



# OPEN The analysis of deep reinforcement learning for dynamic graphical games under artificial intelligence

Yuyang Yan<sup>1</sup>, Jiahui Li<sup>1</sup>✉ & Cristina Zaggia<sup>2</sup>

This paper explores the use of deep reinforcement learning (DRL) to enable autonomous decision-making and strategy optimization in dynamic graphical games. The proposed approach consists of several key components. First, local performance metrics are defined to reduce computational complexity and minimize information exchange among agents. Second, an online iterative algorithm is developed, leveraging Deep Neural Networks to solve dynamic graphical games with input constraints. This algorithm employs an Actor-Critic framework, where the Actor network learns optimal policies and the Critic network estimates value functions. Third, a distributed policy iteration mechanism allows each intelligent agent to make decisions based solely on local information. Finally, experimental results validate the effectiveness of the proposed method. The findings show that the DRL-based online iterative algorithm significantly improves decision accuracy and convergence speed, reduces computational complexity, and demonstrates strong performance and scalability in addressing optimal control problems in dynamic graphical intelligent games.

**Keywords** Deep reinforcement learning, Dynamic graphical games, Online iterative algorithm, Actor-critic, Artificial intelligence

## Research background and motivations

In recent years, with the rapid advancement of artificial intelligence, deep reinforcement learning (DRL) has emerged as a powerful learning paradigm and has become an essential tool for solving complex decision-making problems<sup>1–3</sup>. Dynamic graphical games, as an important model of strategic interaction, are widely applied in fields such as economics, social sciences, and military operations<sup>4,5</sup>. These games involve agents making decisions in constantly evolving environments, which places high demands on computational intelligence<sup>6</sup>. However, traditional approaches based on classical game theory and decision theory often struggle to cope with the dynamic and uncertain nature of such environments<sup>7–9</sup>. This paper focuses on zero-sum dynamic graphical games with input constraints, characterized by three key features. First, the game environment is represented as a weighted directed graph, where nodes correspond to agents and edge weights reflect interaction characteristics, such as the strength of cooperation or the cost of competition. Second, agents make decisions based on local neighborhood information, while facing constraints related to resource allocation and communication bandwidth. Third, the game aims to balance individual payoffs with the pursuit of global equilibrium in a dynamic setting. Consequently, developing effective methods for autonomous decision-making and strategy optimization using DRL has become both a significant research challenge and a growing area of interest.

In dynamic graphical games, each intelligent agent must make decisions based on the current state of the environment to achieve optimal control outcomes<sup>10</sup>. These games often involve both cooperation and competition among multiple entities, requiring agents to act within constantly changing conditions to maximize either individual gains or overall collective performance. Traditional decision-making methods often fall short in such complex scenarios, as they typically depend on static models or predefined rules that lack adaptability to dynamic environments<sup>11</sup>. The inclusion of practical constraints—such as input limitations—further complicates the problem, making it even more difficult for conventional methods to handle effectively<sup>12</sup>. Input constraints may involve resource limitations, environmental factors, or bandwidth restrictions, all of which reduce the available action space and narrow the decision-making scope of agents. In these situations, traditional approaches often struggle to find optimal solutions efficiently, as they tend to rely on exhaustive searches or comprehensive modeling of the entire state space. This leads to high computational complexity and makes them particularly vulnerable to the curse of dimensionality<sup>13</sup>. To address these challenges, there is a clear need for a

<sup>1</sup>School of Education, Guangzhou University, Guangzhou 510006, China. <sup>2</sup>Department of Philosophy, Sociology, Pedagogy and Applied Psychology (FISPPA), University of Padova, 35139 Padova, Italy. ✉email: jiahui.li@gzhu.edu.cn

novel decision-making framework that can utilize real-time environmental data and local information to achieve optimal control through learning and adaptation. DRL, as a powerful learning paradigm, offers the flexibility and scalability required to operate in complex, dynamic environments. By leveraging DRL models, intelligent agents can learn from experience and continuously improve their decision strategies, enabling them to better adapt to the uncertainties and structural complexities inherent in dynamic graphical games.

In current research on dynamic graphical games, traditional approaches—such as static game-theoretic models—can handle certain deterministic scenarios but often fail to address the complexity introduced by environmental dynamics and multi-agent interactions. The key innovation of this paper lies in the integration of local performance metrics (LPM) with a distributed Actor-Critic framework, tackling these challenges from both theoretical and methodological perspectives. While prior work has explored the use of Graph Neural Networks (GNNs) or modifications of Q-learning algorithms, their performance is often constrained by a dependence on global state information sharing, which becomes inefficient in environments with input constraints. In contrast, the proposed LPM approach defines local value functions within each agent's neighborhood, effectively decomposing the global game problem into a set of distributed optimization subproblems. This design significantly reduces the computational complexity of policy iteration. Furthermore, the online iterative algorithm developed in this paper facilitates asynchronous policy updates and localized information fusion across agents by dynamically adjusting the weighting matrix  $H$ . This mechanism overcomes the limitations of traditional Actor-Critic methods by enabling agents to optimize their policies using only local observations, thus supporting efficient cooperation in complex and dynamic settings. Additionally, the proposed method introduces an active disturbance compensation mechanism, which enhances the robustness of policy performance even under strict input constraints. These theoretical advancements and technical innovations constitute the core contributions of this paper, establishing a novel methodological foundation for real-time decision-making and optimization in dynamic graphical games.

### Research objectives

The primary objective of this paper is to investigate autonomous decision-making and strategy optimization for intelligent agents in dynamic graphical games through the application of DRL. In such games, agents must make decisions based on the current state of a continuously evolving environment to achieve optimal control outcomes. Traditional decision-making methods often fall short in handling the complexity and uncertainty inherent in these environments, particularly when practical constraints—such as input limitations—are introduced, further increasing the difficulty of the problem. To address these challenges, this paper aims to develop a DRL-based approach that enables intelligent agents to autonomously learn and optimize their decision strategies. The ultimate goal is to provide a robust and scalable solution for decision-making in dynamic graphical games, capable of adapting to environmental changes and operating effectively under resource constraints.

### Recent works

Dynamic graphical games, as a prominent research focus within the field of artificial intelligence, have attracted considerable attention and exploration from scholars in recent years. Numerous studies have examined the application of DRL to address the challenges of decision-making in such environments. For example, Chen et al.<sup>14</sup> proposed an innovative algorithm that integrates Graph Convolutional Neural Networks with Deep Q-Networks (DQN), resulting in a Graph Convolutional Q-Network designed to serve as both an information fusion module and a decision processor. Their results demonstrated that the model effectively aggregated information from sensory and connectivity sources, enabling efficient lane-changing decisions for multiple Connected Autonomous Vehicles, thereby improving safety and mobility. Similarly, Oh et al.<sup>15</sup> combined reinforcement learning with Deep Neural Networks (DNN) to present a practical approach incorporating a novel self-play curriculum and a data skipping technique. In their study, three distinct agent behaviors were created using reward shaping, and agents were co-trained to enhance performance. The data skipping technique significantly improved data efficiency and encouraged exploration within a large action space. In foundational work, Momenikorbekandi and Abbod<sup>16</sup> employed classic reinforcement learning algorithms such as Q-learning and State-Action-Reward-State'-Action to establish a basic framework for dynamic decision-making problems. However, these traditional approaches often face challenges related to state space explosion and high computational complexity when applied to dynamic graphical games. In response, many researchers have sought to enhance the expressiveness and scalability of decision models by incorporating deep learning techniques. Stember and Shalu<sup>17</sup>, for instance, proposed a DQN that integrates deep learning with reinforcement learning, offering a novel solution for handling large-scale state spaces. Despite these advancements, applying such methods to dynamic, multi-agent environments remains difficult, particularly due to the complexity of cooperation and competition among agents. To address these limitations, Qiu et al.<sup>18</sup> introduced a DQN-based method tailored to optimal control in multi-agent games, demonstrating promising performance under dynamic conditions. Meanwhile, Zhang et al.<sup>19</sup> explored the use of GNNs to model the state and action spaces in dynamic graphical games. Their findings showed that GNNs effectively capture complex relational structures, providing valuable support for intelligent agent decision-making. Additionally, Ye et al.<sup>20</sup> examined the potential of distributed policy iteration in dynamic graphical games. They proposed a distributed learning framework based on multi-agent systems, enabling each agent to optimize its decisions using only local information while engaging in interactive learning with other agents. Their research offers a new perspective on addressing the challenges of coordination and competition in dynamic multi-agent games.

In recent years, Xu et al.<sup>21</sup> proposed a data-driven path-tracking control method that enables collaborative path planning for autonomous vehicles through offline policy learning. However, their framework depends on centralized optimization based on global environmental information, which limits its effectiveness in multi-agent dynamic games where only local observations are available. This reliance on centralized data restricts the

scalability of the algorithm, particularly in input-constrained scenarios. Similarly, Nguyen et al.<sup>22</sup> conducted a comprehensive review of multi-agent DRL approaches. While their work offers valuable insights, it primarily focuses on centralized value function update mechanisms. Consequently, it does not address the integration and optimization of local information within distributed policy iteration frameworks. This gap makes it difficult to effectively handle the high-frequency state variations and decentralized decision-making requirements typical of dynamic graphical games.

Furthermore, Gu et al.<sup>23</sup> developed a metaverse evacuation training system, which innovatively applies DRL to multi-agent collaboration. However, their reward function primarily focuses on global evacuation efficiency, without considering a quantitative model for local interaction efficiency. This limitation reduces the algorithm's robustness, particularly in resource-constrained scenarios. In contrast, the LPMs proposed here defines value functions within each agent's local neighborhood, decomposing the global optimization task into distributed subproblems. This approach not only reduces computational complexity but also preserves the flexibility required for multi-agent cooperation. Table 1 summarizes the core differences between the proposed method and the approaches discussed in the literature.

Additionally, Zhao et al.<sup>24</sup> explored the integration of DRL with adaptive dynamic programming, but their study did not address the modeling of graph structures in dynamic graphical games. In contrast, this paper establishes a strong connection between graph topology and policy updates through matrix symmetry. Sadeghi et al.<sup>25</sup> proposed a supply chain finance optimization solution within a blockchain and machine learning framework; however, their model lacks adaptability to dynamic game environments. In comparison, the online iterative algorithm introduced in this paper effectively responds to environmental disturbances by dynamically adjusting the weight matrix. These comparisons highlight the limitations of existing research and underscore the innovative advancements in method design and applicability presented in this paper.

Although recent work has enhanced state representation through the use of GNNs, the efficiency of local information exchange in multi-agent games remains constrained by the computational burden of global policy updates. Existing methods often fail to address the crucial aspects of information exchange and collaboration among intelligent agents, leading to a performance decline when applied to multi-agent games. Furthermore, many approaches exhibit high computational complexity when factoring in practical constraints, such as input limitations, which undermines their effectiveness in real-world applications. Additionally, existing models often fail to adequately capture and analyze the dynamic changes in states and actions within evolving environments, leading to poor generalization of the algorithms. In light of these challenges, this paper investigates how to enhance existing methods, improve algorithm performance, and address the practical application issues in dynamic graphical games.

## Research methodology

### Definition and significance of LPM

Existing Actor-Critic methods, while capable of addressing certain dynamic decision-making problems, rely on global state information sharing for policy updates. This reliance proves challenging in multi-agent games, where local observation constraints are inherent. In response, this paper proposes a distributed Actor-Critic framework driven by LPMs, featuring three key innovations: First, by defining local value functions, the global game is decomposed into neighborhood optimization subproblems, thus avoiding the "curse of dimensionality" typically encountered in traditional methods, such as centralized Q-learning. Second, an online iterative algorithm is introduced, facilitating asynchronous policy updates for multi-agent strategies based on dynamically adjusted weight matrices. This overcomes the synchronization communication bottleneck that often hinders distributed policy iteration. Third, an active compensation mechanism for disturbance noise is incorporated, ensuring policy robustness under input constraints, and enhancing adaptability to dynamic uncertainty in comparison to global reward-based designs.

By integrating LPMs into the distributed Actor-Critic framework, this paper decomposes the global optimization problem into local decision sub-tasks, thereby overcoming the limitations of traditional approaches. To formalize the zero-sum game problem, this paper adopts the Q-function framework, where the value of the Q-function is defined with respect to the agent's state  $\alpha$ , current control strategy  $u$ , and disturbance  $w$ <sup>26–29</sup>. The function  $Q(x, u, \omega)$  replaces the traditional value function  $V(a)$ , which satisfies the Riccati equation and depends solely on the current state  $\alpha$ . The dynamic behavior of the discrete-time linear system is modeled as Eq. (1):

$$x(k+1) = Ax(k) + Bu(k) + E\omega(k) \quad (1)$$

In Eq. (1), the formulation characterizes the evolution of agent states within a dynamic graphical game. The state transition matrix  $A$  captures the intrinsic dynamics of the environment, the control input matrix  $B$  quantifies

Method	Computational complexity	Applicable scenarios	Supports input constraints	Information interaction mode
Xu et al. <sup>21</sup>	High (global optimization)	Single-agent path tracking	No	Centralized global information sharing
Nguyen et al. <sup>22</sup>	Medium	Multi-agent general framework	Partial	Hybrid (local + global)
Gu et al. <sup>23</sup>	High	Collaborative evacuation training	No	Global reward-driven
Proposed Method	Low	Dynamic graph games	Yes	Distributed local information fusion

**Table 1.** Core differences between the proposed method and related research.

the influence of control actions on state transitions, and the disturbance matrix  $B$  models uncertainties induced by external perturbations. This formal structure lays the dynamical foundation for the definition of LPMs used in subsequent analysis. The vector  $x(k)$  denotes the system's state at time step  $k$ , encompassing internal information about the agent or system. The control input vector  $u(k)$  represents the externally applied actions intended to influence the system's trajectory<sup>30</sup>. The disturbance vector  $\omega(k)$  captures external disruptions, which may stem from environmental variability, sensor noise, or other uncertainties. The matrix  $A$  is the state transition matrix, which describes how the system state evolves from time step  $k$  to  $(k + 1)$ . It encapsulates the internal structure and dynamic properties of the system. The matrix  $B$  is the control input matrix, characterizing the effect of control inputs on the system's state changes. It defines the functional relationship between the applied control actions and resulting state variations. The matrix  $E$  is the disturbance matrix, modeling the influence of external disturbances or noise on state transitions. It quantifies the extent to which uncertainties, such as environmental perturbations or sensor errors, affect the system's evolution<sup>31-33</sup>. The performance metric for discrete linear systems is formally defined in Eq. (2):

$$= \sum_{i=k}^{\infty} \alpha^T R \alpha + \sum_{k=1}^{\infty} u(k)^T Q u(k) - \sum_{i=k}^{\infty} \omega^T \lambda^2 \omega \tag{2}$$

In Eq. (2),  $R$  is a symmetric weight matrix that reflects the relative importance of the state variables, and  $W$  is a constant representing disturbance gain. The optimal performance metric corresponds to the value function<sup>34</sup>, as defined in Eq. (3):

$$V(\alpha) = \min_{\mu} \max_{\omega} J = \sum_{i=k}^{\infty} [\alpha^T R \alpha + u(k)^*{}^T Q u(k)^* - \omega^{*T} \lambda^2 \omega^*] \tag{3}$$

In Eq. (3),  $u(k)^*$  denotes the control policy that minimizes the performance cost, while  $\omega^*$  represents the worst-case disturbance. By starting from  $i = k + 1$ , the optimal Bellman equation is derived, as shown in Eqs. (4) and (5):

$$V^*(\alpha_k) = \alpha^T R \alpha + u(k)^*{}^T Q u(k)^* - \lambda \omega^{*T} \omega^* + \sum_{i=k+1}^{\infty} [\alpha^T R \alpha + u(k)^*{}^T Q u(k)^* - \omega^{*T} \lambda^2 \omega^*] = r(\alpha(k), u(k)^*, \omega^*) + V^*(\alpha_{k+1}) \tag{4}$$

$$r = \alpha^T R \alpha + u(k)^*{}^T Q u(k)^* - \lambda^2 \omega^{*T} \omega^* \tag{5}$$

Assuming that the zero-sum game has a solution, the optimal value function derived from the Bellman equation satisfies Eq. (6):

$$V^*(\alpha_k) = \alpha^T P \alpha \tag{6}$$

In Eq. (6),  $P > 0$  denotes a positive definite matrix that satisfies the discrete-time Riccati equation<sup>35,36</sup>, as expressed in Eq. (7):

$$P = A^T P A + R - \begin{bmatrix} A^T P B \\ A^T P E \end{bmatrix}^T \times \begin{bmatrix} I + B^T P B & B^T P E \\ E^T P B & E^T P E - \lambda^2 I \end{bmatrix}^{-1} \times [B^T P A \quad E^T P A] \tag{7}$$

The paper introduces a distributed strategy optimization framework based on the online iterative algorithm. The core idea is to dynamically update control strategies using local neighborhood information, facilitating efficient decision-making in dynamic graph games. The global optimization problem is transformed into an iterative update process for the local weight matrix  $H$  through block matrix inversion, as expressed in Eq. (7). This approach ensures that the policy  $L$  and disturbance compensation term  $K$  can be computed efficiently, as detailed in Eqs. (8) and (9), which represent the optimal control strategy and disturbance compensation, respectively.

$$u(k)^* = Lx(k) \tag{8}$$

$$\omega^* = Kx(k) \tag{9}$$

The conditions for  $L$  and  $K$  are derived from the system's dynamics and are given by Eqs. (10) and (11):

$$L = (I + B^T P B - B^T P E (E^T P E - \lambda^2 I)^{-1} E^T P B)^{-1} \times (B^T P E (E^T P E - \lambda^2 I)^{-1} E^T P A - B^T P A) \tag{10}$$

$$K = (E^T P E - \lambda^2 I - E^T P B (I + B^T P B)^{-1} B^T P E)^{-1} \times (E^T P B (I + B^T P B)^{-1} B^T P A - E^T P A) \tag{11}$$

The experimental section introduces the definition of the intelligent agent's Q-function<sup>37,38</sup>, which quantifies the long-term performance of the agent's policy under system dynamics and disturbances. Equation (12) presents

the Q-function in a quadratic form, incorporating the state  $x_k$ , the control strategy  $\mu_k$ , and the disturbance noise  $\omega_k$ :

$$Q^*(x_k, \mu_k, \omega_k) = [x_k^T \ \mu_k^T \ \omega_k^T] H [x_k^T \ \mu_k^T \ \omega_k^T]^T \tag{12}$$

In Eq. (12),  $H$  is a symmetric weight matrix. This Q-function serves as a surrogate for the value function<sup>39</sup>, incorporating both immediate and future performance criteria. Leveraging Eqs. (1) and (6), the Q-function is expanded as Eq. (13):

$$Q^*(x_k, \mu_k, \omega_k) = x_k^T R x_k + \mu_k^T \mu_k - \lambda^2 \omega_k^T \omega_k + x_{k+1}^T P x_{k+1} \tag{13}$$

Rewriting Eq. (13) in matrix form yields a compact quadratic representation:

$$Q^*(x_k, \mu_k, \omega_k) = \begin{bmatrix} x_k \\ \mu_k \\ \omega_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ \mu_k \\ \omega_k \end{bmatrix} \tag{14}$$

The explicit construction of the symmetric weight matrix  $H$  is provided in Eq. (15):

$$H = \begin{bmatrix} A^T P A + R & A^T P B & A^T P E \\ B^T P A & B^T P B + I & B^T P E \\ E^T P A & E^T P B & E^T P E - \lambda^2 I \end{bmatrix} \tag{15}$$

Equation (15) presents the explicit formulation of the matrix  $H$  in terms of the system matrices and the weighting matrix<sup>40-43</sup>. This formulation plays a critical role in minimizing the degree of information coupling among multiple agents in a distributed control setting. The symmetric structure of  $H$  is deliberately designed to ensure the decomposability of LPMs, which in turn facilitates independent yet coordinated optimization of control policies across agents in a multi-agent system. The equivalence between the Q-function and the value function is illustrated in Eq. (16):

$$V^*(\alpha_k) = \alpha^T P \alpha = Q^*(x_k, \mu_k, \omega_k) = \begin{bmatrix} x_k \\ \mu_k \\ \omega_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ \mu_k \\ \omega_k \end{bmatrix} \tag{16}$$

Through further derivation, matrix relationships involving<sup>44,45</sup> yield the optimal control strategy  $L$  and the optimal disturbance response strategy  $K$ , as expressed in Eqs. (17) and (18):

$$L = (H_{\mu\mu} - H_{\mu\omega} H_{\omega\omega}^{-1} H_{\omega\mu})^{-1} (H_{\mu\omega} H_{\omega\omega}^{-1} H_{\omega x} - H_{\mu x}) \tag{17}$$

$$K = (H_{\omega\omega} - H_{\omega\mu} H_{\mu\mu}^{-1} H_{\mu\omega})^{-1} (H_{\omega\mu} H_{\mu\mu}^{-1} H_{\mu x} - H_{\omega x}) \tag{18}$$

### Design of online iterative algorithm based on DNNs

This paper employs DNNs to approximate the Q-function, leveraging their powerful representation capability to enhance the learning of optimal policies. The architecture of the neural network plays a pivotal role in influencing both the performance and convergence rate of the algorithm<sup>46-48</sup>. Key design considerations for each layer of the neural network are summarized in Table 2.

To train the neural network, this paper integrates Q-learning—an online iterative reinforcement learning algorithm used to derive optimal action strategies across different states. The Q-learning-based iterative process enables the intelligent agent to learn optimal strategies in dynamic game environments, particularly in the context of distributed graphical games<sup>49</sup>. The main steps of the online iterative algorithm using DNNs are presented in Table 3:

In summary, the Q-learning algorithm updates the Q-value function iteratively using Eq. (19):

$$Q_{i+1}(x_k, \mu_k, \omega_k) = r(x_k, \mu_k, \omega_k) + V_i(x_{k+1}) \tag{19}$$

In Eq. (19),  $i$  represents the current iteration number,  $Q_{i+1}$  represents the Q-value corresponding to the current state  $x_k$ , control strategy  $\mu_k$ , and disturbance  $\omega_k$  that the  $(i + 1)$ -th iteration. This value is obtained by summing the immediate reward  $r(x_k, \mu_k, \omega_k)$  and the estimated value function  $V_i(x_{k+1})$  of the subsequent state  $x_{k+1}$ <sup>50</sup>. The value function is approximated using Eq. (20):

Neural networks	Considerations at each level
Input layer	The number of neurons equals the dimensionality of the state space plus the dimensions of the control input and the disturbance
Hidden layer	Learns feature representations of states, control strategies, and disturbances to approximate the Q-function effectively
Output layer	The number of neurons corresponds to the number of feasible control actions

**Table 2.** Factors considered in designing the neural network.

Steps	Description
Step 1:	Initialize parameters, including the weight vector, the number of iterations, and initial policy weights (all set to zero)
Step 2:	Sample multiple groups of states, compute relevant values using current weights, and calculate the average over multiple samples
Step 3:	Use the computed values to construct matrix $H$ , and update the control policy $L$ and disturbance compensation term $K$ for the $(i + 1)$ -th iteration
Step 4:	Repeat Step 2-Step 3 until the weight $h$ converges

**Table 3.** Main steps of the online iterative algorithm using DNNs.

$$\widehat{Q}(z, H) = z^T H z \quad (20)$$

In Eq. (20),  $z$  is a state vector composed of the variables  $x$ ,  $\mu$ , and  $\omega$ , and  $\widehat{Q}(z, H)$  is the estimated Q-value for a given state vector  $z$  and symmetric weight matrix  $H$ . Since  $H$  is symmetric, it can be vectorized into a column vector  $h$  consisting of its unique elements, with  $H_{ij} + H_{ji}$  used to reduce redundancy. This leads to the expression of the Q-function in vectorized form, as shown in Eq. (21):

$$\widehat{Q}_i(\bar{z}, h) = h^T \bar{z}_i \quad (21)$$

In Eq. (21),  $\bar{z}$  represents the transformed input vector composed of quadratic and interaction terms from  $z$ , such as  $[z_1^2, z_1 z_2, \dots, z_2^2, z_2 z_3, \dots, z_n^2]$ <sup>51</sup>. The objective function used to fit the Q-function is defined in Eq. (22):

$$d_i(\bar{z}, h) = r(x_k, \hat{\mu}_k, \hat{\omega}_k) + Q_i(x_{k+1}, \hat{\mu}_{k+1}, \hat{\omega}_{k+1}) \quad (22)$$

In Eq. (22),  $d_i(\bar{z}, h)$  represents the target output, composed of the immediate reward and the predicted Q-value of the next state under the current approximation  $h$  in the  $i$ -th iteration<sup>52</sup>. The corresponding square approximation error between the target and the predicted value is expressed in Eq. (23):

$$err = 1/2(d_i(\bar{z}, h) - \widehat{Q}_i(\bar{z}, h))^2 \quad (23)$$

This error quantifies the difference between the true target  $d_i$  and the current estimate  $\widehat{Q}_i$ . To minimize this error, the weight vector  $h$  is updated via gradient descent, as shown in Eq. (24):

$$h_{i+1} = (d_i(\bar{z}, h) - \widehat{Q}_i(\bar{z}, h))\bar{z}_i \quad (24)$$

In Eq. (24),  $h_{i+1}$  denotes the updated weight vector in the  $(i + 1)$ -th iteration, and  $\bar{z}_i$  the transformed state-action-disturbance vector used for training<sup>53</sup>.

### Application and algorithm details of the actor-critic framework

The Actor-Critic framework is a widely used reinforcement learning algorithm, which can be visualized in Fig. 1.

In this framework, the Actor network, responsible for policy generation, outputs the control action  $\mu_k$  based on the current state  $x_k$ . The Critic network, on the other hand, evaluates the long-term reward using the state value function  $V(x_k)$ . Both networks are updated collaboratively through gradient descent, allowing for alternating optimization of the policy and the value function.

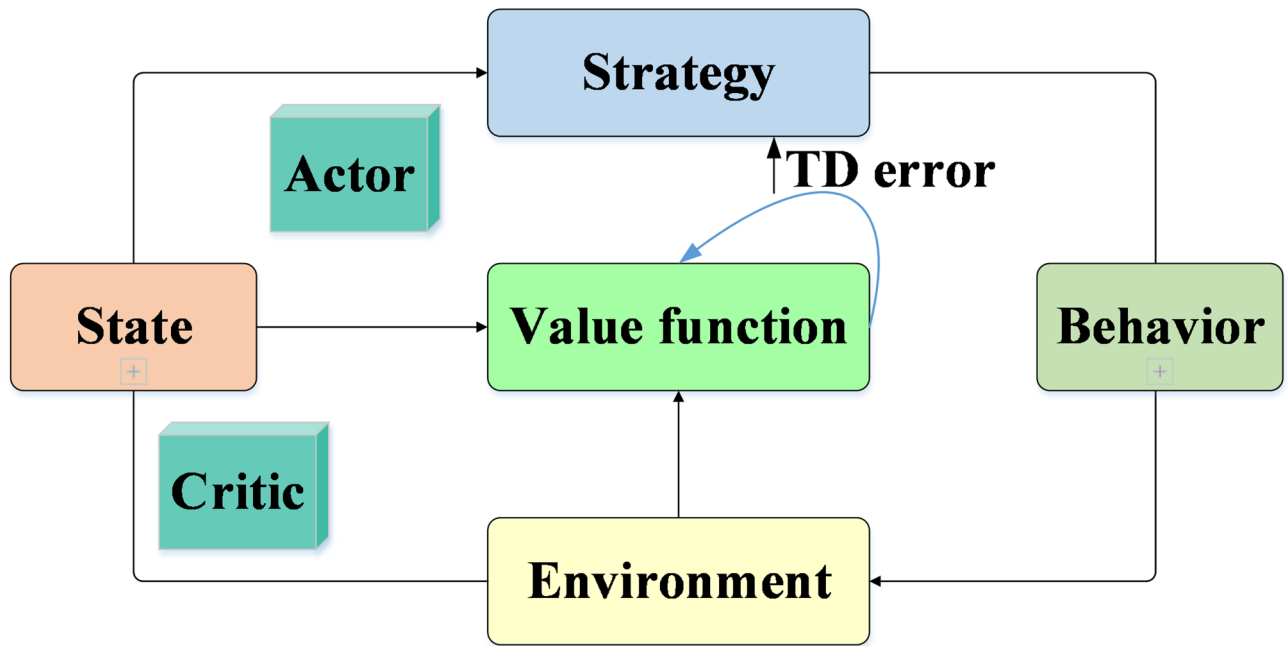
The Actor-Critic framework is composed of two main components: the Actor and the Critic. The Actor is responsible for learning the policy, which involves selecting actions based on the current state. The Critic, on the other hand, is tasked with learning the value function, evaluating the long-term value of states. These two components communicate and collaborate through DNNs, continuously optimizing and improving the overall performance of the agent.

In the algorithm details section of the Actor-Critic framework, the design and implementation of the online iterative algorithm are outlined. Key aspects include:

1. LPMs: To reduce computational complexity and minimize information exchange, LPMs are introduced. These metrics assess the agent's performance within its local state space, reducing the computational cost associated with global performance metrics. By guiding the learning of the Actor in local decision-making, these metrics enhance the efficiency of the online iteration process, improving the overall effectiveness of the algorithm.
2. Temporal Difference Method (Q-learning): The algorithm incorporates the temporal difference method from Q-learning, combining the strengths of deep learning and reinforcement learning. This approach enables better handling of complex state and action spaces. Through the use of DNNs, the functions of both the Actor and Critic are approximated, facilitating efficient learning even in high-dimensional environments.

The framework includes two core components:

1. Actor Network: The Actor is responsible for learning the policy, selecting actions based on the current state. The network's output is a probability distribution over actions, which can be deterministic or probabilistic,



**Fig. 1.** Actor-critic framework.

depending on the specific nature of the problem. The parameters of the Actor network are updated through gradient descent, maximizing the LPMs.

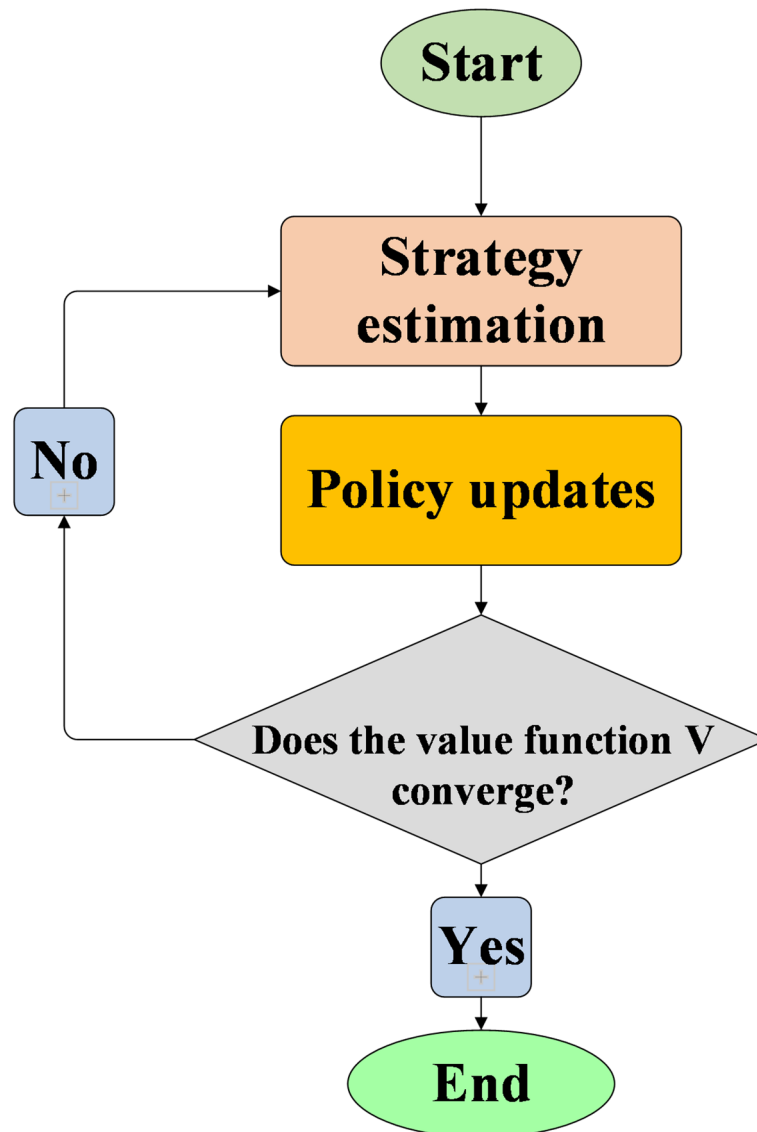
2. **Critic Network:** The Critic is responsible for learning the value function, which evaluates the long-term return of states. The Critic network outputs an estimate of the state value, representing the expected return the agent can achieve in a given state. The value function is updated by minimizing the temporal difference error, which is the difference between the actual received reward and the current estimate.

Through the continuous collaboration between the Actor and Critic networks, the Actor-Critic framework learns optimal policies and more accurately estimates the value function of states. This mutual interaction between the two networks enables the agent to refine its strategy, ultimately leading to improved overall performance.

The flow of the online iterative algorithm within the network is depicted in Fig. 2.

In Fig. 2, the algorithm begins by initializing the weight matrices and initial trajectories for all agents. These trajectories include state, control inputs, and disturbance noise. At this stage, an initial policy is generated randomly to ensure diversity among the agents during the early exploration phase. The Actor network then generates the control policy based on the current state, and its fitted value is calculated using the LPMS. The primary function of this step is to evaluate the immediate reward of the current policy, which serves as a benchmark for subsequent weight updates. The Environment-Interaction system then generates the next state based on the current policy and disturbance noise, following the state transition equation. This step simulates the interaction process between the environment and the agents within the dynamic game, providing the data required for policy optimization. Step 2 involves calculating the average value of the weight matrix  $H$  across multiple state samples, which helps avoid bias from single-sample variations. In Step 3, the updated  $H$  is used to solve for the control policy  $L$  and disturbance compensation  $K$ , reflecting the core principle of local information fusion. If the policy has not yet converged, the Critic network updates the value function parameters using temporal difference error, while the Actor network adjusts the control policy parameters through policy gradients. This collaborative mechanism between the two networks—referred to as the Actor-Critic framework—ensures the alternating optimization of the policy and evaluation of the value function. The algorithm continues this process until the change rate of the weight matrix falls below a specified threshold, or the average return stabilizes, signaling that convergence has been achieved. At this point, training is terminated. The process of the distributed online iterative algorithm is illustrated in Fig. 3.

In Fig. 3, the environment parameters, such as graph topology and input constraints, along with the initial states of the agents, are set during the game initialization. Agents update their strategies based on local neighborhood information and counteract environmental noise through disturbance compensation terms. This step embodies the core concept of distributed optimization—achieving global equilibrium through the reliance on local information. The algorithm then checks whether the current policy satisfies the value function coverage (Value Coverage), ensuring that all potential states are sufficiently explored. If the coverage condition is not met (No), an additional exploration mechanism is triggered, such as increasing the probability of random actions to encourage further exploration. If the condition is met (Yes), the algorithm proceeds to the termination check. The game continues until it reaches predefined termination conditions, such as a maximum number of steps or a victory/loss determination. Once these conditions are met, the current round concludes, and the performance data of the strategies is recorded for further analysis.



**Fig. 2.** Flow of the online iterative algorithm in the network.

## Experimental design and performance evaluation

### Datasets collection

This paper uses a Web graphical dataset, which includes networks extracted from 62 games. The networks are weighted, directed, and temporal. In this dataset, each node represents a participant. Directed edges are established from node  $u$  to node  $v$  every 1/3 s, with the edge weights reflecting the probability that participant  $u$  is looking at participant  $v$  or a laptop.

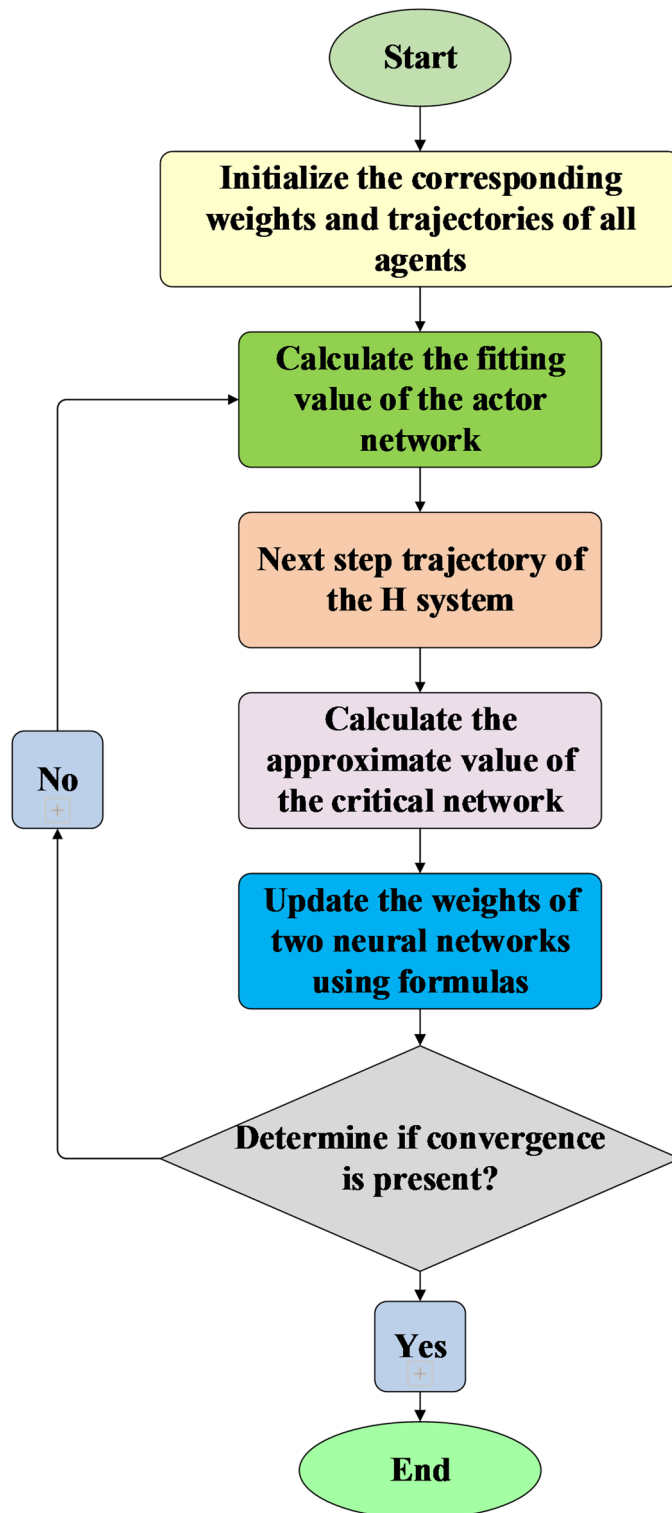
### Experimental environment

The experimental environment is configured as follows:

1. Processor: Intel Core i9-10900 K, featuring a 3.7 GHz frequency and 20 cores
2. Memory: 64 GB DDR4
3. Storage: 1 TB NVMe SSD, enabling fast data read and write operations
4. Operating System: Ubuntu 20.04 LTS, ensuring a stable environment for development and execution
5. GPU: NVIDIA GeForce RTX 3080, with 10 GB GDDR6X VRAM, accelerating deep learning model training and effectively managing large-scale parallel computing tasks.

### Parameters setting

The experimental setup based on the Online Iterative Algorithm, as outlined in the previous sections, is designed to validate the effectiveness of local policy optimization under input constraints. The neural network structure (Actor network: 2 layers with 64 neurons; Critic network: 2 layers with 128 neurons) is grounded in literature



**Fig. 3.** Flow of dynamic graphical intelligent games in actor-critic DRL.

and experimental validation. Research by Zhou and Zhang<sup>54</sup> shows that medium-sized hidden layers (64–128 neurons) offer the best balance between model expressiveness and computational efficiency, making them particularly suitable for real-time decision-making in dynamic games<sup>54</sup>. Pre-experimentation compared different hidden layer configurations (e.g., 32/64/128 neurons) in terms of convergence speed and final performance. The results revealed that the 64-neuron structure provided the optimal trade-off between training steps (350 K steps) and average return (81.40%). The Critic network with 128 neurons, on the other hand, was necessary to handle

higher-dimensional value functions (such as the state value matrix in Eq. (6)), requiring a larger capacity to avoid underfitting. The experimental parameter settings for this paper are shown in Table 4:

The parameter choices strike a balance between algorithm efficiency and policy exploration. The batch size for experience replay was set to 64, determined through pre-experiments. Smaller batches (e.g., 32) speed up individual training steps but increase gradient estimate variance. Larger batches (e.g., 128) stabilize the learning process but significantly increase memory usage. The 64 batch size provides the best trade-off, in line with the parameter design for the material optimization task. The exploration rate ( $\epsilon$ ) follows a linear decay mechanism. The initial value of  $\epsilon$  is set to 0.8 to ensure sufficient random exploration, decreasing by 0.01 per training round until it stabilizes at  $\epsilon = 0.1$ . This design avoids the early convergence problem often caused by traditional exponential decay, ensuring the algorithm retains some exploration capacity in later training phases to handle dynamic environmental disturbances.

### Performance evaluation

The experimental results for various neural network structures are presented in Fig. 4. As training steps increase, agent performance improves across all tested configurations; however, notable differences emerge based on network size. Under the 64-neuron configuration, agent performance remains relatively stable, with the average return increasing from 25.69% to a peak of 65.88% after 350 training steps. The 128-neuron structure exhibits some initial performance fluctuations but ultimately achieves the highest average return of 70.29% due to its enhanced capacity to capture complex policy patterns. In contrast, both the 256- and 384-neuron configurations show strong early-stage performance. However, the 256-neuron network slightly underperforms in the later stages compared to other configurations. Although the 384-neuron structure achieves good early performance with fewer training steps, its learning curve flattens as training progresses, indicating a slower convergence rate. Overall, the 128-neuron configuration demonstrates superior long-term performance, balancing model expressiveness with training efficiency. The 384-neuron structure, while initially effective, likely suffers from parameter redundancy that hinders convergence speed. These findings empirically support the selection of appropriately scaled network architectures that optimize both computational efficiency and learning effectiveness.

Figure 5 compares the test accuracy of various models. The proposed model achieves the best performance, with a maximum accuracy of 88.20% and an average accuracy of 81.40%, reflecting both high precision and robustness in test scenarios. In contrast, the DQN achieves 79.20% maximum accuracy and 78.00% average accuracy, while the Double Deep Q-Network (DDQN) reaches a higher peak accuracy of 83.00% but a lower average accuracy of 68.80%. These results highlight the superior generalization capability of the proposed method.

The impact of various reinforcement learning enhancements is illustrated in Fig. 6. Among the applied techniques, prioritized experience replay yields the highest gains, boosting maximum accuracy to 91.20% and average accuracy to 84.10%. Experience replay alone also provides a significant improvement, achieving a maximum accuracy of 90.50% and an average of 83.20%. In contrast, techniques such as DDQN and multi-step returns offer more modest enhancements, suggesting that memory-based sampling mechanisms contribute most significantly to the model's learning effectiveness.

Table 5 presents the key performance metrics of several models evaluated in dynamic graph game scenarios. As shown, the complete model proposed in this paper outperforms traditional DQN and DDQN models in both computational efficiency (9.6 s per round) and stability (average accuracy of 81.40%). Although the prioritized experience replay technique achieves the highest peak accuracy (91.20%) among the individual enhancements, it falls slightly behind the complete model in terms of convergence speed (320 k steps) and computational time (10.8 s per round). These results highlight the advantages of integrating LPMs within a distributed Actor-Critic framework, which allows the proposed method to significantly reduce computational overhead without sacrificing accuracy. For example, compared to DDQN, the proposed model reduces computation time by 32.4%

Parameter	Value
Number of agents	10
State space dimensions	4
Action space dimensions	2
Number of training rounds	500
Steps per round	50
Exploration rate	Initial value $\epsilon$ : 0.8, decreases by 0.01 per round, minimum $\epsilon_{min}$ : 0.1
Reward discount factor	0.9
Learning rate	0.001
Neural network structure	Actor: two hidden layers, 64 neurons each; Critic: two hidden layers, 128 neurons each
Optimizer	Adam
Loss function	mean square error
Training batch size	64
The initial state of the agent	Randomly generated within the state space
Interference noise	Gaussian white noise with mean 0 and standard deviation 0.1

**Table 4.** experimental parameter settings.

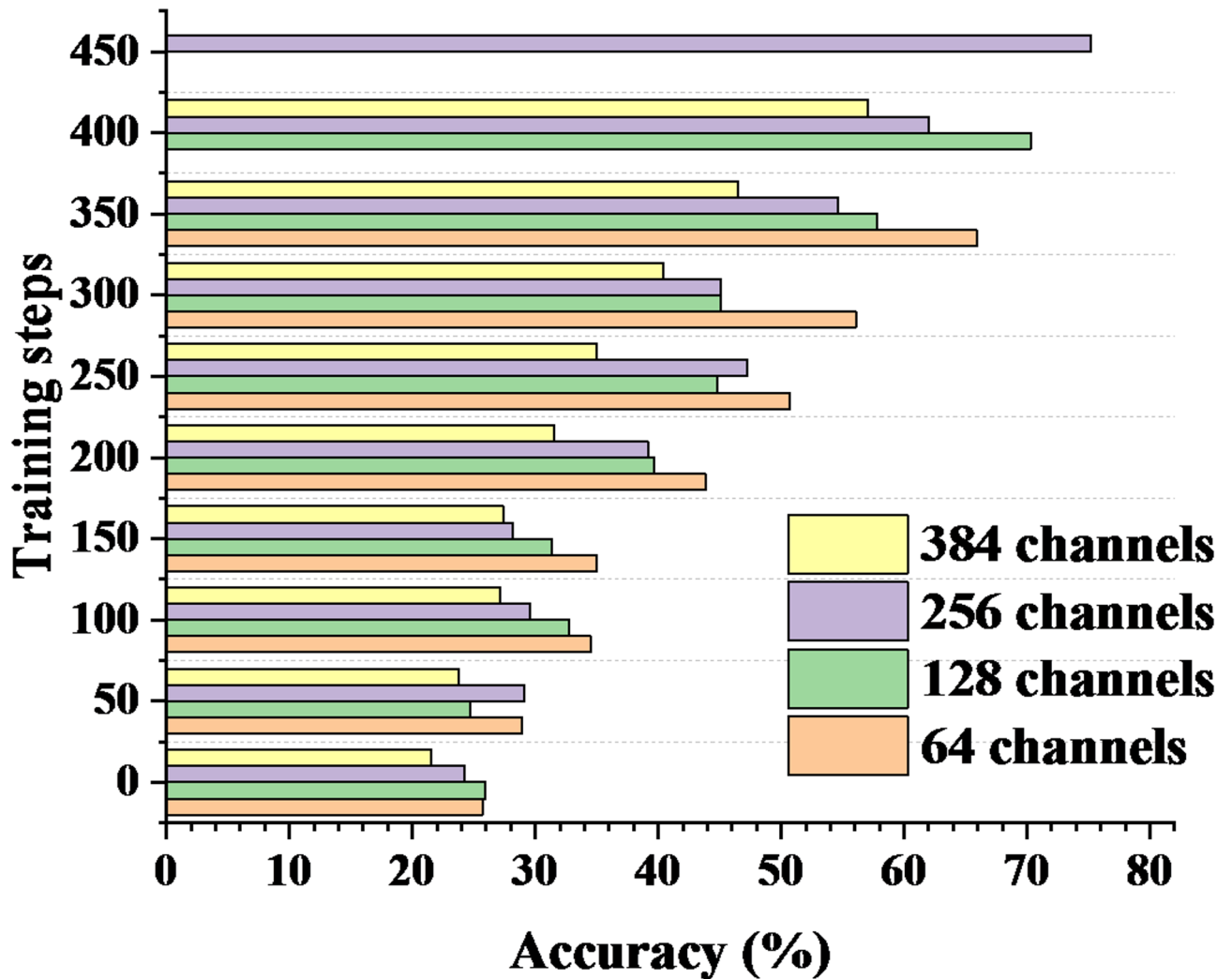


Fig. 4. Experimental results for different neural network structures.

and improves average accuracy by 12.6 percentage points—an advantage that is particularly important in large-scale dynamic game applications, such as multi-Unmanned Aerial Vehicle cooperative control. Additionally, the model achieves convergence in 350 k steps, which is 16.7% faster than the DQN baseline (420 k steps), demonstrating the effectiveness of the online iterative algorithm in accelerating policy optimization. These comparative results not only validate the practical efficacy of the proposed approach but also offer empirical guidance for algorithm selection in real-world engineering deployments.

### Discussion

The distributed Actor-Critic framework proposed in this study, driven by LPMs, is experimentally validated in a zero-sum game setting. However, its theoretical foundation suggests strong potential for extension to non-zero-sum scenarios. For example, in cooperative dynamic graph games, the reward function can be adapted by incorporating team-based reward components, transforming competitive payoffs into shared gains. Additionally, the disturbance compensation mechanism remains relevant in hybrid game environments that blend adversarial and cooperative dynamics, where disturbance noise can be redefined as environmental uncertainty or behavioral perturbations from non-hostile agents. Recent research, such as the study by Xue et al.<sup>55</sup> on complex social systems, has shown that similar distributed architectures can facilitate multi-objective optimization through reward shaping, further validating the generalizability of the proposed method. While the current experiments primarily assess the model's performance in zero-sum contexts, future work will focus on extending the framework to mixed-game environments that involve both cooperative and competitive interactions.

The DRL model proposed in this paper demonstrates significant performance improvements in dynamic graphical intelligent gaming. These results align with those of Hu et al.<sup>56</sup>, who highlighted the advantages of DRL in addressing complex environments, particularly in dynamic decision-making problems. By utilizing the Actor-Critic framework, the model in this paper successfully achieves autonomous decision-making and strategy optimization for agents, effectively addressing optimal control challenges in dynamic graphical games. Furthermore, the experimental results reveal that the choice of neural network structures, particularly the number

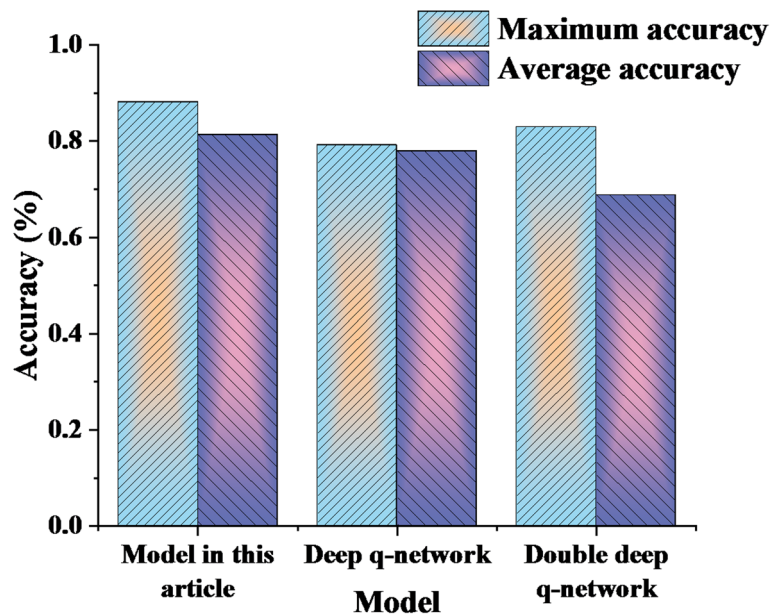


Fig. 5. Test accuracy results for different models.

of channels, has a significant impact on model performance. This finding is consistent with the conclusions of Nafisah et al.<sup>57</sup>, who emphasized the critical role of neural network architecture in deep learning models. Additionally, by comparing the accuracy results across different models, the proposed model outperforms traditional DQN and DDQN models in both maximum and average accuracy. This further supports Nafisah et al.'s work, which underscores the importance of improvement techniques in enhancing the performance of DRL models.

## Conclusion

### Research contribution

This paper introduces a novel approach that combines LPMs with a distributed Actor-Critic framework, effectively addressing the bottlenecks of multi-agent cooperation in dynamic graph games. Traditional methods, such as global Q-learning or centralized policy optimization, struggle to adapt to input constraints in dynamic environments due to their dependence on complete state information. In contrast, by defining local value functions and employing an online iterative algorithm, this paper allows agents to autonomously optimize their decisions using only local neighborhood information. Experimental results not only validate the theoretical design by demonstrating advantages in convergence speed and computational efficiency, but also underscore the method's potential for real-time decision-making applications, such as vehicle coordination in intelligent transportation systems and dynamic path planning for drone formations.

### Future works and research limitations

Future research can be pursued in three key directions: First, exploring the generalization of non-zero-sum games by introducing cooperative reward mechanisms, thereby extending the method's applicability to hybrid cooperation-competition scenarios. Second, optimizing algorithm efficiency by designing lightweight network structures, such as incorporating graph attention mechanisms, to handle high-dimensional state spaces and support more complex industrial control problems. Third, enhancing the model's adaptability to different environments through the application of transfer learning in heterogeneous agent games. These research directions not only build on the theoretical innovations presented in this paper but also offer practical technical pathways for the engineering implementation of dynamic graph games.

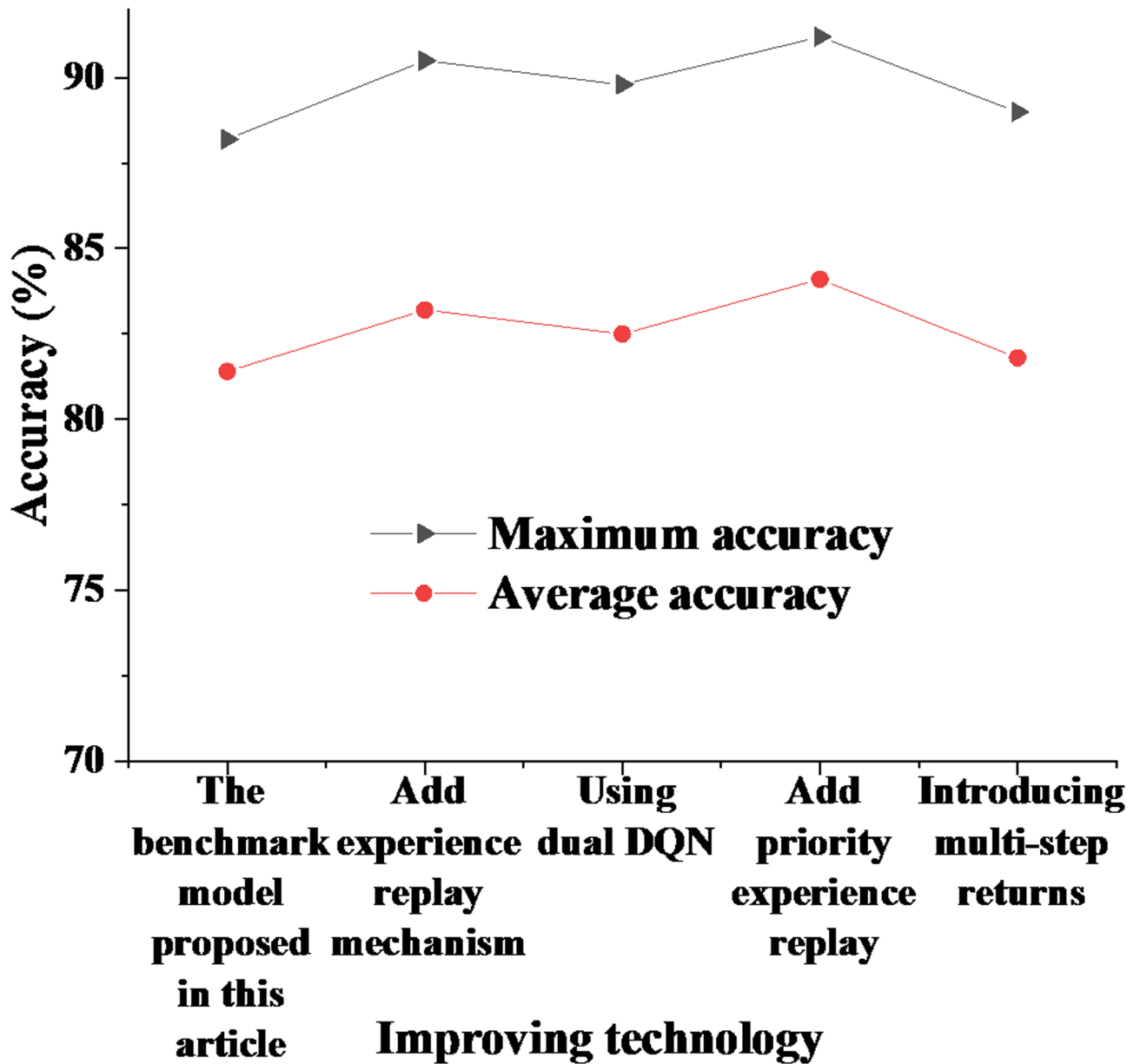


Fig. 6. Results of different enhancement techniques on the performance of the reinforcement learning model proposed in this paper.

Model	Peak accuracy (%)	Average accuracy (%)	Convergence steps (k)	Computation time (s/round)
Model	79.20	78.00	420	12.5
DQN (Ref <sup>17</sup> )	83.00	68.80	380	14.2
DDQN (Ref <sup>18</sup> )	91.20	84.10	320	10.8
Prioritized Experience Replay	90.50	83.20	340	11.3
Improved Experience Replay	88.20	81.40	350	9.6

Table 5. Performance comparison of different models in dynamic graph games.

**Data availability**

The datasets used and/or analyzed during the current study are available from the corresponding author Jiahui Li on reasonable request via e-mail [jiahui.li@gzhu.edu.cn](mailto:jiahui.li@gzhu.edu.cn).

Received: 1 April 2025; Accepted: 2 June 2025

Published online: 02 July 2025

## References

- Wang, L. et al. Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing. *IEEE Trans. Mob. Comput.* **21**(10), 3536–3550 (2021).
- Oroojlooyjadid, A. et al. A deep q-network for the beer game: Deep reinforcement learning for inventory optimization. *Manuf. Serv. Oper. Manag.* **24**(1), 285–304 (2022).
- Xie, Y. et al. Virtualized network function forwarding graph placing in SDN and NFV-enabled IoT networks: A graph neural network assisted deep reinforcement learning method. *IEEE Trans. Netw. Serv. Manag.* **19**(1), 524–537 (2021).
- Du, X. et al. Multi-agent reinforcement learning for dynamic resource management in 6G in-X subnetworks. *IEEE Trans. Wirel. Commun.* **22**(3), 1900–1914 (2022).
- Song, W. et al. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Trans. Ind. Inf.* **19**(2), 1600–1610 (2022).
- Liu, J. S. et al. Joint beamforming, power allocation, and splitting control for SWIPT-enabled IoT networks with deep reinforcement learning and game theory. *Sensors* **22**(6), 2328 (2022).
- Ji, J., Nie, B. & Gao, Y. DAGA: Dynamics aware reinforcement learning with graph-based rapid adaptation. *IEEE Robot. Autom. Lett.* **8**(4), 2189–2196 (2023).
- Yang, Z., Bi, L. & Jiao, X. Combining reinforcement learning algorithms with graph neural networks to solve dynamic job shop scheduling problems. *Processes* **11**(5), 1571 (2023).
- Tong, J. et al. GLDAP: Global dynamic action persistence adaptation for deep reinforcement learning. *ACM Trans. Auton. Adapt. Syst.* **18**(2), 1–18 (2023).
- Souchleris, K., Sidiropoulos, G. K. & Papakostas, G. A. Reinforcement learning in game industry—Review, prospects and challenges. *Appl. Sci.* **13**(4), 2443 (2023).
- Bougueroua, S. et al. Algorithmic graph theory, reinforcement learning and game theory in MD simulations: From 3D structures to topological 2D-molecular graphs (2D-MolGraphs) and vice versa. *Molecules* **28**(7), 2892 (2023).
- Chen, W. et al. A multi-agent deep reinforcement learning-based popular content distribution scheme in vehicular networks. *Entropy* **25**(5), 792 (2023).
- Rani, G. et al. A deep reinforcement learning technique for bug detection in video games. *Int. J. Inf. Technol.* **15**(1), 355–367 (2023).
- Chen, S. et al. Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles. *Comput. Aided Civ. Infrastruct. Eng.* **36**(7), 838–857 (2021).
- Oh, I. et al. Creating pro-level AI for a real-time fighting game using deep reinforcement learning. *IEEE Trans. Games* **14**(2), 212–220 (2021).
- Momenikorbekandi, A. & Abbod, M. Intelligent scheduling based on reinforcement learning approaches: Applying advanced Q-learning and state-action-reward-state-action reinforcement learning models for the optimisation of job shop scheduling problems. *Electronics* **12**(23), 4752 (2023).
- Stember, J. N. & Shalu, H. Reinforcement learning using Deep Q networks and Q learning accurately localizes brain tumors on MRI with very small training sets. *BMC Med. Imaging* **22**(1), 224 (2022).
- Qiu, S. et al. Multi-agent optimal control for central chiller plants using reinforcement learning and game theory. *Systems* **11**(3), 136 (2023).
- Zhang, M. et al. Dynamic graph neural networks for sequential recommendation. *IEEE Trans. Knowl. Data Eng.* **35**(5), 4741–4753 (2022).
- Ye, M. et al. Distributed Nash equilibrium seeking in games with partial decision information: a survey. *Proc. IEEE* **111**(2), 140–157 (2023).
- Xu, Y., Wu, Z. G. & Pan, Y. J. Cooperative path following control in autonomous vehicles graphical games: A data-based off-policy learning approach. *IEEE Trans. Intell. Transp. Syst.* **25**(8), 9364–9374 (2024).
- Nguyen, T. T., Nguyen, N. D. & Nahavandi, S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Trans. Cybern.* **50**(9), 3826–3839 (2020).
- Gu, J. et al. A metaverse-based teaching building evacuation training system with deep reinforcement learning. *IEEE Trans. Syst. Man Cybern. Syst.* **53**(4), 2209–2219 (2023).
- Zhao, D. et al. Special issue on deep reinforcement learning and adaptive dynamic programming. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2038–2041 (2018).
- Sadeghi, R., Hajian, A. & Rabiee, M. Blockchain and machine learning framework for financial performance in pharmaceutical supply chains. *Advancement in Business Analytics Tools for Higher Financial Performance*. IGI Global 112–128 (2023).
- Gong, Z., Xu, Y. & Luo, D. UAV cooperative air combat maneuvering confrontation based on multi-agent reinforcement learning. *Unmanned Syst.* **11**(03), 273–286 (2023).
- Han, D. et al. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors* **23**(7), 3762 (2023).
- Xu, C. & Song, W. Intelligent task allocation for mobile crowdsensing with graph attention network and deep reinforcement learning. *IEEE Trans. Netw. Sci. Eng.* **10**(2), 1032–1048 (2023).
- Naseer, F., Khan, M. N. & Altalbe, A. Intelligent time delay control of telepresence robots using novel deep reinforcement learning algorithm to interact with patients. *Appl. Sci.* **13**(4), 2462 (2023).
- Feng, T. et al. Contact tracing and epidemic intervention via deep reinforcement learning. *ACM Trans. Knowl. Discov. Data* **17**(3), 1–24 (2023).
- Guo, D. et al. A proactive eavesdropping game in MIMO systems based on multiagent deep reinforcement learning. *IEEE Trans. Wirel. Commun.* **21**(11), 8889–8904 (2022).
- Liu, X. Dynamic coupon targeting using batch deep reinforcement learning: An application to livestream shopping. *Mark. Sci.* **42**(4), 637–658 (2023).
- Gao, M. et al. Intelligent pursuit-evasion game based on deep reinforcement learning for hypersonic vehicles. *Aerospace* **10**(1), 86 (2023).
- Liu, R., Piplani, R. & Toro, C. Deep reinforcement learning for dynamic scheduling of a flexible job shop. *Int. J. Prod. Res.* **60**(13), 4049–4069 (2022).
- Frattolillo, F., Brunori, D. & Iocchi, L. Scalable and cooperative deep reinforcement learning approaches for multi-UAV systems: A systematic review. *Drones* **7**(4), 236 (2023).
- Dizaji, S. H. S., Patil, K. & Avrachenkov, K. Influence maximization in dynamic networks using reinforcement learning. *SN Comput. Sci.* **5**(1), 169 (2024).
- Yan, H. et al. Distributed multiagent deep reinforcement learning for multiline dynamic bus timetable optimization. *IEEE Trans. Ind. Inf.* **19**(1), 469–479 (2022).
- Chen, S., Yang, Y. & Su, R. Deep reinforcement learning with emergent communication for coalitional negotiation games. *Math. Biosci. Eng.* **19**(5), 4592–4609 (2022).
- Wang, W. et al. Research and challenges of reinforcement learning in cyber defense decision-making for intranet security. *Algorithms* **15**(4), 134 (2022).

40. Lu, Y., Ruan, X. & Huang, J. Deep reinforcement learning based on social spatial–temporal graph convolution network for crowd navigation. *Machines* **10**(8), 703 (2022).
41. Zhao, J. et al. Deep reinforcement learning-based model-free on-line dynamic multi-microgrid formation to enhance resilience. *IEEE Trans. Smart Grid* **13**(4), 2557–2567 (2022).
42. Benaddi, H. et al. Robust enhancement of intrusion detection systems using deep reinforcement learning and stochastic game. *IEEE Trans. Veh. Technol.* **71**(10), 11089–11102 (2022).
43. Orr, J. & Dutta, A. Multi-agent deep reinforcement learning for multi-robot applications: a survey. *Sensors* **23**(7), 3625 (2023).
44. Yuan, S. et al. Graph convolutional reinforcement learning for resource allocation in hybrid overlay–underlay cognitive radio network with network slicing. *IET Commun.* **17**(2), 215–227 (2023).
45. Kaufmann, E. et al. Champion-level drone racing using deep reinforcement learning. *Nature* **620**(7976), 982–987 (2023).
46. Theodoropoulos, T. et al. Graph neural networks for representing multivariate resource usage: A multiplayer mobile gaming case-study. *Int. J. Inf. Manag. Data Insights* **3**(1), 100158 (2023).
47. Guan, Z., Wang, Y. & He, M. Deep reinforcement learning-based spectrum allocation algorithm in Internet of vehicles discriminating services. *Appl. Sci.* **12**(3), 1764 (2022).
48. Wang, Q., He, Y. & Tang, C. Mastering construction heuristics with self-play deep reinforcement learning. *Neural Comput. Appl.* **35**(6), 4723–4738 (2023).
49. Hippalgaonkar, K. et al. Knowledge-integrated machine learning for materials: Lessons from gaming and robotics. *Nat. Rev. Mater.* **8**(4), 241–260 (2023).
50. Hazra, T. & Anjaria, K. Applications of game theory in deep learning: A survey. *Multimedia Tools Appl.* **81**(6), 8963–8994 (2022).
51. Zhu, T. et al. Optimal strategy of the simultaneous dice game Pig for multiplayer: When reinforcement learning meets game theory. *Sci. Rep.* **13**(1), 8142 (2023).
52. Wurman, P. R. et al. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature* **602**(7896), 223–228 (2022).
53. Yan, L. et al. A cooperative charging control strategy for electric vehicles based on multiagent deep reinforcement learning. *IEEE Trans. Ind. Inf.* **18**(12), 8765–8775 (2022).
54. Zhou, Z., Zhang, H. & Effatparvar, M. Improved sports image classification using deep neural network and novel tuna swarm optimization. *Sci. Rep.* **14**(1), 14121 (2024).
55. Xue, X. et al. Computational experiments for complex social systems: Integrated design of experiment system. *IEEE/CAA J. Autom. Sin.* **11**(5), 1175–1189 (2024).
56. Hu, J. et al. Autonomous maneuver decision making of dual-UAV cooperative air combat based on deep reinforcement learning. *Electronics* **11**(3), 467 (2022).
57. Nafisah, S. I. & Muhammad, G. Tuberculosis detection in chest radiograph using convolutional neural network architecture and explainable artificial intelligence. *Neural Comput. Appl.* **36**(1), 111–131 (2024).

## Acknowledgements

This work was supported by the Youth Fund for Humanities and Social Sciences Research, Ministry of Education of the People's Republic of China (No. 24YJC880067).

## Author contributions

Yuyang Yan: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation Jiahui Li: writing—review and editing, visualization, supervision, project administration, funding acquisition Cristina Zaggia: methodology, software, validation, formal analysis.

## Declarations

### Competing interests

The authors declare no competing interests.

### Ethics approval

The studies involving human participants were reviewed and approved by School of Education, Guangzhou University Ethics Committee (Approval Number: 2022.02510032). The participants provided their written informed consent to participate in this study. All methods were performed in accordance with relevant guidelines and regulations.

### Additional information

**Correspondence** and requests for materials should be addressed to J.L.

**Reprints and permissions information** is available at [www.nature.com/reprints](http://www.nature.com/reprints).

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025