinforcement learning (DRL) in networks up to 20 nodes—the ideal comparison to demonstrate the effectiveness of our topological hierarchical approach. However, as network size increases beyond this threshold, the computational demands of flat DRL become prohibitive, highlighting a key limitation addressed by our method. To further validate HGRL's effectiveness, we employ two additional benchmarks: a random intervention policy and a "joint learning" approach that simultaneously optimizes node and link decisions without hierarchical decomposition. The random policy comparison reveals that flat DRL performance degrades significantly in larger networks, often performing no better than random interventions—underscoring the critical need for our hierarchical approach. The general diagram for the framework is in Fig. 1.

Our experimental results conclusively demonstrate HGRL's distinct advantages across several dimensions. The framework exhibits markedly improved resilience against aggressive imitation dynamics—a critical capability as AI agents increasingly trained on similar datasets adopt homogeneous behaviors that can destabilize cooperation in networked systems.

Beyond demonstrating improved system-level alignment between individual and collective goals, analysis of the emergent network topologies reveals instructive patterns that align with established network science principles. With low imitation rates, HGRL naturally evolves toward robust core-periphery structures where cooperative agents occupy central positions—consistent with findings on optimal arrangements for sustaining cooperation in social networks. Conversely, when facing high social learning pressures where defection propagates rapidly, HGRL adaptively forms sparse, chain-like structures that effectively contain contagious noncooperative behaviors, mirroring theoretical predictions about optimal containment strategies in epidemic-like behavioral spread. These findings underscore the critical role of calibrated managerial authority in preventing system-wide failure modes, including cascading defection and network fragmentation—challenges that become increasingly relevant in large-scale autonomous systems.

The remainder of this article is organized as follows. Section 2 surveys relevant literature in hierarchical RL and GNN-based approaches, underscoring the gap that HGRL addresses. Section 3 details our framework, including the architecture of the node-level and link-level agents, as well as the imitation mechanism. Section 4 presents our experimental setup, evaluation metrics, and results on multiple network topologies and payoff matrices. Finally, Sec. 5 concludes with a discussion of limitations and potential avenues for future work.

## 2 Background

This section reviews literature related to our article. We begin by providing background information on network interventions in multiagent systems, which is the central topic of this article. Subsequently, we discuss articles that integrate graph neural networks with deep reinforcement learning, a methodology our approach also employs by combining these two methods.

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

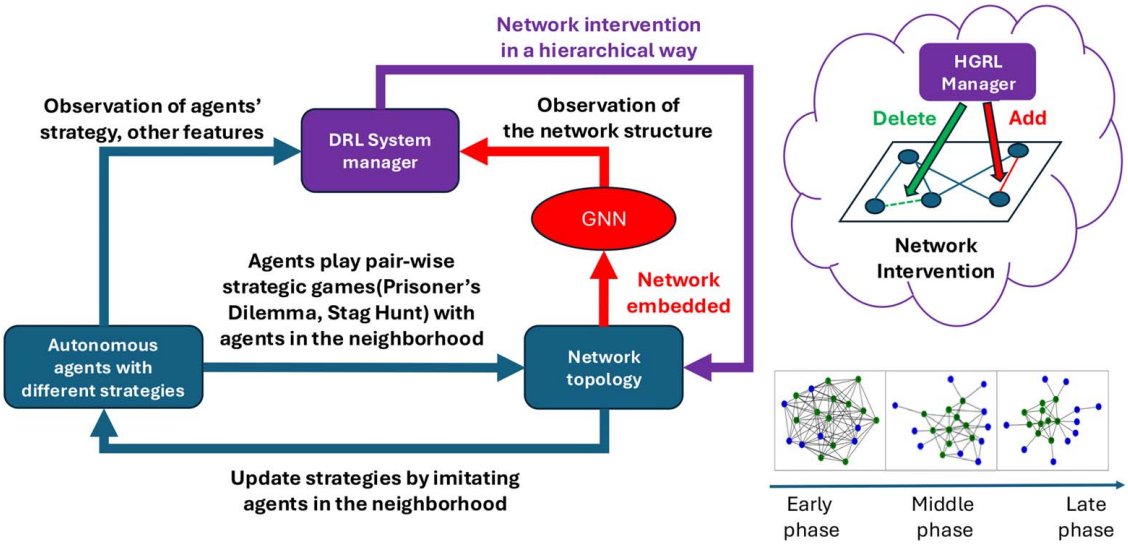


**Fig. 1 The diagram of the general process in the environment. We have three entities in the whole process: HGRL framework (details are shown in Fig. 2), autonomous agents, and network topology. In each round of the game, agents with different types interact with their first-order neighbors in the networks to play pairwise strategic games (like prisoner's dilemma, stag hunt, etc.); the HGRL framework will observe the information from agents and network topology using GNN to make decision of network intervention by adding or deleting links in the network; agents will then imitate others with higher utilities under different imitation probabilities. This process will be iterated for several time steps and we can generate the analysis of performance and network evolution from different perspectives.**

**2.1 Network Intervention in Multiagent System.** Comprehensive research in network science strongly supports the idea that the performance and behaviors of systems with network structures can be effectively managed through network intervention. Besides, numerous studies build on this premise, developing general models and methodologies applicable to various contexts.

Li et al. [27] integrate graph and control theory to address challenges in structural controllability and optimal control of large-scale networks. Besides, Ding et al. [28] develop a series of algorithms to identify key nodes in networks. Intervention in these nodes can significantly influence the overall system. Apart from cooperative scenarios, Zhang et al. [29] introduce a mathematical framework using signed graphs and dynamical system theories in multiagent systems with "coopetition networks," leveraging distributed interaction laws and neural network-based estimators for disturbance adaptation.

Additionally, several research efforts aim to solve network intervention tasks in specific areas, further expanding the scope and application of these methodologies. In the public health area, Siciliano and Whetsell [30] propose a new framework for network interventions in public administration, drawing on Valente's 2012 topology and public health concepts, to enhance behavioral

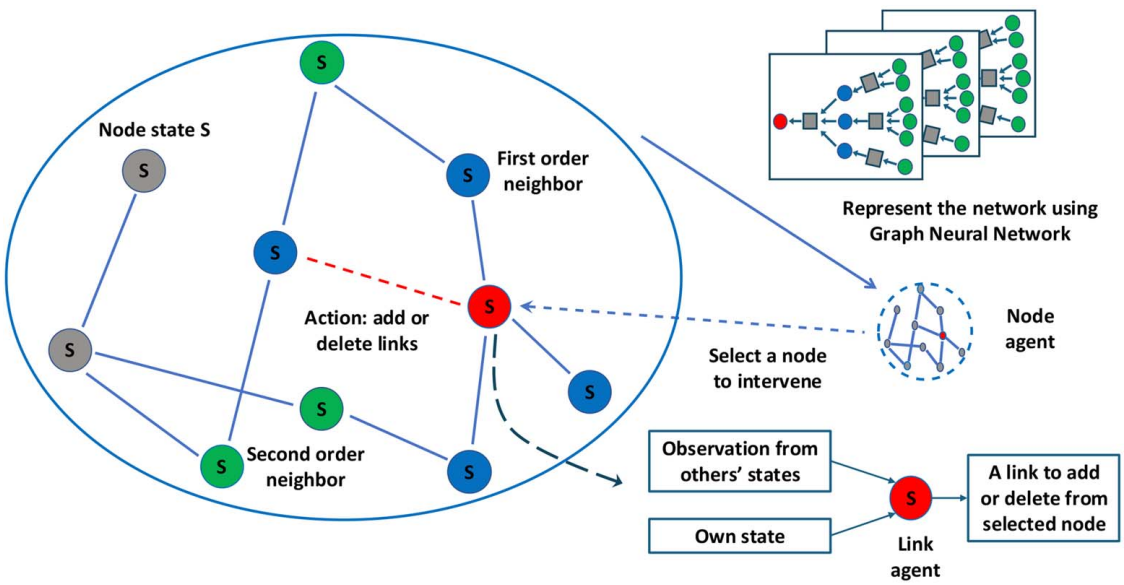


**Fig. 2 General framework of HGRL. HGRL has two agents: node agent and link agent. The node agent will collect graph information using GNNs and select one node to intervene; the link agent will rely on the information related to the selected node to decide to add or delete links to this node. The action spaces for node agent and link agent are $O(N)$, where $N$ is the number of nodes in the network.**



Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

change and improve outcomes at social, organizational, and community levels. Various aspects of dynamic network interventions have also been explored within socio-technical systems, such as assessing the effectiveness of *network governance* through the ratio of two game-theoretic concepts—the price of anarchy and the price of stability [31]—or examining how network formation shapes the implementation of self-management support interventions in community settings [32]. In the infrastructure systems, Hearnshaw and Wilson [33] suggest that the system manager builds a supply chain with the presence of hub firms and redundancy and undertake a multisourcing strategy or intermediation between hub firms to increase system resilience.

Most existing studies in this field concentrate on specific domains, employing extensive domain knowledge to devise strategies for network intervention in multiagent systems. However, the transferability of these strategies across various fields is limited. Our work is pioneering in addressing these common challenges we proposed in the context of network intervention problems in multiagent systems with an innovative HGRL framework, designed to be versatile and applicable across a variety of application areas.

**2.2 Frameworks With the Combination of Graph Neural Network and Deep Reinforcement Learning.** The evolution of RL has emerged as a universal solution for decision-making across various domains, yielding remarkable results in fields such as robotics [34–36], video gaming [37], and design and governance of engineering systems [18,38]. Besides, GNNs have demonstrated significant advancements in the perception of graph information in different fields like bio-medicine [39], computer vision [40], and social science [41–43]. Consequently, when addressing decision-making in graph-based environments, the integration of GNNs with RL becomes a popular approach.

Almasan et al. [20] proposed a GNN-based DRL agent to solve routing optimization problems and the results show that the DRL + GNN agent can outperform state-of-the-art solutions in the testing phase. To improve the network embedding task, Yan et al. [44] combined deep reinforcement learning with a novel neural network structure based on graph convolutional networks and proposed a new algorithm for automatic virtual network embedding. Chen et al. proposed a deep reinforcement learning-based algorithm in Ref. [45] that combines a graph convolutional neural network with a deep $Q$-network to serve as the information fusion module and decision processor, enhancing the operation of a connected autonomous vehicle network. Meirom et al. [46] focus on the problem of controlling a partially observed dynamic process on a graph with a limited number of interventions. They formulate the problem as a sequential decision problem over a temporal graph process with GNNs and the experiments focus on prioritizing which nodes should be tested to curb the spread of an epidemic and on influence maximization within a graph, demonstrating good performance compared to baseline methods. In chemical process design, Stops et al. [47] modeled chemical processes as graphs and used GNN and DRL to learn from process graphs, which is capable of designing viable flowsheets economically in an illustrative case study. In smart manufacturing, Yang et al. [48] combined GNN and DRL to solve dynamic job shop scheduling problems, with higher effectiveness and feasibility than regular DRL. McKee et al. [49] tried to tackle the problem of promoting pro-social behaviors through network intervention with GNN plus DRL, and the results on human groups show the superiority of the framework.

While the integration of GNNs with DRL has been applied across various domains, its implementation in addressing network intervention issues remains underexplored. Our research marks a significant advancement in crafting effective network intervention strategies within multiagent systems. Additionally, our proposed HGRL framework considerably simplifies the complexity of the state and action spaces, enhancing the efficiency of the learning process.

## 3 Methods

In this section, we will begin by presenting a general mathematical formulation of the problem we aim to solve, which includes the initial settings of the environment, system dynamics, and key mechanisms embedded within the problem, such as imitation learning and the process of governance employed by HGRL managers. Subsequent sections will detail several critical components of the general framework, such as partially observable Markov decision processes, deep $Q$-learning, and GNNs. Finally, we will discuss the specifics of our proposed HGRL framework, including how it learns and executes policies to address the problem after training.

**3.1 Mathematical Formulation.** *Problem setting.* There are $N$ agents (nodes) $\{1, 2, \ldots, N\}$ playing a repeated social dilemma game. These agents form a dynamic network $G(t) = (V, E(t))$ at each discrete time $t$, where $V = \{1, \ldots, N\}$ and $E(t) \subseteq V \times V$. Each agent $i$ has an internal strategy $s_i(t)$ for the stage game, which is not directly observable by the RL controllers. The stage game actions (e.g., cooperate/defect) of the agents, however, are observable. Let $a_i(t)$ denote the observed action of agent $i$ at time $t$.

*Partial observability.* We assume the true environment state $S(t) = (G(t), \mathbf{s}(t))$, $\mathbf{s}(t) = (s_1(t), \ldots, s_N(t))$, is hidden. Instead, the RL agents receive an *observation* $O(t)$, which is an *embedding* of the network and the agents' last actions, for instance via a GNN:

$$O(t) = \phi(G(t), \mathbf{a}(t)) \in \mathbb{R}^d$$

where $\mathbf{a}(t) = (a_1(t), \ldots, a_N(t))$ is the vector of all agents' actions at time $t$ and $\phi(\cdot)$ is a graph neural network or similar encoder.

*Hierarchical RL agents.* There are two RL agents acting in a hierarchical fashion each time step:

- *Node RL agent*: chooses a node $i^*(t) \in \{1, \ldots, N\}$ to *target*.
- *Link RL agent*: decides how to modify links (add/remove) *for that targeted node* $i^*(t)$, subject to a budget of $l_{\max}$ changes per period.

Formally, the overall action at time $t$ is

$$A(t) = (A_{\text{node}}(t), A_{\text{link}}(t)) = \left(i^*(t), \Delta E_{i^*(t)}(t)\right)$$

Here, $\Delta E_{i^*(t)}(t) = \left(\Delta E^+_{i^*(t)}(t), \Delta E^-_{i^*(t)}(t)\right)$ specifies which edges are added ($\Delta E^+_{i^*(t)}(t)$) or removed ($\Delta E^-_{i^*(t)}(t)$) for node $i^*(t)$, with

$$\left|\Delta E^+_{i^*(t)}(t)\right| + \left|\Delta E^-_{i^*(t)}(t)\right| \le l_{\max}$$

*Network transition.* After the hierarchical action is chosen,

$$\begin{aligned} G'(t) &= f_{\text{network}}\left(G(t), i^*(t), \Delta E_{i^*(t)}(t)\right) \\ &= \left(V, \left[E(t) \setminus \Delta E^-_{i^*(t)}(t)\right] \cup \Delta E^+_{i^*(t)}(t)\right) \end{aligned}$$

*Agent stage game and strategy update.* Each of the $N$ agents then takes a stage game action $a_i(t)$ (observable to RL) based on its internal strategy $s_i(t)$. The payoff for agent $i$ is

$$\Pi_i(t) = \sum_{j:(i,j)\in G'(t)} \pi_i\left(s_i(t), s_j(t)\right)$$

where $\pi_i(\cdot\,,\,\cdot)$ is the payoff function of the social dilemma. Afterward, agents update their internal strategies via social learning:

$$s_i(t+1) = f_{\text{strategy}}\left(s_i(t), \{s_j(t)\}, \{\Pi_j(t)\}\right)$$

which the RL agents do *not* directly observe.

*Next state.* Hence, the unobserved new state will be calculated based on the transitions function: $S(t+1) = (G'(t), \mathbf{s}(t+1))$.

*Reward function.* Both RL agents share the same global reward, e.g. the sum of all agent payoffs at time $t$: $R(t) = \sum_{i=1}^{N} \Pi_i(t)$, where $\Pi_i$ is the payoff function for agent $i$.

*Objective.* We treat this as a partially observable MDP (Markov Decision Process). The two-agent hierarchical policy $\pi^H_{\text{RL}} = (\pi_{\text{node}}, \pi_{\text{link}})$ seeks to maximize the expected discounted reward:

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

$$\max_{\pi_{\mathrm{RL}}^{H}} \quad \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(t)\right]$$

The mathematical formulation presented above is at a high level, capable of encompassing more complex problems by adding intricacies to the transition function, reward function, and the objectives the manager aims to optimize. However, to initially evaluate the core functionality of our framework, we make some assumptions in subsequent experiments and implementations. Although our framework and problem formulation accommodate partially observable environments, the features of the agents—represented as different types (allC, allD)—do not introduce sufficient uncertainty in our settings, so the observation level in our experiment is equivalent to full observability.

Second, although the strategy set for agents is limited (allC, allD)—where allC always chooses cooperation and allD always opts for defection—the dynamic network structure and the imitation learning mechanism complicate the evolution of the agents' types. From our experimental results analysis, we have also uncovered some interesting and nontrivial insights from the current settings.

Third, in our experiments, the budget $l_{\max}$ is constrained to 1, indicating that HGRL managers can only modify one network link at each time step. This budget symbolizes the extent of managerial authority. Although it can be adjusted to various values to depict more flexible scenarios, we have restricted it to 1 in our cases to represent the most stringent scenarios, where the manager has very limited authority.

In the following sections, we will provide detailed explanations of several components related to the high-level framework. The mathematical variables will be replaced or adapted to enhance the explainability of specific formulas.

**3.2 Deep *Q*-Learning and Actor–Critic.** In our framework, we utilize a reinforcement learning approach based on the partially observable Markov decision process (POMDP) model, which is particularly well-suited for environments where the state space is not fully visible. A POMDP can be described using a tuple $\langle S, A, O, T, R\rangle$. $S$ is a set of possible states. $A$ is a set of available actions. $O$ is the observation perceived from the state of the system which may not fully cover all information of the state. $T$ is transition function, where $T$: $O \times A \to O$. $R$ is reward function, where $O \times A \to R$. The objective in POMDP is to learn the optimal policy $P_\theta$: $O \times A \to [0, 1]$, which can maximize the expected return: $\sum_{t=0}^{T} \gamma^t r^t$ where $\gamma$ is a discount factor, $T$ is the time horizon and $r$ is the reward at each time step.

Deep $Q$-learning, as outlined in Ref. [37], is a widely utilized algorithm within the field of RL. In this approach, the agent first perceives an observation $o$, which is a representation of the environmental state $s$. Based on this observation, the agent utilizes the action-value function $Q^{\pi}(o, a) = E[R|o = o^t, a = a^t]$ under a policy $\pi$ and selects actions in a greedy manner to maximize the action value. The core objective of $Q$-learning is to refine the action-value function towards the optimal policy. This is achieved by minimizing the loss function:

$$L_\theta = E_{o,a,r,o'}[(Q(o, a|\theta) - y)^2]$$

where $y$ is defined as $r + \gamma \max_{a'} Q'(o', a')$. Here, $Q$ function is parameterized by $\theta$ using a deep neural network. The observation at the next time step is denoted as $o'$, and $a'$ represents the action taken at that time. The target $Q$ function, $Q'$, which aids in stabilizing the learning dynamics, is periodically updated and is a clone of the operational $Q$ function.

The actor–critic method combines policy-based (actor) and value-based (critic) approaches in reinforcement learning. The actor selects actions via a policy $\pi(a|s; \theta)$, updated using the policy gradient:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi(a|s; \theta) A(s, a)$$

where $\alpha$ is the learning rate and $A(s, a)$ is the advantage function. The critic estimates the value function $V(s; \phi)$, updated via temporal difference (TD) learning:

$$\phi \leftarrow \phi + \beta \delta \nabla_\phi V(s; \phi)$$

where $\beta$ is the learning rate and $\delta = r + \gamma V(s'; \phi) - V(s; \phi)$ is the TD error. The advantage $A(s, a)$ reduces variance in updates:

$$A(s, a) = Q(s, a) - V(s)$$

where $Q(s, a)$ is the action-value function.

**3.3 Graph Neural Network.** GNN [50] is a class of deep learning models that operate on graphs and have shown promising results in various tasks such as node classification, link prediction, and graph classification. Besides, GNN has demonstrated significant potential for efficient learning on large graphs [16]. Similar to convolutional neural networks (CNNs), which utilize a filter that slides across an image, GNNs employ a neighborhood aggregation function. This function slides over the graph, thereby conserving computational resources when dealing with large graphs. GNNs also adopt a pooling strategy akin to CNNs, aggregating information for each node. Once the node embeddings are obtained, they can be utilized for various downstream tasks as required. The following paragraphs provide some details about GNNs.

A graph can be represented as $G = (V, E)$, where $V$ is a set of nodes and $E$ is a set of edges connecting these nodes. The goal of GNN is to learn a function $f(G, X)$, where $X$ is a matrix of node features, that maps the graph and its node features to a target variable. There are three main operations in GNN, which are message passing, message aggregation, hidden representation update, and a readout operation, which converts all node embeddings to satisfy specific requirements of tasks.

The message-passing function can be represented as

$$m_{v\to u}^{(l)} = M^{(l)}\left(h_u^{(l-1)}, h_v^{(l-1)}, e_{u,v}\right) \tag{1}$$

where $h_u^{(l-1)}$ and $h_v^{(l-1)}$ are the hidden representations of nodes $u$ and $v$ at layer $l-1$, $e_{u,v}$ is the edge feature between nodes $u$ and $v$, $M^{(l)}$ is a message function that generates a message from node $v$ to node $u$ at layer $l$, and $m_{v\to u}^{(l)}$ is the message passed from node $v$ to node $u$ at layer $l$. The message-passing function can be modeled using deep neural networks. The Aggregation function can be written as

$$a_u^{(l)} = \mathrm{AGG}^{(l)}\left(\{m_{v\to u}^{(l)} | v \in \mathcal{N}(u)\}\right) \tag{2}$$

where $\mathcal{N}(u)$ is the set of neighboring nodes of node $u$ and $\mathrm{AGG}^{(l)}$ is an aggregation function that aggregates the messages received by node $u$ at layer $l$. The update function can be represented as

$$h_u^{(l)} = U^{(l)}\left(h_u^{(l-1)}, a_u^{(l)}\right) \tag{3}$$

where $U^{(l)}$ is an update function that updates the hidden representation of node $u$ at layer $l$ based on the hidden representation $h_u^{(l-1)}$ from the last layer and the aggregated message $a_u^{(l)}$. The above equations are applied iteratively for multiple layers until the final representation of each node is obtained. The final node representations can then be used for various downstream tasks.

In our case, we need to represent the whole graph, which is similar to the operation in GNN-based graph classification, so the readout function can be written as $z_G = \mathrm{READOUT}(h_v^{(l)} | v \in G)$ where $z_G$ is the graph-level representation obtained by aggregating the hidden representations of all nodes in the graph $G$ at the final layer $K$, and READOUT is an aggregation function that combines the hidden representations of all nodes. There are various options for the READOUT function, such as concatenation, max pooling, mean pooling, sum pooling, and some long short-term memory (LSTM)-based pooling methods [50]. We used the concatenation for the READOUT function to aggregate the information of all nodes.

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

**3.4 Hierarchical Graph Reinforcement Learning Framework.** Theoretically, to learn a policy of network intervention aimed at optimizing predefined goals, the standard deep $Q$-learning method [flat reinforcement learning (Flat-RL)] can be employed, which utilizes system information to determine which links to add or remove in various situations. However, Flat-RL encounters significant challenges when applied to network structure interventions, primarily due to the rapidly increasing complexity of both the state and action spaces as the number of nodes increases. To illustrate, consider a network with $N$ nodes, the potential structures for undirected networks scale as $O(2^{N(N-1)/2})$, which becomes intractable easily. Besides, the variability of potential node features will make it even more complex. The action space, involving the addition or deletion of links, has a complexity of $O(N(N-1))$. Consequently, both the state and action spaces in Flat-RL become exceedingly complex and potentially unmanageable as the network size increases.

To overcome this challenge, we propose the HGRL framework. The overall process and structure of this framework are visualized in Fig. 2. This framework leverages GNN for network representation, utilizes a hierarchical structure [51], and takes advantage of the separable cost idea in networks [23] to mitigate the issue of exploding state space and action space encountered in Flat-RL. GNNs efficiently embed network information, which reduces the state space complexity with less computational and memory demand. To tackle the action space reduction in link intervention, HGRL draws on the concept of separable costs in networks. Previous work used this idea by decomposing the link costs to the sum of costs from two separate end nodes, to find solutions for efficient network structures [23]. Inspired by this idea, links can be decomposed into pairs of end nodes. Instead of direct link selection, HGRL proposes selecting each end node independently. In undirected networks, the sequence of choosing these nodes is irrelevant. Thus, HGRL's approach involves a two-step hierarchical link intervention process. The initial step involves selecting one end node based on overall network information. The subsequent step depends on the first step, where the second end node is chosen based on the initially selected node and its associated information. This step effectively determines which links to add or remove from the chosen node. Essentially, HGRL simplifies the joint link selection process into two hierarchical, interdependent steps, significantly reducing the compounded action space from $O(N^2)$ to $O(N)$, while $N$ is the number of nodes in the network.

HRL [51] is a type of DRL method that decomposes complex tasks into a hierarchy of subtasks. This hierarchical approach facilitates learning and optimization, particularly in environments with long-term dependencies and sparse rewards. In HRL training, a high-level policy learns to select abstract options or subgoals, while a low-level policy executes primitive actions to achieve these subgoals or complete the assigned subtasks. This structure enables efficient learning by leveraging temporal abstraction, where macro-actions or options span multiple time steps instead of being executed in a single step. Our proposed HGRL method is inspired by the concept of separate costs introduced in Ref. [23]. However, unlike HRL, which focuses on hierarchical decision-making for subtasks, HGRL builds its hierarchical structure differently. Instead of selecting a single link, HGRL hierarchically chooses two end nodes at each time step. Despite sharing the term "hierarchical," the fundamental approach to constructing the hierarchy in HGRL is distinct from that in HRL.

In the execution phase of the HGRL framework, the initial stage involves the node agent selecting the first end node based on graph information. This is achieved using a GNN to extract network data. The core idea of combining GNN with deep reinforcement learning for node agents is straightforward. Take deep $Q$-learning as an example, in many benchmark environments, such as Atari games or robotic tasks, the $Q$ function is typically approximated using fully connected neural networks due to the relatively small state dimension. However, in our case, the state is network based. Instead of directly using a fully connected neural network to approximate the $Q$ function, we first employ a GNN to aggregate information from the network. The embedded network information is then processed by a fully connected neural network to approximate the $Q$ function, followed by the standard deep $Q$-learning process. One challenge we encountered when combining GNN with deep $Q$-learning is the need for a smaller learning rate to ensure stable training. The incorporation of GNN introduces additional complexity in feature extraction, making the learning process more sensitive to large updates. Specifically, the node agent employs the message-passing process to embed each node as we introduced in Sec. 3.2. This embedding considers both the hidden information of the node and its neighbors. The initial hidden features for layer 0 are the original node features. The embedded hidden information is then aggregated using mean pooling to obtain the first layer embedding. This embedding process can be repeated up to a total of $l$ layers, where $l$ is the number of GNN layers. The number of layers is critical as it defines the extent of the neighborhood perceived around each node; essentially, $l$ layers capture the information of the $l$th order neighbors for each node. However, it is crucial to limit the number of layers because using too many layers can lead to an oversmoothing problem, where node representations become indistinguishable as the number of layers increases, as highlighted in the study by Ref. [52]. For this application, a two-layer GNN is employed. After processing through these steps, the graph's embedding information, denoted as $h_{u_i}^{(l)} = \mathrm{GNN}(u_i^{(0)}, E)$, is obtained.

This information serves as the observation for the $Q$ function in the reinforcement learning algorithm. A fully connected neural network represents the $Q$ function, calculating the $Q$ values for selecting each node:

$$Q_{\mathrm{node}}((V, E), a_{u_i}) = \mathrm{FC}(h_{u_i}^{(l)}) = \mathrm{FC}(\mathrm{GNN}(u_i^{(0)}, E)) \tag{4}$$

where GNN means the combination of the message passing, message aggregation, and hidden representation update we mentioned above and FC is a fully connected neural network. $u_i^{(0)}$ is the original node feature for node $i$, $a_{u_i}$ is selecting node $i$ to intervene, and $h_{u_i}^{(l)}$ is the hidden information of node $i$ at $l$-layer GNNs. Then, node agent will choose the node with the highest $Q$ value.

In the second stage of the HGRL framework, the link agent is tasked with selecting the second end node to either form a new link or delete an existing link with the node chosen in the first stage. The link agent's observation includes both the aggregated information of the neighboring nodes of the initially selected node and the existing connection status of that node selected from the first stage. The $Q$ function in this stage can be represented either using a GNN or a fully connected neural network. This flexibility is due to the reduced requirement for embedding the entire graph; instead, the focus is on calculating the $Q$ values for all possible actions (adding or deleting links) from the node chosen in the first stage. Upon processing these calculations, the link agent will select the top $K$ actions that have the highest $Q$ values ($K = 1$ in our experiment). This selection is based on the intervention capability of the agent, as defined by the manager's authority. The process of conducting the second stage is similar to the first stage with different state space and action space accordingly. Importantly, the action spaces for both the node agent in the first stage and the link agent in the second stage are $O(N)$, where $N$ is the number of nodes in the network. This approach effectively mitigates the complexity of the action space, which in Flat-RL is $O(N^2)$.

The training process consists of two distinct phases. In the first phase, we train the link agent while employing a random strategy for the policy of the node agent. Initially, the node agent randomly selects one node. Subsequently, the link agent learns the policy to select a second node in order to determine which link to add or delete, based on the first selected node. Exploration is one of the important topics in RL; we use the $\epsilon$-greedy strategy to balance the exploration and exploitation in the training phase. The core idea is that when making decisions at each time step, the RL

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

agent will have *epsilon* probability to select actions randomly (exploration) instead of choosing the action with the highest $Q$ values (exploitation), where *epsion* will decrease gradually as the training proceeds to make the policy converge. After making decisions at each time step, the trajectory of observation, actions, rewards, and next observations will be stored in a replay buffer. When updating the policy, a mini-batch of trajectory will be sampled from the replay buffer, and the $Q$ function will be updated to minimize the loss function, as we introduced in Sec. 3.1.

Once the link agent's training is complete, its policy will be fixed, and we will proceed to train the node agent. The general procedure is similar, but only the definitions of observation, action, and reward for node agent are changed accordingly. The observation of the node agent will become all nodes' features and the connection status of the whole network. The action space of the node agent is selecting the first node strategically. The objectives for both the node and link agents are to improve social welfare in the system.

## 4 Experiment and Results

In this section, we first introduce the environment we have designed in this work, which effectively captures the challenges of network intervention in the multiagent systems we mentioned earlier. We then present and discuss the results of our proposed methods, examining both performance and behavioral aspects.

**4.1 Environment Design.** To assess the effectiveness of the proposed HGRL framework, it is necessary to create an environment that encapsulates the key aspects of challenges as previously discussed. This environment should accurately represent the intricate interactions among agents within a multiagent system structured as a network. Additionally, it must reflect the dynamic and complex nature of agent behavior and account for the constrained authority held by the system manager. First, we chose the PD as the central mechanism in our network model, which is grounded in its proven effectiveness in representing the complexities of cooperative and competitive interactions [53]. This fundamental game in Game theory exemplifies the conflict between individual rationality and collective benefit, making it an ideal framework for studying decision-making in socio-technical systems. This game has been widely studied in various disciplines, including economics [54], sociology [55], and evolutionary biology [56], to understand cooperative behavior among rational individuals. The nodes in the network are players, and they will play PD with their neighbors using their own policies and receive rewards based on the utility matrix.

Besides, agents can only observe their direct neighbors and imitate others' policies if the others' total utility is higher than theirs. The limited range of observation represents the partial observable challenges, and the process of imitation mirrors the social learning process, a fundamental aspect of human society, which is extensively discussed in Bandura's social learning theory [8]. This imitation mechanism not only reflects the adaptability and evolution of strategies within a network but also underscores the importance of social influence and learning in shaping individual and collective behavior.

In addition, the environment takes into account the limited authority of the system manager. This is achieved by constraining the system manager's ability to intervene in the network structure, allowing modifications to only a limited number of links at each time step, which, in our specific scenario, is restricted to one link. This aspect is crucial for ensuring the realism and applicability of our study to real-world socio-technical systems. In practical scenarios, system managers often do not possess infinite power to enforce rules or dictate behaviors within a network. Their influence is invariably circumscribed by legal, ethical, and practical constraints [57]. By considering the constraints on the system manager's authority, our model accurately represents the complexities and limitations that are inherent in the management and regulation of socio-technical systems.

We provide some details of the environmental properties as follows. The agents will be randomly initialized with one of the two policy types: "cooperator" or "defector." A cooperator consistently opts to cooperate, while a defector consistently chooses to defect during the game. In our environment, the utility matrix for the PD is defined as follows: for mutual cooperation (CC), both players receive a utility of $(-0.5, -0.5)$; if one cooperates and the other defects (CD), the cooperator receives $-4$ while the defector gets 0; in the reverse scenario (DC), the cooperator receives $-4$ and the defector 0; and for mutual defection (DD), both players receive a utility of $(-3, -3)$. It is important to highlight that the HGRL model is adaptable to various utility matrices and capable of learning different policies accordingly. The utility matrix used here serves merely as an illustrative example, without any specific intent or preference for this particular set of utilities. The initial network structures are randomly established using the Erdos–Rényi model of network formation [58] with 0.5 probability of building links. The dynamics of agent interaction further involve a probability factor $p$. After interacting with their neighbors, each agent has a $p$ chance of evaluating one of their neighbors. This evaluation is based on comparing the total utilities gained in the last time step. If the neighbor's total utilities are higher, the agent will then imitate the type of that selected neighbor. The system manager can only add or delete one link at each time step because of limited authority, and its objective is to maximize the social welfare of the system. We tested our framework on 10-node networks and 20-node networks. After the network intervention, the disconnected networks were not allowed in our experiments. In the later section, we have conducted experiments with the larger network with 50 nodes, across different games and initial networks, to validate the scalability and robustness of the proposed HGRL.

We provide some details related to training. The rewards for both the link agent and the node agent are the social welfare of the system, defined as the sum of agents' utility at each time step.

For the 20-node experiments:

- The *link agent* used a fully connected neural network with a $256 \times 256$ architecture to approximate the $Q$ function, with a learning rate of $lr = 0.0001$ and a discount factor of $\gamma = 0.99$.
- The *node agent* used a GNN with two graph convolutional layers (GCNs) and 64 feature vectors to approximate the $Q$ function, with a learning rate of $lr = 0.0001$ and a discount factor of $\gamma = 0.99$. Both agents have been trained for 5000 episodes, with each episode consisting of 100 steps.

For the 50-node experiments:

- The *link agent* used a fully connected neural network with a $512 \times 512$ architecture to approximate the Q function, with a learning rate of $lr = 0.0001$ and a discount factor of $\gamma = 0.99$.
- The *node agent* used a GNN with two GCNs and 128 feature vectors to approximate the actor and critic, with a learning rate of $lr = 1 \times 10^{-6}$ for actor and $lr = 5 \times 10^{-7}$ for critic and a discount factor of $\gamma = 0.99$. Both agents have been trained for 10,000 episodes, with each episode consisting of 100 steps.

We include details about the training time and computational resources used for the 50-node experiments to provide insights into the computational requirements. We used an Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz with 16 cores and an NVIDIA GeForce RTX 2080 Ti for training. In the 50-node experiments, each setting involves training both the link agent and the node agent. Each episode consists of 100 steps, with a total of 10,000 episodes per agent. The experiments include 27 different settings, covering combinations of 3 games, 3 network structures, and 3 imitation probabilities. The total physical training time is approximately 4 days.

The training time of HGRL for the 50-node scenarios on the current device remains manageable. However, as the network size

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

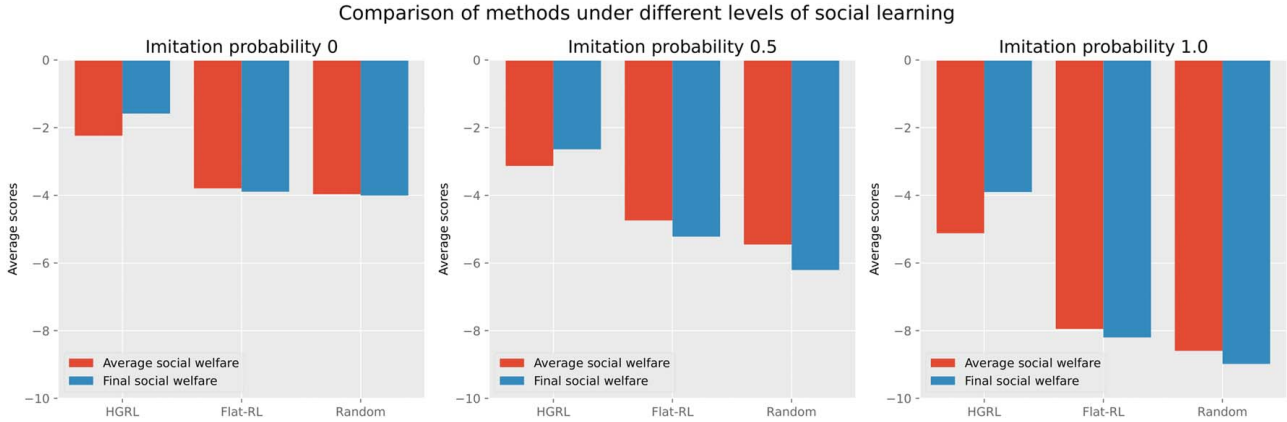


**Fig. 3 The performance comparison of HGRL, Flat-RL, and random strategy in the system with 10 nodes. The figures show results from three scenarios with different probabilities of behavior imitation. Average social welfare (left bar) during the game and average final social welfare (right bar) at the end of the game are used as metrics.**

scales to hundreds or thousands of nodes, the current HGRL framework may face significant computational challenges. Governing complex systems with very large networks remains an open question. One potential solution could involve using clustering methods to generate a "hypernetwork," where each node represents a subnetwork. HGRL could then be applied hierarchically to both the hypernetwork and individual hypernodes. However, this approach falls outside the scope of this article.

**4.2 Experiment Results—Performance.** In our study, we compared the performance of the HGRL model with that of Flat-RL and random strategy within the designed environment in both 10-node networks and 20-node networks. The HGRL model and Flat-RL model use similar sizes of parameters in neural networks. We trained the system manager for 10,000 rounds, with each round including 50 time steps for the 10-node networks and 100 time steps for the 20-node networks. The results presented are averaged from 1000 test rounds. Our experiment was conducted under three different scenarios, each characterized by varying imitation probabilities $p$ of 0, 0.5, and 1. These probabilities represent different levels of imitation tendency among the agents. To assess the effectiveness of the models, we employed two metrics: average social welfare of the game and average final social welfare at the end of the game. The performance is also averaged by the number of agents in the system. Average social welfare offers a comprehensive evaluation of the models' effectiveness, encompassing not only the general performance but also the pace of improvement in social welfare, as a swifter enhancement results in a higher average social welfare. Meanwhile, the social welfare at the end of the game evaluates the models' ability to guide the system to an optimal or desirable state by the end of the game. This dual-metric approach allows for a comprehensive analysis of HGRL performance in managing the system with a network structure. The results of the 10-node networks are in Fig. 3, and the results of the 20-node networks are in Fig. 4.

From the results, it is evident that under all scenarios, the HGRL method consistently outperforms both the Flat-RL and the random strategy in terms of both metrics. This superior performance of HGRL highlights its ability to learn more effective and adaptable policies for network structure intervention across various environmental settings. In contrast, Flat-RL can only surpass the random strategy with little improvement in small networks with 10 nodes, but in general, its performance is comparable to the random strategy, especially in large networks with 20 nodes. This suggests that with similar size of model parameters and training duration, Flat-RL struggles to develop a robust policy even in relatively small networks with just 10 nodes, which underscores the critical need and promising potential for HGRL in real-world applications. Given that many real-world multiagent systems with large networks, Flat-RL's utility diminishes, whereas HGRL retains its capacity to learn effective policies by reducing the complexity of the state and action spaces.

Besides, an insightful observation is that only the HGRL method demonstrates an ability to achieve higher social welfare at the end of games compared to the average social welfare. This observation

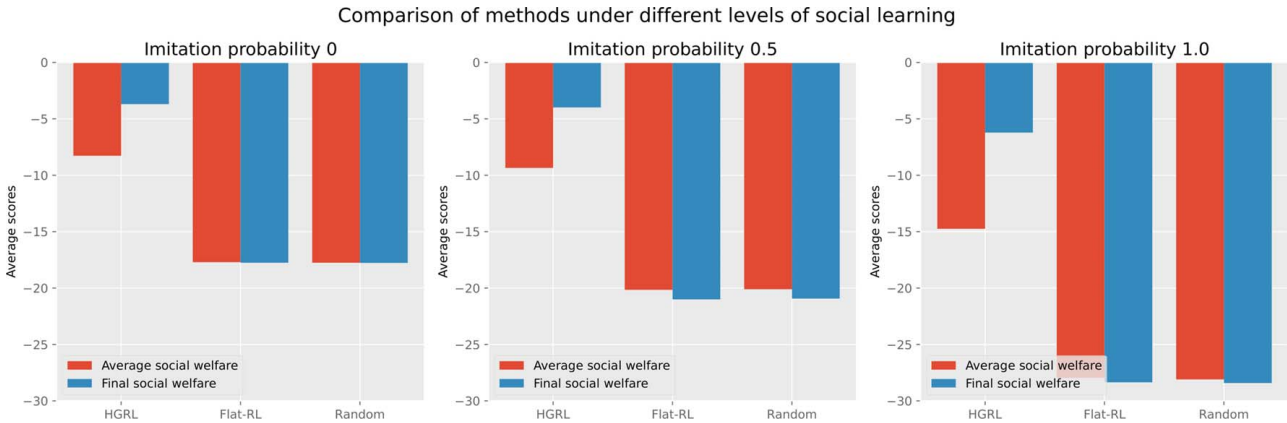


**Fig. 4 The performance comparison of HGRL, Flat-RL, and random strategy in the system with 20 nodes. The figures show results from three scenarios with different probabilities of behavior imitation. Average social welfare (left bar) during the game and average final social welfare (right bar) at the end of the game are used as metrics.**



Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

reveals that HGRL uniquely facilitates a progressive enhancement in the system's social welfare throughout the game. In contrast, neither Flat-RL nor the random strategy exhibits this capability for gradual improvement. Various studies, such as those by Refs. [59–62], have shown that defection can be "contagious" in multiagent systems due to the selfish behavior of each individual and also the modularity and cliques in network structures. Another key factor driving this trend is the bounded rationality of agents [63], particularly humans, who are often swayed by immediate rewards at the expense of long-term benefits. While defection may cater to an agent's inherent preferences, offering higher immediate rewards, it can prompt other agents to defect rather than cooperate, ultimately undermining the overall social welfare of the system. This phenomenon underscores the importance of strategies like those learned in HGRL, which not only counter short-term defection tendencies but also enhance long-term cooperative behavior, thereby improving the overall health of the system.

Lastly, the performance of all methods decreases as the probability of imitation increases. Defection can be contagious and easily spread to the system because of social learning. When the probability of imitation increases, the speed of social learning becomes more rapid, and this tendency of defection spreading can easily dominate the whole system. In our experimental settings, if every agent becomes a defector and chooses to defect all the time, the social welfare will be the lowest, which justifies the results in the figure. In the real-world situation, we can also see some evidence. If every person is extremely selfish and only considers themselves, and when this behavior spreads rapidly, society will become unsympathetic and inconsiderate, the social welfare will decrease, and the system may even collapse.

**4.3 Experiment Results—Performance Across Expanded Environments.** In this section, we extend our environment to 50 agents and evaluate it using three different games: PD, stag hunt (SH), and snowdrift (SD). We also test the environment with various initial network structures, including the Erdos–Rényi (ER) network, Barabási–Albert (BA) network, and Watts–Strogatz (WS) network. The results are presented in Fig. 5. For baselines, we introduce a method called *joint learning*, which incorporates the idea of separated costs [23]. Unlike the hierarchical training approach, where the node agent and link agent are trained separately, joint learning trains them simultaneously. Additionally, we include a *random strategy* baseline, as in previous experiments. Notably, we exclude the Flat-RL approach in the 50-agent setting, as it failed to learn a good policy in the 20-agent environment. The action space in the 50-agent case expands to $50 \times 49/2$, resulting in an excessively large hidden layer in the deep neural network, making training intractable.

The utility matrix we are using for PD is as follows: [C, C]-[−0.5, −0.5], [C, D]-[−4, 0], [D, C]-[0, −4], [D, D]-[−3, −3]; for SH: [C, C]-[−1, −1], [C, D]-[−5, −2], [D, C]-[−2, −5], [D, D]-[−2, −2]; and for SD: [C, C]-[−2, −2], [C, D]-[−5, −1], [D, C]-[−1, −5], [D, D]-[−4, −4]. The ER network has 50 nodes, where each pair of nodes is connected with a probability of 0.3; the BA network has 50 nodes, where each new node is connected to $m = 4$ existing nodes; and the WS network has 50 nodes, an initial degree of $k = 5$, and a rewiring probability of $p = 0.2$. Note that we are selecting values without a specific purpose; the only intentional choice is making the utility matrix negative, which is a common practice in RL training to

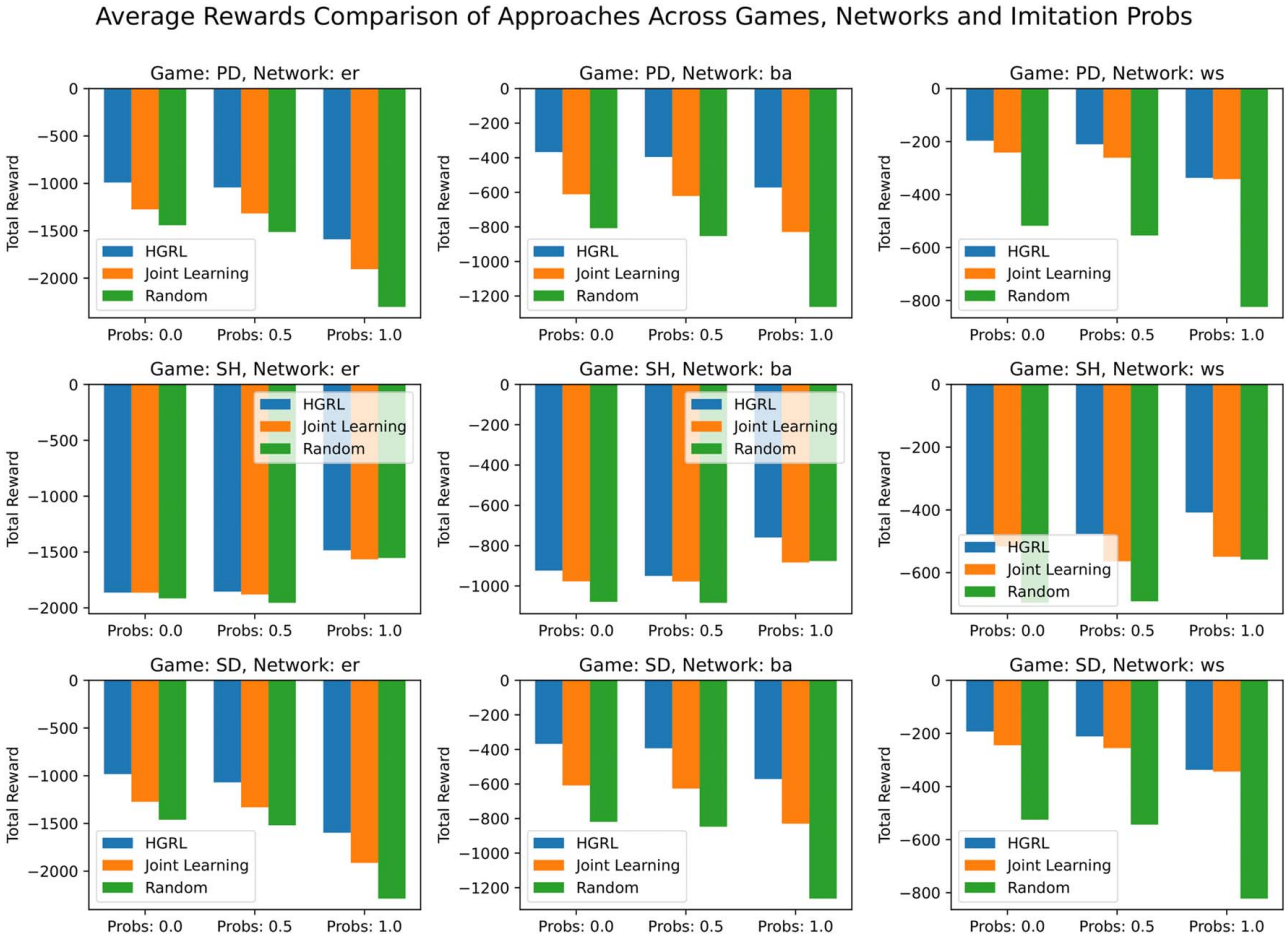


**Fig. 5 Performance comparison of HGRL, joint learning, and a random strategy in a system with 50 nodes across three strategic games (prisoner's dilemma, stag hunt, and snowdrift), three initial network structures (Erdos–Rényi, Barabási–Albert, and Watts–Strogatz), and three imitation probabilities (0.0, 0.5, and 1.0)**

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

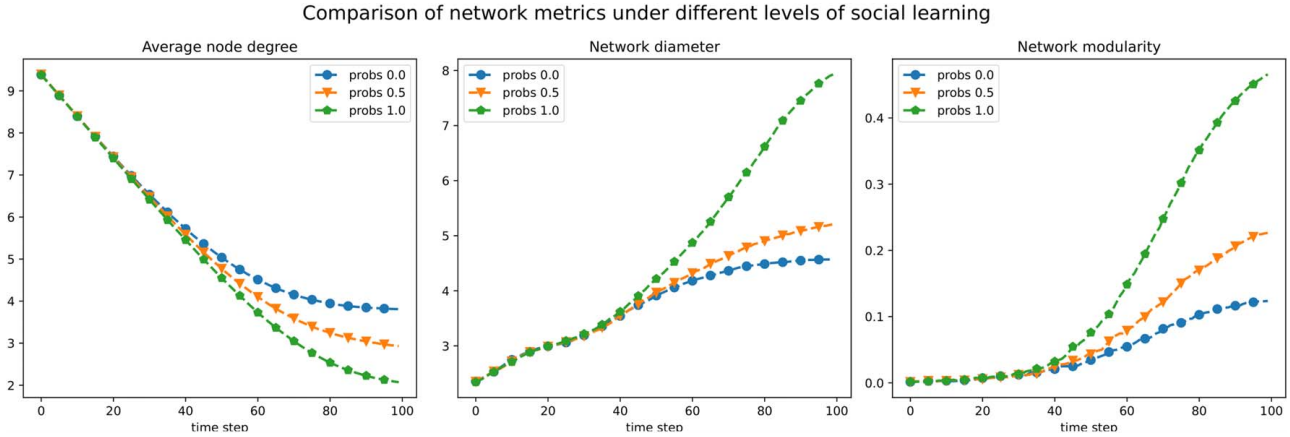


**Fig. 6 The evolution of different network metrics over time with various imitation probabilities. The figures show three network metrics: average node degree, network diameter, and network modularity. The *x*-axis is the time step.**

minimize regret. The performance in the figure is the social welfare averaged by the number of agents in the environment.

The first key insight from the results is that our HGRL method achieves the highest performance across all combinations of games and networks. This demonstrates its ability to handle larger networks with more agents while adaptively learning policies for different games with varying dilemma magnitudes. Additionally, regardless of the initial network structure, HGRL consistently learns the policy that outperforms the baselines.

Second, in some cases, the joint learning method can also discover decent policies, indicating that the concept of separate costs significantly improves training efficiency. However, in joint learning, node agents and link agents are trained simultaneously, transforming the approach into a multiagent reinforcement learning (MARL) problem [64]. Specifically, our joint learning setup resembles independent MARL, which can be effective in certain tasks but may introduce nonstationarity issues. We assume that this is the primary reason why joint learning falls short compared to the HGRL method.

Third, we observe certain trends when comparing different networks and games. The ER network tends to have lower performance than the BA and WS networks. This is primarily because the ER network has a higher number of links, leading to lower average social welfare, given that our utility matrix values are negative. When comparing performance across different games, we do not see significant differences in scale, mainly because our utility matrix settings for different games fall within a similar range.

Finally, we observe interesting insights when comparing performance across different imitation probabilities. In all networks and games—except for the stag hunt game—a higher imitation probability consistently leads to lower social welfare. This occurs because when defection spreads through the network, social welfare declines significantly, as [D, D] results in the lowest combined payoff for both agents in PD and SD. However, in the stag hunt game, mismatched types between the two agents yield the lowest performance. With a higher imitation probability, agents in the system tend to converge to the same type—whether cooperators or defectors—resulting in higher payoffs between paired agents and ultimately improving overall social welfare.

**4.4 Experiment Results—Behavior.** In the previous section, we demonstrated the superior performance of the proposed HGRL model over Flat-RL in various aspects, then it is equally crucial to examine the learned behavior of the HGRL manager. As artificial intelligence (AI) models evolve, concerns about the "black-box" problem become increasingly pronounced. Understanding the decision-making process of the HGRL model could provide valuable insights into the development of more efficient and transparent AI governance strategies.

We only analyzed the behavior of the HGRL manager and the evolution of systems in 20-node networks. Three metrics were used to study the evolution of the networks over time under varying imitation probabilities: average node degrees, network diameter, and network modularity. The results are presented in Fig. 6. Generally, the curves for different imitation probabilities begin to diverge after 30 time steps. This pattern indicates that the influence of varying levels of social learning becomes significant after 30 time steps. Consequently, the systems initially exhibit similar trends but gradually diverge under the flexible intervention of HGRL's policies.

Specifically, the average node degrees decrease through time in all scenarios, with higher social learning scenarios resulting in lower average node degrees. The initial random networks form several irrational connections, such as cooperator–defector and defector–defector links. Deleting these links benefits all scenarios, leading to generally sparser networks. In scenarios with higher social learning, it is more difficult for the system to conserve cooperators. As a result, these systems tend to be sparser which can reduce connections between defectors. Additionally, the network diameter generally increases over time, with scenarios of higher social learning resulting in larger network diameters. This pattern indicates that the networks tend to become more chain-like over time. Specifically, in scenarios with an imitation probability of 1.0, the networks conclude with 2 node degrees and a network diameter of 8 on average, essentially forming chain-like networks with few branches. Finally, the network modularity in all scenarios tends to increase over time, with higher social learning scenarios resulting in higher modularity. This indicates that agents in the system become more separated during evolution. In scenarios with lower imitation probability, more cooperators can be conserved and the presence of more connections to cooperators is beneficial to social welfare, which tends to form a large single community with lower network modularity in these scenarios.

We also visualized some snapshots of how the network evolves through time, and we randomly selected one random seed as an example to take a look at the learned behaviors in 20-node scenarios. We took some snapshots of the system evolution under three different scenarios with the imitation probability similar to previous settings, which can be seen in Fig. 7. The first row illustrates the system's evolution under a scenario with 0 imitation probability, where agent types remain constant. Observing snapshots from different times during the game, it becomes apparent that the HGRL manager has learned to strategically manipulate network connections. This strategy involves fostering connections among cooperators, while simultaneously deleting links both between defectors and between cooperators and defectors. As a result, the final state



Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

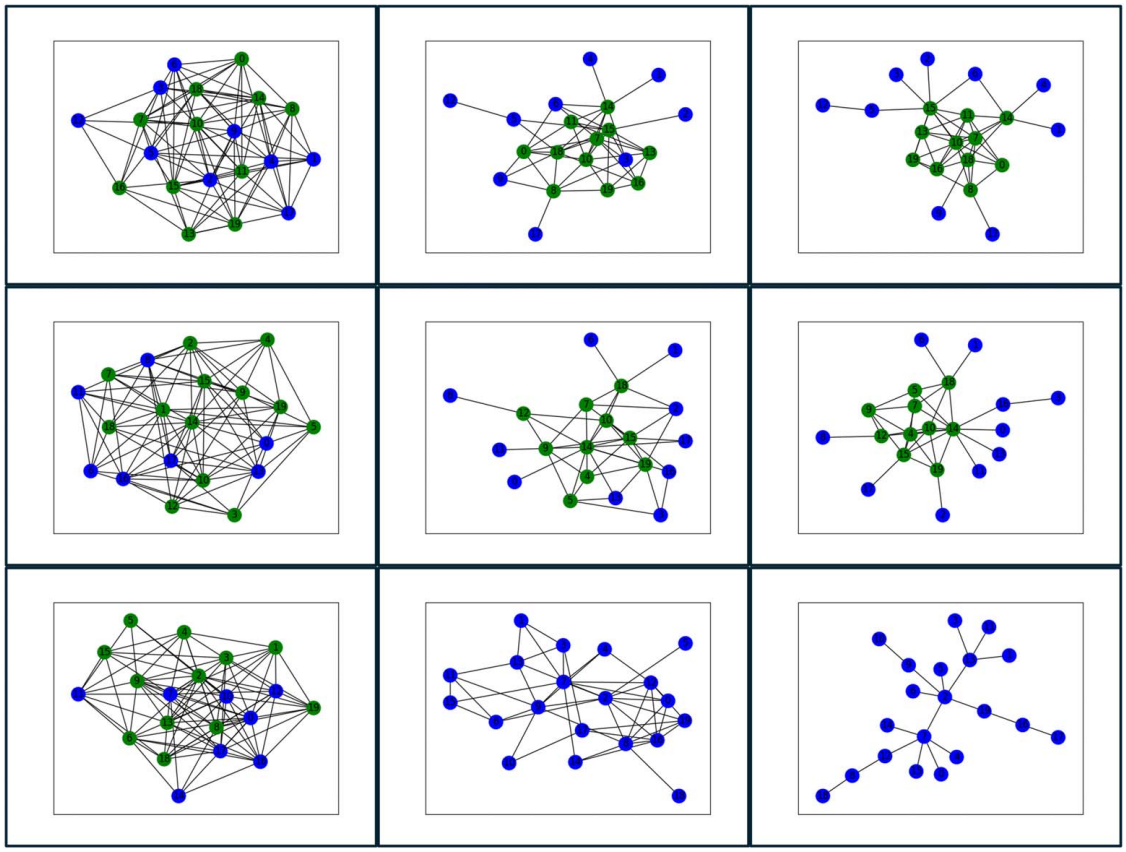

**Fig. 7 The snapshots of the system evolution with 20 nodes in the initial, middle, and final phases of the game under 0, 0.5, and 1 imitation probabilities. The green agents (lighter in grayscale) are cooperators, and the blue agents (darker in grayscale) are defectors. The images from left to right show the system status in the early phase, middle phase and late phase. The images from top to bottom show the system status when the imitation probability is 0.0, 0.5, and 1.0.**

of the system becomes a core-periphery network [65,66]: a densely connected group of cooperators as cores and more loosely connected defectors. It is important to note that our environmental settings prohibit the disconnection of the network. Since interactions between cooperators and defectors, as well as between defectors themselves, tend to negatively impact social welfare, the framework will prevent these situations by disconnecting those interactions and promoting the existence of cooperator–cooperator connections.

The second row showcases the system's evolution in a scenario with a 0.5 imitation probability, where agents gradually modify their types through social learning. During this process, defection swiftly spreads among the agents, with 3 cooperators out of 13 shifting to defectors in mid-game. In contrast to the 0 imitation probability case, where disconnecting between cooperators-defectors and deleting connections between defectors are similarly prioritized, the HGRL manager in this scenario focuses more on breaking links between cooperators and defectors. This approach is understandable given that defection, with its temptation of higher immediate rewards, spreads via interactions between cooperators and defectors. While deleting connections among defectors does enhance social welfare in the short term, the cooperators are at risk of being "contaminated" by defection through social learning. This shift can lead to a long-term decline in social welfare, as defection becomes more dominant and agents increasingly opt to defect. The HGRL manager, therefore, adopts a forward-looking strategy that prioritizes long-term welfare over immediate gains.

Interestingly, despite the spread of defection, it does not completely overpower the system by the game's end. There remain 10 cooperators due to the HGRL manager's strategic interventions. The final state reveals a robust cooperator community, resilient to defection despite some interactions with defectors. This cooperator community maintains a higher overall social welfare, safeguarding its members from deviating towards defection. To counter the spread of defection in this dynamic social learning environment, the HGRL manager's strategy is twofold: first, it targets critical links between cooperators and defectors for disconnection and, second, it rapidly builds a strong cooperator community before they become influenced by defectors. This dual-aspect approach not only preserves more cooperators but also elevates the system's overall social welfare.

The third row provides insights into scenarios where the imitation probability is set to 1.0, indicating a highly aggressive adaptation of types through social learning. In such a scenario, the transition from cooperators to defectors happens rapidly. By mid-game, all cooperators have already converted to defectors. Given these circumstances, the HGRL manager's only viable strategy is to delete connections between defectors, aiming to mitigate the deteriorating situation. Consequently, at the game's end, we observe a sparse chain-like network comprising solely defectors. In this scenario, the authority of the system manager is deliberately constrained, allowing only minor alterations—one link change per time step. Given the rapid spread of defection, this limited authority proves insufficient to manage the situation effectively. Despite the manager's efforts to disconnect defectors, which may marginally preserve social welfare, defection ultimately prevails, leading to the system's collapse.

This outcome offers valuable insights, particularly in the context of real-world applications. It underscores the importance of appropriately balancing the level of managerial authority. Excessive authority can negatively impact agent behavior, especially when agents are human. In such cases, agents might refuse to follow commands, feeling coerced into actions against their will. Conversely,



Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

insufficient authority risks leading to an anarchic system where negative outcomes can be inevitable in some scenarios, and all agents suffer. This delicate balance is crucial in ensuring effective governance and maintaining a stable, functional multiagent system.

## 5 Conclusion

This article introduces the HGRL framework, designed to govern multiagent systems with network structures through strategic network interventions. This framework is adaptable to a variety of initial network configurations and agent characteristics, accommodating the dynamics of social learning. Importantly, it operates under a realistic premise of limited authority, making it applicable to real-world systems where full control is often impractical. A key innovation of the HGRL framework is its ability to address the exponential complexity inherent in the state and action spaces of network structure interventions. It employs GNNs to manage the vast state space effectively and reduces the $O(N^2)$ action space of traditional Flat-RL to $O(N)$, due to its hierarchical structure.

Furthermore, the article introduces the creation of an environment that encapsulates critical aspects of multiagent systems with network structures. This environment not only simulates strategic interactions among agents but also incorporates the element of social learning and constrained managerial authority. From the results in the testing phase, the HGRL framework demonstrates superior performance over conventional Flat-RL in enhancing overall system performance and in guiding systems to states of higher social welfare. The improvement is consistently observed across various scenarios, each characterized by differing levels of imitation behavior in the social learning process. The learned behaviors of the HGRL framework are analyzed, affirming the rationality of its learned strategies. These strategies generate valuable heuristics that can be applied more broadly. Although the HGRL manager cannot prevent system collapse in cases of intense social learning, this outcome highlights the importance of authority levels in real-world applications and strategical governance of multiagent systems with the complex dynamics of agents and their interactions.

While the HGRL framework demonstrates significant advances in network-based governance for multiagent systems, several limitations merit acknowledgment and suggest promising directions for future research. First, our simulation environments, though diverse in game structures and network topologies, still represent abstractions of real-world systems. Future work should examine HGRL's effectiveness in domain-specific environments with more realistic agent interaction models, heterogeneous agent capabilities, and complex utility functions. Applications in supply chain management [67], digital platform governance [68,69], and critical infrastructure networks would provide valuable insights into the framework's practical utility. Second, our current implementation assumes relatively consistent agent behavior models and complete observability of network states. Extending HGRL to accommodate partial observability, stochastic agent behaviors, and adversarial agents attempting to manipulate the governance system would significantly enhance its robustness. Additionally, exploring hybrid approaches that combine network interventions with other governance mechanisms—such as incentive design, reputation systems, or norm enforcement—could yield more comprehensive governance frameworks. Finally, investigating the theoretical convergence properties of hierarchical topological abstractions in dynamic networks and developing formal guarantees for performance bounds would strengthen the mathematical foundations of this approach. As multiagent systems continue to grow in scale and complexity, particularly with the integration of AI agents exhibiting increasingly sophisticated social learning capabilities, these research directions will become increasingly vital for effective governance.

Beyond addressing the outlined limitations, certain applications require governance approaches with even lighter authority than the HGRL framework provides. A more subtle intervention mechanism would preserve agent autonomy while still effectively guiding system-wide behavior. One potential solution involves manipulating the information flow as a tool for network intervention. This approach aims to not only conserve the initial cooperators but also convert defectors to cooperators under nontrivial situations. Manipulating information as an intervention tool is advantageous because it is subtler and less noticeable to users, facilitating easier implementation. In the current framework, the policy of network structure intervention is trained to optimize social welfare, and the social learning process is influenced by the change of network structure dependently as indirect effects. Adding the idea of information manipulation, we can separate the social learning process and train another manager who recommends which agents to observe and imitate with reasonable constraints. Additionally, the imitation probability can be seen as the transparency or trustworthiness of the information. With a higher imitation probability, the information that agents can receive is more trustworthy, and they will base on it to update their strategy with less doubt. It is promising to consider changing the imitation probability dynamically to see how it can promote social welfare.

## Conflict of Interest

There are no conflicts of interest.

## Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

## References


[1] Govindan, K., Fattahi, M., and Keyvanshokooh, E., 2017, "Supply Chain Network Design Under Uncertainty: A Comprehensive Review and Future Research Directions," Eur. J. Oper. Res., **263**(1), pp. 108–141.

[2] Wu, J., and Wang, P., 2023, "Generative Design for Resilience of Interdependent Network Systems," ASME J. Mech. Des., **145**(3), p. 031705.

[3] Su, R., Zhang, D., Venkatesan, R., Gong, Z., Li, C., Ding, F., Jiang, F., and Zhu, Z., 2019, "Resource Allocation for Network Slicing in 5G Telecommunication Networks: A Survey of Principles and Models," IEEE Netw., **33**(6), pp. 172–179.

[4] Heydari, B., and Pennock, M. J., 2018, "Guiding the Behavior of Sociotechnical Systems: The Role of Agent-Based Modeling," Syst. Eng., **21**(3), pp. 210–226.

[5] Sony, M., and Naik, S., 2020, "Industry 4.0 Integration With Socio-Technical Systems Theory: A Systematic Review and Proposed Theoretical Model," Technol. Soc., **61**, p. 101248.

[6] Soria Zurita, N. F., Colby, M. K., Tumer, I. Y., Hoyle, C., and Tumer, K., 2018, "Design of Complex Engineered Systems Using Multi-Agent Coordination," ASME J. Comput. Inf. Sci. Eng., **18**(1), p. 011003.

[7] Ji, H., and Jin, Y., 2022, "Knowledge Acquisition of Self-Organizing Systems With Deep Multiagent Reinforcement Learning," ASME J. Comput. Inf. Sci. Eng., **22**(2), p. 021010.

[8] Bandura, A., and Walters, R. H., 1977, *Social Learning Theory*, Vol. 1, Prentice Hall, Englewood Cliffs.

[9] Ndousse, K. K., Eck, D., Levine, S., and Jaques, N., 2021, "Emergent Social Learning Via Multi-Agent Reinforcement Learning," Proceedings of the 38th International Conference on Machine Learning, Virtual Event, July 18–24, PMLR, pp. 7991–8004.

[10] Lorè, N., and Heydari, B., 2024, "Strategic Behavior of Large Language Models and the Role of Game Structure Versus Contextual Framing," Sci. Rep., **14**(1), p. 18490.

[11] Mosleh, M., Ludlow, P., and Heydari, B., 2016, "Resource Allocation Through Network Architecture in Systems of Systems: A Complex Networks Framework," 2016 Annual IEEE Systems Conference (SysCon), Orlando, FL, Apr. 18–21, IEEE, pp. 1–5.

[12] Zhang, X., Tian, S., Liang, X., Zheng, M., and Behdad, S., 2024, "Early Prediction of Human Intention for Human–Robot Collaboration Using Transformer Network," ASME J. Comput. Inf. Sci. Eng., **24**(5), p. 051003.

[13] Mosleh, M., Ludlow, P., and Heydari, B., 2016, "Distributed Resource Management in Systems of Systems: An Architecture Perspective," Syst. Eng., **19**(4), pp. 362–374.

[14] Trigeorgis, L., 1996, *Real Options: Managerial Flexibility and Strategy in Resource Allocation*, MIT Press, Cambridge, MA.

[15] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E., 2017, "Neural Message Passing for Quantum Chemistry," Proceedings of the 34th




Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025

International Conference on Machine Learning, Sydney, NSW, Australia, Aug. 6–11, PMLR, pp. 1263–1272.
[16] Kipf, T. N., and Welling, M., 2016, "Semi-Supervised Classification With Graph Convolutional Networks," arXiv preprint arXiv:1609.02907
[17] Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J., 2016, "Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation," Advances in Neural Information Processing Systems 29, Barcelona, Spain, Dec. 5–10.
[18] Chen, Q., Heydari, B., and Moghaddam, M., 2021, "Leveraging Task Modularity in Reinforcement Learning for Adaptable Industry 4.0 Automation," ASME J. Mech. Des., **143**(7), p. 071701.
[19] Allen, B., Lippner, G., Chen, Y.-T., Fotouhi, B., Momeni, N., Yau, S.-T., and Nowak, M. A., 2017, "Evolutionary Dynamics on Any Population Structure," Nature, **544**(7649), pp. 227–230.
[20] Almasan, P., Suárez-Varela, J., Rusek, K., Barlet-Ros, P., and Cabellos-Aparicio, A., 2022, "Deep Reinforcement Learning Meets Graph Neural Networks: Exploring a Routing Optimization Use Case," Comput. Commun., **196**, pp. 184–194.
[21] Yan, X.-W., Shi, L.-B., Yao, L.-Z., Ni, Y.-X., and Bazargan, M., 2014, "A Multi-Agent Based Autonomous Decentralized Framework for Power System Restoration," 2014 International Conference on Power System Technology, Chengdu, China, Oct. 20–22, IEEE, pp. 871–876.
[22] Hoff, P. D., Raftery, A. E., and Handcock, M. S., 2002, "Latent Space Approaches to Social Network Analysis," J. Am. Stat. Assoc., **97**(460), pp. 1090–1098.
[23] Heydari, B., Mosleh, M., and Dalili, K., 2015, "Efficient Network Structures With Separable Heterogeneous Connection Costs," Econ. Lett., **134**, pp. 82–85.
[24] Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T., 2017, "Multi-Agent Reinforcement Learning in Sequential Social Dilemmas," Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems, Sao Paulo, Brazil, May 8–12, pp. 464–473.
[25] Gianetto, D. A., and Heydari, B., 2013, "Catalysts of Cooperation in System of Systems: The Role of Diversity and Network Structure," IEEE Syst. J., **9**(1), pp. 303–311.
[26] Acemoglu, D., Dahleh, M. A., Lobel, I., and Ozdaglar, A., 2011, "Bayesian Learning in Social Networks," Rev. Econ. Stud., **78**(4), pp. 1201–1236.
[27] Li, G., Deng, L., Xiao, G., Tang, P., Wen, C., Hu, W., Pei, J., Shi, L., and Stanley, H. E., 2018, "Enabling Controlling Complex Networks With Local Topological Information," Sci. Rep., **8**(1), p. 4593.
[28] Ding, J., Wen, C., Li, G., and Chen, Z., 2019, "Key Nodes Selection in Controlling Complex Networks Via Convex Optimization," IEEE Trans. Cybern., **51**(1), pp. 52–63.
[29] Zhang, X., Mahadevan, S., Sankararaman, S., and Goebel, K., 2018, "Resilience-Based Network Design Under Uncertainty," Reliab. Eng. Syst. Saf., **169**, pp. 364–379.
[30] Siciliano, M. D., and Whetsell, T. A., 2021, "Strategies of Network Intervention: A Pragmatic Approach to Policy Implementation and Public Problem Resolution Through Network Science," arXiv preprint arXiv:2109.08197
[31] Maddah, N., and Heydari, B., 2024, "Building Back Better: Modeling Decentralized Recovery in Sociotechnical Systems Using Strategic Network Dynamics," Reliab. Eng. Syst. Saf., **246**, p. 110085.
[32] Ellis, J., Vassilev, I., James, E., and Rogers, A., 2020, "Implementing a Social Network Intervention: Can the Context for Its Workability Be Created? A Quasi-Ethnographic Study," Implement. Sci. Commun., **1**, pp. 1–11.
[33] Hearnshaw, E. J. S., and Wilson, M. M. J., 2013, "A Complex Network Approach to Supply Chain Network Theory," Int. J. Oper. Prod. Manag., **33**(4), pp. 442–469.
[34] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., 2017, "Proximal Policy Optimization Algorithms," arXiv preprint arXiv:1707.06347
[35] Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S., 2020, "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning," Conference on Robot Learning, Auckland, New Zealand, Dec. 14–18, PMLR, pp. 1094–1100.
[36] Poudel, L., Zhou, W., and Sha, Z., 2020, "A Generative Approach for Scheduling Multi-Robot Cooperative Three-Dimensional Printing," ASME J. Comput. Inf. Sci. Eng., **20**(6), p. 061011.
[37] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., et al., 2015, "Human-Level Control Through Deep Reinforcement Learning," Nature, **518**(7540), pp. 529–533.
[38] Chen, Q., and Heydari, B., 2022, "Dynamic Resource Allocation in Systems-of-Systems Using a Heuristic-Based Interpretable Deep Reinforcement Learning," ASME J. Mech. Des., **144**(9), p. 091711.
[39] Zhang, X.-M., Liang, L., Liu, L., and Tang, M.-J., 2021, "Graph Neural Networks and Their Current Applications in Bioinformatics," Front. Genet., **12**, p. 690049.
[40] Cao, P., Zhu, Z., Wang, Z., Zhu, Y., and Niu, Q., 2022, "Applications of Graph Convolutional Networks in Computer Vision," Neural Comput. Appl., **34**(16), pp. 13387–13405.
[41] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D., 2019, "Graph Neural Networks for Social Recommendation," WWW '19: The World Wide Web Conference, San Francisco, CA, May 13–17, pp. 417–426.
[42] Gao, L., Wang, H., Zhang, Z., Zhuang, H., and Zhou, B., 2022, "HetInf: Social Influence Prediction With Heterogeneous Graph Neural Network," Front. Phys., **9**, p. 787185.
[43] Tomy, A., Razzanelli, M., Di Lauro, F., Rus, D., and Santina, C. D., 2022, "Estimating the State of Epidemics Spreading With Graph Neural Networks," Nonlinear Dyn., **109**(1), pp. 249–263.
[44] Yan, Z., Ge, J., Wu, Y., Li, L., and Li, T., 2020, "Automatic Virtual Network Embedding: A Deep Reinforcement Learning Approach With Graph Convolutional Networks," IEEE J. Sel. Areas Commun., **38**(6), pp. 1040–1057.
[45] Chen, S., Dong, J., Ha, P., Li, Y., and Labi, S., 2021, "Graph Neural Network and Reinforcement Learning for Multi-Agent Cooperative Control of Connected Autonomous Vehicles," Comput.-Aided Civil Infrastruct. Eng., **36**(7), pp. 838–857.
[46] Meirom, E., Maron, H., Mannor, S., and Chechik, G., 2021, "Controlling Graph Dynamics With Reinforcement Learning and Graph Neural Networks," Proceedings of the 38th International Conference on Machine Learning, Virtual Event, July 18–24, PMLR, pp. 7565–7577.
[47] Stops, L., Leenhouts, R., Gao, Q., and Schweidtmann, A. M., 2023, "Flowsheet Generation Through Hierarchical Reinforcement Learning and Graph Neural Networks," AIChE J., **69**(1), p. e17938.
[48] Yang, Z., Bi, L., and Jiao, X., 2023, "Combining Reinforcement Learning Algorithms With Graph Neural Networks to Solve Dynamic Job Shop Scheduling Problems," Processes, **11**(5), p. 1571.
[49] McKee, K. R., Tacchetti, A., Bakker, M. A., Balaguer, J., Campbell-Gillingham, L., Everett, R., and Botvinick, M., 2023, "Scaffolding Cooperation in Human Groups With Deep Reinforcement Learning," Nat. Human Behav., **7**(10), p. 1787.
[50] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M., 2020, "Graph Neural Networks: A Review of Methods and Applications," AI Open, **1**, pp. 57–81.
[51] Pateria, S., Subagdja, B., Tan, A.-h., and Quek, C., 2021, "Hierarchical Reinforcement Learning: A Comprehensive Survey," ACM Comput. Surv. (CSUR), **54**(5), pp. 1–35.
[52] Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X., 2020, "Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks From the Topological View," Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, Paper No. 04, pp. 3438–3445.
[53] Axelrod, R., and Hamilton, W. D., 1981, "The Evolution of Cooperation," Science, **211**(4489), pp. 1390–1396.
[54] Arce, D. G., 2010, "Economics, Ethics and the Dilemma in the Prisoner's Dilemmas," Am. Econ., **55**(1), pp. 49–57.
[55] Swedberg, R., 2001, "Sociology and Game Theory: Contemporary and Historical Perspectives," Theory Soc., **30**(3), pp. 301–335.
[56] Le, S., and Boyd, R., 2007, "Evolutionary Dynamics of the Continuous Iterated Prisoner's Dilemma," J. Theor. Biol., **245**(2), pp. 258–267.
[57] Ostrom, E., 1990, *Governing the Commons: The Evolution of Institutions for Collective Action*, Cambridge University Press, Cambridge, UK.
[58] Erdos, P., and Rényi, A., 1960, "On the Evolution of Random Graphs," Publ. Math. Inst. Hung. Acad. Sci., **5**(1), pp. 17–60.
[59] Gracia-Lázaro, C., Cuesta, J. A., Sánchez, A., and Moreno, Y., 2012, "Human Behavior in Prisoner's Dilemma Experiments Suppresses Network Reciprocity," Sci. Rep., **2**(1), p. 325.
[60] Gianetto, D. A., and Heydari, B., 2015, "Network Modularity Is Essential for Evolution of Cooperation Under Uncertainty," Sci. Rep., **5**(1), p. 9340.
[61] Gianetto, D. A., and Heydari, B., 2016, "Sparse Cliques Trump Scale-Free Networks in Coordination and Competition," Sci. Rep., **6**(1), p. 21870.
[62] Fulker, Z., Forber, P., Smead, R., and Riedl, C., 2021, "Spite is Contagious in Dynamic Networks," Nat. Commun., **12**(1), p. 260.
[63] Simon, H., 1957, "A Behavioral Model of Rational Choice," *Models of Man, Social and Rational: Mathematical Essays on Rational Human Behavior in a Social Setting*, Vol. 6.1, Wiley, Hoboken, NJ, pp. 241–260.
[64] Zhang, K., Yang, Z., and Başar, T., 2021, "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms," *Handbook of Reinforcement Learning and Control*, Springer, Princeton, NJ, pp. 321–384.
[65] Puck Rombach, M., Porter, M. A., Fowler, J. H., and Mucha, P. J., 2014, "Core-Periphery Structure in Networks," SIAM J. Appl. Math., **74**(1), pp. 167–190.
[66] Borgatti, S. P., and Everett, M. G., 2000, "Models of Core/Periphery Structures," Soc. Netw., **21**(4), pp. 375–395.
[67] Ergun, O., Hopp, W. J., and Keskinocak, P., 2023, "A Structured Overview of Insights and Opportunities for Enhancing Supply Chain Resilience," IISE Trans., **55**(1), pp. 57–74.
[68] Heydari, B., Ergun, O., Dyal-Chand, R., and Bart, Y., 2023, *Reengineering the Sharing Economy: Design, Policy, and Regulation*, Cambridge University Press.
[69] Ke, L., O'Brien, D., and Heydari, B., 2021, "Airbnb and Neighborhood Crime: The Incursion of Tourists or the Erosion of Local Social Dynamics?" PLoS One, **16**(7), p. e0253315.

Downloaded from http://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/6/061006/7467114/jcise-24-1571.pdf by guest on 16 May 2025