Non-Cooperative Multi-Agent Reinforcement Learning Exploiting Population Dynamics

Junling Li, *Member, IEEE*, Hao Zhang, Shuqi Ke, Jianwei Huang, *Fellow, IEEE*, Nan Chen, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—Non-cooperative multi-agent reinforcement learning (MARL) faces significant challenges due to non-stationarity and non-unique learning goals. While equilibrium-based analysis frameworks effectively address these challenges, existing approaches suffer from high computational complexity as the number of agents increases. To overcome this limitation, we propose a population game-based Q-learning (Pop-Q) algorithm that computes Nash equilibrium (NE) policies through efficient population dynamics. Our approach represents population evolution using ordinary differential equations (ODEs) and introduces two key mechanisms to reduce the complexity of solving these ODEs. By adjusting the number of iterations in population dynamics, our algorithm enables a controllable trade-off between computational complexity and equilibrium accuracy. Experimental results demonstrate that Pop-Q achieves competitive performance in two-agent settings and superior performance in three-agent environments compared to existing equilibrium-based MARL algorithms. The proposed algorithm has significant potential applications in modern systems requiring decentralized coordination, including intelligent traffic systems, warehouse automation, and autonomous aerial vehicle (AAV) swarm-aided communication networks.**

Received 24 January 2025; revised 1 June 2025; accepted 27 June 2025. Date of publication 7 July 2025; date of current version 21 November 2025. The work of Hao Zhang and Nan Chen were supported by the General Research Fund Scheme (GRF) of Hong Kong Research Grant Council under Grant 14211023 and Grant 14205824. This work was supported in part by the National Natural Science Foundation of China under Grant 62271434 and Grant 62301151, in part by the Shenzhen Key Lab of Crowd Intelligence Empowered Low-Carbon Energy Network under Grant ZDSYS20220606100601002, in part by Shenzhen Stability Science Program 2023, in part by the Shenzhen Institute of Artificial Intelligence and Robotics for Society, and in part by the Longgang District Shenzhen's "Ten Action Plan" for Supporting Innovation Projects under Grant LGKCSDPT2024002. Recommended for acceptance by Dr. Y. Wu. *(Corresponding author: Jianwei Huang.)*

Junling Li is with the National Mobile Communications Research Laboratory, School of Information Science and Engineering, Southeast University, Nanjing 210096, China (e-mail: junlingli@seu.edu.cn).

Hao Zhang and Nan Chen are with the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: frank_zhang110407@outlook.com; nchen@se.cuhk.edu.hk).

Shuqi Ke is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: shuqik@andrew.cmu.edu).

Jianwei Huang is with the Shenzhen Key Laboratory of Crowd Intelligence Empowered Low-Carbon Energy Network, School of Science and Engineering, Shenzhen Institute of Artificial Intelligence and Robotics for Society, The Chinese University of Hong Kong, Shenzhen 518172, China, and also with the CSIJRI Joint Research Centre on Smart Energy Storage, The Chinese University of Hong Kong, Shenzhen 518172, China (e-mail: jianweihuang@cuhk.edu.cn).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/TNSE.2025.3586602

***Index Terms*—Evolutionary dynamics, game theory, multi-agent reinforcement learning (MARL), Nash equilibrium (NE), population games.**

I. Introduction

MANY real-world problems, such as multi-robot control, autonomous aerial vehicle (AAV) swarm, and autonomous driving cars, involve complex (cooperative or non-cooperative) interactions among agents and require decentralized decision-making and coordination. For example, in traffic systems, the coordination of multiple self-driving cars and traffic light optimization is essential to achieve reduced traffic congestion and improved traffic flow efficiency [1]. In multi-AAV-aided communication systems, trajectory planning and resource allocation of multiple AAVs are imperative to achieve faster task completion and improved resource utilization [2], [3], [4], [5], [6]. Multi-agent reinforcement learning (MARL) represents the algorithmic paradigm to understand how multiple agents can learn their optimal policies in such complicated scenarios, e.g., [7], [8]. In the non-cooperative MARL paradigm, several agents maximize their individual rewards respectively, by interacting with the environment and each other [9], [10], [11], [12].

Non-cooperative MARL induces two critical challenges for performance analysis: 1) the non-stationary issue, and 2) the non-unique learning goals. In the non-cooperative MARL paradigm, agents learn simultaneously and their joint actions change the environment. Hence, in order to learn his optimal policy, an agent needs to understand not only the environment but also other agents' policies that change the environment. As a result, we cannot directly apply the traditional single-agent RL to multi-agent scenarios as there is no convergence guarantee of such an approach [13]. Another challenge in non-cooperative MARL is how to deal with multiple (possibly conflicting) learning goals [14], [15]. In the single-agent RL environment, a single agent wants to maximize its own long-term reward. In contrast, in the non-cooperative MARL environment, each agent wants to maximize its own long-term reward. Therefore, how to obtain a policy where no agent has the incentive to change his strategy (i.e., the equilibrium policy) during the entire course of the learning process remains a significant challenge.

One potential solution is to leverage the equilibrium concept in game theory to enforce the agents to play equilibrium policies (known as the equilibrium-based MARL framework) [16], [17], [18], [19]. This framework models the strategic interactions

among agents over time as a stochastic game. At each state of the stochastic game (namely, each state of the system), the framework computes the equilibrium for the corresponding normal-form game and updates the Q-functions of the agents according to the chosen equilibrium. If such an approach converges, all the agents play equilibrium policies at each state derived from the learned Q-functions.

Most existing equilibrium-based MARL algorithms compute the exact equilibria at each state with high computational complexity. For example, computing the Nash equilibrium (NE) using classical algorithms can be polynomial parity arguments on directed graphs (PPAD)-hard [20]. The complexity of computing the correlated equilibrium (CE) using linear programming increases exponentially with the number of agents [18]. Consequently, the existing equilibrium-based MARL approaches are not directly applicable to a multi-agent system with many agents. This motivates us to ask the following key question in this paper:

Key Question: How to achieve the best performance-complexity trade-off in the equilibrium-based MARL framework?

Our contributions: This paper provides a possible answer to the above question by introducing the population game theory [21] into the equilibrium-based MARL framework. We first model the MARL problem as a stochastic game, similar to other equilibrium-based MARL frameworks. Then, we formulate the normal-form game at each state in the non-cooperative MARL as a population game, where we expand an agent into a population of "atom-agents". We define the state of such a population as the probability distribution of the "atom-agents" choosing among all the available pure strategies of the corresponding agent. The population game provides a theoretical guarantee that the equilibrium policy can be found by solving the ordinary differential equation (ODE) [22]. Under this formulation, we can efficiently find the equilibrium policies of the agents by solving a set of ODEs. We further devise two mechanisms to control the trade-off between complexity and accuracy of the NE. Through benchmarking experiments, we empirically demonstrate that the proposed MARL framework (called the Pop-Q algorithm) exploiting population dynamics achieves a better performance-complexity trade-off compared with existing equilibrium-based MARL algorithms.

Our main contributions are summarized as follows:

- *Addressing the high complexity issue of equilibrium computation:* Existing equilibrium-based MARL approaches compute exact equilibria at each state to update the agents' Q-values. The computation becomes intractable when the number of agents becomes large. To the best of our knowledge, this paper is the first one introducing the population game theory into equilibrium-based MARL to deal with the high complexity issue of equilibrium computation.
- *A novel population game formulation:* We formulate the normal-form game at each stage game in MARL as a population game. In this game, we expand each agent in MARL into a group of "atom-agents" (called a "population") and use a population's state to represent the corresponding agent's mixed-strategy. Under the population game formulation at each state, we can find the agents' equilibrium policies by solving a set of ODEs.
- *Efficient equilibrium computation framework:* To improve the efficiency of equilibrium computation, we design a threshold-based mechanism to tune the number of iterations in solving the ODE. We further propose a heuristic mechanism to inherit the precomputed equilibrium as the initial condition for solving the ODE corresponding to a state visited for the second time. These mechanisms allow us to control the convergence speed of solving the ODE and to obtain approximate equilibria with a lower computation complexity.
- *Performance evaluation:* We evaluate our proposed algorithm via extensive numerical experiments, benchmarking non-cooperative general-sum games. We demonstrate that the proposed Pop-Q algorithm achieves better performance in a three-agent environment than existing equilibrium-based MARL algorithms.

The remainder of the paper is organized as follows. Section II reviews the related works. Section III introduces the MARL model. Section IV describes the proposed Pop-Q algorithm in details. Section V presents the experiment settings and results, followed by a conclusion in Section VI. Table I lists the key notations used in this paper.

II. Related Works

In the past decade, numerous MARL algorithms have been developed to improve the performance of each agent and the whole multi-agent system for different engineering applications. According to whether the concept of equilibrium policies is explicitly considered in the learning process, existing MARL algorithms can be classified into non-equilibrium-based algorithms and equilibrium-based algorithms.

A. Non-Equilibrium-Based MARL Approaches

Non-equilibrium-based MARL approaches typically allow the agents to iteratively update their policies without incorporating equilibrium solution concepts in game theory into MARL [1], [2], [3], [4], [7], [10], [11], [15], [23], [24], [25]. In the past few years, this type of MARL approaches have found significant applications in modern systems/networks, such as collaborative jamming decision-making in cognitive electronic warfare [1], dynamic trajectory design and resource management in multi-AAV-aided vehicular communication networks [2], [4], [10], [11], [23], joint optimization of data sensing and computation offloading in AAV-aided mobile crowdsensing systems [7], [24], trajectory design and task offloading in AAV-aided mobile edge computing systems [3], [15], and joint traffic control and multi-channel reassignment for core networks [26]. For example, in [2], Chang et al. proposed a centralized deep reinforcement learning method and a decentralized MARL framework to jointly optimize trajectory design, power allocation, and user association in multi-AAV networks. In [4], Lu et al. proposed a two-timescale resource allocation framework for Vehicle-to-Everything network slicing, combining an MARL approach with Proximal Policy Optimization for dynamic inter-slice bandwidth allocation. The framework achieves considerable spectral efficiency gains while preserving privacy and meeting quality-of-service requirements. In [7], He

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Description
$\mathcal{N}$	set of agents (players)
N	number of agents in the environment
$\mathcal{S}$	state space of the stochastic game
s	state of the environment
$\mathcal{A}_i$	action (pure strategy) space of agent $i \in \mathcal{N}$
a_i	action (pure strategy) of agent i
a_j	action (pure strategy) of agent j, $j \in \mathcal{N}, j \neq i$
$\boldsymbol{a}$	joint action (i.e., action profile) of all the agents
$\mathcal{A}$	joint action space of all the agents
$\boldsymbol{a}_{-i}$	action profile of all the agents except agent i
$(a_i, \boldsymbol{a}_{-i})$	agents' action (pure strategy) profile
$\pi_i(s)$	policy (mixed strategy) of agent i at state s
$\boldsymbol{\pi}$	joint policy (mixed strategy profile) of all the agents
$R_i(s, \boldsymbol{a})$	reward of agent i at state s when the agents take the joint action $\boldsymbol{a}$
$U_i(s, \boldsymbol{\pi})$	payoff function for player i at state s corresponding to joint policy $\boldsymbol{\pi}$
$Q_i^{\boldsymbol{\pi}}(s, \boldsymbol{a})$	action-value function (i.e., Q-function) for agent i under the joint policy $\boldsymbol{\pi}$
$V_i(s')$	expected return of agent i for selecting state $s' \in \mathcal{S}$
$\boldsymbol{Q}_i(s)$	Q-function of agent i at state s
F_i^k	payoff of the atom-agents that choose strategy $k \in \mathcal{A}_i$ in population i
$\mathcal{P}$	a society consisting of P populations
$\boldsymbol{x}_i(s)$	population state of population $i \in \mathcal{P}$
$x_i^k(s)$	proportion of atom-agents in population i choosing the kth strategy in $\mathcal{A}_i$
$\boldsymbol{x}(s)$	social state describing the behavior of atom-agents in all the N populations
$F_i^k(\boldsymbol{x}(s))$	payoff for the atom-agents choosing the kth strategy in $\mathcal{A}_i$ at social state $\boldsymbol{x}(s)$
$x_j^{a_j}(s)$	proportion of the atom-agents in population j choosing strategy a_j
$\bar{F}_i(\boldsymbol{x}(s))$	average payoff obtained by the atom-agents in population i at social state $\boldsymbol{x}(s)$
$\tilde{F}_i^k(\boldsymbol{x}(s))$	excess payoff of the kth strategy in $\mathcal{A}_i$
$\Theta_i(s)$	expected payoff of agent i at the NE point of state s
$\boldsymbol{\xi}(s)$	initial condition for solving the ODE for state s
$\delta(s)$	maximum allowed threshold of $\|\|\dot{\boldsymbol{x}}(s)\|\|$

et al. introduced an MARL algorithm with future state prediction and parameter resetting for dynamic CPU-GPU task scheduling in large-scale Internet-of-Things (IoT) systems. Substantial reductions in average task processing time and improvements in system throughput have been achieved. In [3], Wang et al. presented an MARL framework for trajectory planning in AAV-assisted edge computing, jointly optimizing AAV paths and task offloading. The integration of prioritized experience replay and decentralized execution achieves efficient resource utilization and collision-free AAV coordination.

These non-equilibrium-based MARL algorithms are, in general, easy to implement, with low computational overhead and good scalability, and have been demonstrated effective for cooperative environments. However, they typically suffer from no theoretical convergence guarantee and suboptimality of the learned polices, especially in competitive or mixed settings.

B. Equilibrium-Based MARL Approaches

In contrast, equilibrium-based MARL algorithms explicitly consider game-theoretic equilibrium concepts in the learning process [27]. The idea is that agents learn policies that form an equilibrium, so each agent's policy is optimal given the others'. Based on the solution concept adopted by the algorithm, we can categorize the literature into NE-based [8], [17], [19], [28], [29], [30] and CE-based [16], [31] approaches. Littman proposed in [19] the minimax-Q learning algorithm to solve two-player zero-sum games. Hu and Wellman in [17] proposed the Nash-Q learning algorithm, which generalizes the minimax-Q algorithm to solve general-sum games. Later on, Littman presented in [28] the Friend-or-Foe Q-learning algorithm (FF-Q) to transform the multi-agent general-sum game into a two-agent zero-sum game, overcoming the restricted conditions of Nash-Q for convergence-guarantee. Greenwald et al. proposed in [16] the CE-Q learning algorithm, which computes the CE (instead of the NE) at each state with potential higher payoffs than NE to update the action-value functions of the agents.

These equilibrium-based MARL algorithms, in general, provide stronger theoretical convergence guarantees and strategic optimality under equilibrium assumptions, and are more effective for competitive/mixed environments. However, since equilibrium computations at each system state are required, they often suffer from the issue of high complexity in equilibrium computation as the number of agents increases.

C. Population Game-Based Approaches

Evolutionary game dynamics, combined with game-theoretic modeling, have recently been recognized as an exceptional framework within population games for designing dynamics and solutions for diverse engineering tasks [32], [33], [34], [35], [36]. The strength of this framework lies in its dynamic evolutionary properties rooted in strategic interactions within populations, which naturally derive stable solutions (e.g., NE) applicable to complex systems, thereby tightly linking engineering problem-solving to the rational equilibria of population games. For example, in [33], Tan. et al. presented a distributed population game model with graphical interaction constraints and designs distributed dynamics for static and time-varying communication networks, eliminating the reliance on global information. However, this study focuses only on single-agent static scenarios. In [34], Ming et al. proposed an evolutionary game-based dynamic strategy selection method for hybrid vehicle-to-vehicle communications, improving transmission throughput and robustness in vehicular ad hoc networks. Though dynamic environment is considered, the system is still a single-agent system since all vehicles share the same strategy set. Later on, Tan. et al. proposed a consensus-based multipopulation game dynamics approach to address the centralized information dependency in traditional population dynamics under objective coupling. However, their convergence proofs rely on concave objective functions, limiting applicability to non-convex or non-smooth problems.

To sum up, the application of population games to the equilibrium-seeking problem in multi-agent, state-based games

remains sparse in the literature. Our study aims to fulfill this research gap by leveraging population dynamics to enable efficient equilibrium policy learning in the equilibrium-based MARL framework.

III. MARL Model: A Stochastic Game Approach

Stochastic games are the most widely adopted framework to model the non-cooperative multi-agent environment in the equilibrium-based MARL approaches [16], [17], [18], [19]. In this paper, we follow this tradition and define an N-player stochastic game as follows:

Game 1: A stochastic game is represented by a tuple $\Gamma_{\text{SG}} = (\mathcal{N}, \mathcal{S}, \mathcal{P}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{R_i\}_{i\in\mathcal{N}})$, where

- *Agents:*[1] The set $\mathcal{N}$ of agents in the environment, where $\mathcal{N} = \{1, 2, \ldots, N\}$;
- *States:* The observations that the agents receive from the environment. For example, for a grid-world game, the state is the positions of all the agents. We denote a state of the environment as $s \in \mathcal{S}$, where $\mathcal{S}$ represents the state space;
- *Action:*[2] Let $\mathcal{A}_i$ be the action space of agent i. Note that the agents may be heterogeneous ones with varying action spaces. Each agent chooses an action $a_i \in \mathcal{A}_i$ at each state. We denote the joint action (i.e., action profile) of all the agents by $\boldsymbol{a} \in \mathcal{A}$, where $\mathcal{A}$ is the joint action space of all the agents, defined as the Cartesian product of the agents' potentially different individual action spaces, i.e., $\mathcal{A} = \prod_{i\in\mathcal{N}} \mathcal{A}_i$. For agent i, the agents' action profile is also denoted as $(a_i, \boldsymbol{a}_{-i})$, where $\boldsymbol{a}_{-i} \in \mathcal{A}_{-i}$ denotes the action profile of all the agents except of agent i;
- *Transition probability:* The probability of transitioning from one state of the environment to another. We denote the transition probability from state s to s' when the agents take the joint action $\boldsymbol{a}$ as $P[s'|s, \boldsymbol{a}] \in \mathcal{P}$, where $\mathcal{P}$ is a mapping from $\mathcal{S} \times \mathcal{A}$ to $\mathcal{S}$;
- *Reward:*[3] Let $\mathbb{R}_+$ denote the set of positive numbers. We denote the reward of agent i at the state s when the agents take the joint action $\boldsymbol{a}$ as $R_i(s, \boldsymbol{a}) \in \mathcal{R}_i$, where $\mathcal{R}_i$ is a mapping from $\mathcal{S} \times \mathcal{A}$ to $\mathbb{R}_+$. The agents can have asymmetric rewards, meaning that they may receive different payoffs for the same game state and joint action;
- *Policy:*[4] A policy of agent i assigns each state $s \in \mathcal{S}$ a vector of probabilities of the agent taking actions over its action space, defined as $\pi_i(s) : \mathcal{S} \to \mathbb{R}_+^{|\mathcal{A}_i|}$. Let $\boldsymbol{\pi}$ denote the joint policy of all the agents, where $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$.

In Game 1, the goal of each agent i is to maximize the discounted sum of future rewards for the state s_t corresponding to time step t, formulated as:

$$\max \sum_{k=0}^{\infty} \gamma^k E\{R_i^{t+k} | \boldsymbol{\pi}, s_t = s\}, \tag{1}$$

where s is a particular state, R_i^{t+k} is the reward of agent i at time step $t+k$, and $\gamma \in [0, 1)$ is the discount factor. A stochastic game Γ_{SG} typically starts from an initial state s_{ini}, experiences a number of intermediate states s, and ends with the terminal state s_{term}. For each experienced state s, we can construct a normal-form game to model the interactions among the agents as follows:

Game 2: We represent the N-player normal-form game at a particular state s in a stochastic game by a tuple $\Gamma_s^{\text{NFG}} = (\mathcal{N}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{U_i(s, \boldsymbol{\pi})\}_{i\in\mathcal{N}})$, where $U_i(s, \boldsymbol{\pi})$ represents the payoff function for player i at state s under joint policy $\boldsymbol{\pi}$.

To have agents learn the equilibrium policies with maximized long-term reward as shown in (1), the payoff function $U_i(s, \boldsymbol{\pi})$ should represent the long-term expected return of agent i. The reward function R_i is not a good choice since it only means the immediate return but does not capture the long-term expected return. In this study, we leverage the Q-values as the payoff functions for the agents, as elaborated in Section IV.

NE is an important solution concept to define the equilibrium policies of the agents in Game 2, defined as follows:

Definition III.1 (Nash equilibrium, NE): In the normal-form game Γ_s^{NFG}, a mixed-strategy profile $\boldsymbol{\pi}^* = (\pi_1^*, \ldots, \pi_N^*)$, where π_i^* is a vector of probability distribution over the action space $\mathcal{A}_i$, is a mixed-strategy NE if and only if for any $i \in \mathcal{N}$,

$$U_i(s, \boldsymbol{\pi}^*) \geq U_i(s, \pi_i', \boldsymbol{\pi}_{-i}^*), \forall \pi_i' \in \Delta_i,$$

where $\boldsymbol{\pi}_{-i}^* = (\pi_1^*, \ldots, \pi_{i-1}^*, \pi_{i+1}^*, \ldots, \pi_N^*)$ and Δ_i represents the mixed-strategy space of player i.

In the MARL environment, each agent's reward at each time step depends not only on its own policy but also on other agents' policies. Thus, simply applying the independent Q-learning algorithm [37] to solve (1) may not achieve the guaranteed convergence. In the following section, we design an algorithmic framework that allows the agents to learn the NE policies efficiently at each state in order to solve (1).

IV. Proposed Pop-Q Algorithm

This section proposes an equilibrium-based MARL algorithm exploiting population dynamics (i.e., Pop-Q algorithm) that allows the agents to learn the NE policies to solve (1). We first present an overview of the algorithm. Then, we describe the main components in greater detail, followed by a convergence analysis of the Pop-Q algorithm.

A. Algorithm Overview

The Pop-Q algorithm mainly consists of three components:

- *Equilibrium-based Q-learning framework:* We adopt the equilibrium-based Q-learning framework [18] to allow the agents to learn equilibrium policies in the stochastic game. The core idea is to force the agents to learn equilibrium policies by playing a series of normal-form games sequentially so that they update their Q-functions according to the computed equilibrium at each state. We illustrate the details of this framework in Section IV-B.
- *Population game formulation in each state:* Under the Q-learning framework, we propose to formulate the normal-form game at each state in MARL as a population game.

[1]The term "agent" in an MARL system refers to "player" in the corresponding stochastic game. We will use these two terms interchangeably in this study.

[2]The term "action" in an MARL system refers to "pure-strategy" in the corresponding stochastic game. We will use them interchangeably in this study.

[3]The term "reward" in an MARL system refers to "payoff" in the corresponding stochastic game. We will use them interchangeably in this study.

[4]The term "policy" in an MARL system refers to "mixed-strategy" in the corresponding stochastic game. We will use them interchangeably in this study.

In this game, we expand each agent in MARL into a population of "atom-agents" and use a population's state to represent the corresponding agent's mixed-strategy. We present the details of this formulation in Section IV-C.

- *Inheritance-based population dynamic:* Under the novel population game formulation at each state, we design a population dynamic with inheritance-based initial condition and threshold-based derivative calculation to efficiently find the approximate mixed-strategy NE of the agents, as illustrated in Sections IV-D and IV-F.

B. Equilibrium-Based Q-Learning Framework

Originally developed for solving single-agent RL problems, the standard Q-learning algorithm has been widely used to find the optimal policy of an agent in a dynamic environment. In this study, we use the formulation of standard Q-learning as a starting point and propose a population game-based MARL framework to solve the stochastic game Γ_{SG}.

For each agent i, we define $Q_i^{\boldsymbol{\pi}}(s, \boldsymbol{a})$ as the action-value function (i.e., Q-function) for agent i under the joint policy $\boldsymbol{\pi}$. The value of $Q_i^{\boldsymbol{\pi}}(s, \boldsymbol{a})$ represents the expected return of agent i starting from the state s, taking the joint action $\boldsymbol{a}$, and thereafter all the agents following joint policy $\boldsymbol{\pi}$, i.e., [38]

$$Q_i^{\boldsymbol{\pi}}(s, \boldsymbol{a}) = E^{\boldsymbol{\pi}}\left\{\sum_{k=0}^{\infty} \gamma^k R_i^{t+k} | s_t = s, \boldsymbol{a}_t = \boldsymbol{a}\right\}. \quad (2)$$

We aim to design a learning algorithm that estimates the value of $Q_i^{\boldsymbol{\pi}}(s, \boldsymbol{a})$, which reflects how good it is for agent i to take a given action at a given state. Through continuously interacting with the environment, the agents' joint policy $\boldsymbol{\pi}$ is improved by updating the Q-functions. Given a data sample $(s, \boldsymbol{a}, s', R_i)$ in the learning procedure, the update rule to find the optimal policy for each agent $i \in \mathcal{N}$ is [17]:

$$Q_i(s, \boldsymbol{a}) = (1-\alpha) Q_i(s, \boldsymbol{a}) + \alpha \left[R_i(s, \boldsymbol{a}) + \gamma V_i(s')\right], \quad (3)$$

where $R_i(s, \boldsymbol{a})$ is the reward function for agent i, α is the learning rate, γ is the discount factor, and $V_i(s')$ represents the expected return of agent i for selecting the state s'. In a non-cooperative multi-agent setting, simply maximizing the Q-value of agent i at the next state s' is not suitable to find the value of $V_i(s')$. This is because when multiple agents exist in the environment with individual rewards, the joint action that maximizes the reward of one agent does not guarantee that other agents' rewards are maximized [39]. In subsequent subsections, for the first time in the literature, we introduce the population game theory into non-cooperative MARL to find the value of $V_i(s')$. The contribution of our approach is two-fold: 1) a novel population game formulation in each state; and 2) an efficient computation framework to compute the approximate mixed-strategy NE for the agents.

C. Population Game Formulation

For a given state s, we aim to find the equilibrium policies for all the agents and then take the expected payoffs of the agents at the equilibrium as the value of $V_i(s')$. We achieve this goal in three steps. First, we construct a Q-learning based normal-form game at each state. Second, we reformualte this game into a population game. Third, we compute the approximate equilibrium leveraging an inheritance-based population dynamic. We illustrate the first two steps in more detail here while presenting the details of the last step in Sections IV-D and IV-F.

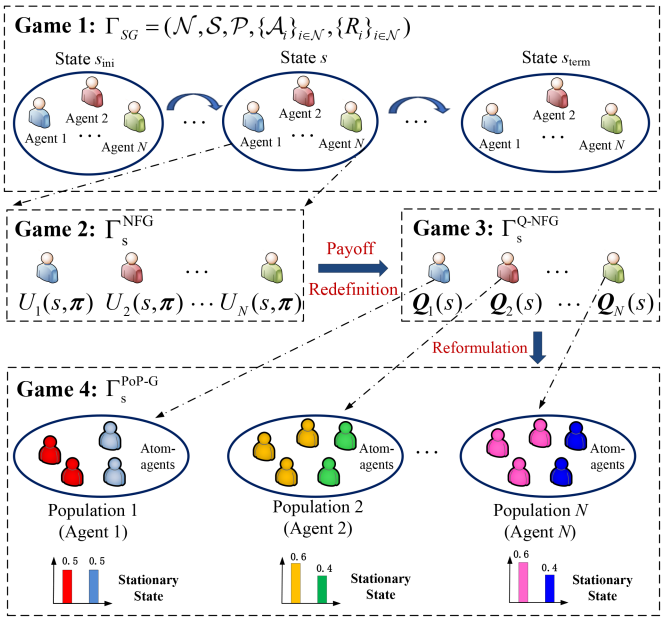


Fig. 1. Illustration of the connections between the four games in this paper. Game 1 represents our foundational model; Game 2 focuses on a single state within game 1; Game 3 maintains the normal-form structure of game 2 but redefines the payoff function for each agent with Q-functions; Game 4 reformulates game 3 to allow the use of population dynamics as an efficient mechanism for finding approximate equilibria.

As illustrated in Fig. 1, we treat the Q-function $Q_i(s, \boldsymbol{a})$ as the payoff function for agent i and use this payoff function to construct an N-player normal-form game at each state $s \in \mathcal{S}$ as follows:

Game 3: We represent the Q-learning based N-player normal-form game at state s by a tuple $\Gamma_s^{\text{Q-NFG}} = (\mathcal{N}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{\boldsymbol{Q}_i(s)\}_{i\in\mathcal{N}})$, where $\mathcal{N}$ is the set of agents, $\mathcal{A}_i$ is the action space of agent i, and $\boldsymbol{Q}_i(s)$ represents the Q-function of agent i at state s, i.e., $\boldsymbol{Q}_i(s) = \{Q_i(s, \boldsymbol{a})\}_{\boldsymbol{a}\in\mathcal{A}}$.

Subsequently, we further model each player in Game 3 as a population comprised of numerous atom-agents. While an agent employs mixed strategies, the atom-agents within that population play pure strategies exclusively. The probability distribution of atom-agents selecting various pure strategies directly represents the mixed strategy of the corresponding agent. This distribution also characterizes the state of the corresponding population. In this way, we can model the strategic interactions among the agents by that of the atom-agents in the populations. As shown in Fig. 1, we reformulate Game 3 as a population game (Game 4) to model the strategic interactions among the agents in game $\Gamma_s^{\text{Q-NFG}}$:

Game 4: We represent the population game formulated at state s by a tuple $\Gamma_s^{\text{POP-G}} = (\mathcal{N}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{F_i^k\}_{i\in\mathcal{N}, k\in\mathcal{A}_i})$, where $\mathcal{N}$ is the set of populations, $\mathcal{A}_i$ is the set of pure strategies available to the atom-agents in population i, and F_i^k represents the payoff of the atom-agents that choose strategy $k \in \mathcal{A}_i$ in population i.

Some important concepts in Game 4 are defined as follows:

- *Society, Populations, and Atom-agents:* Define $\mathcal{P} = \{1, 2, \ldots, P\}$ as a society consisting of P populations. Each population corresponds to an agent in the normal-form game $\Gamma_s^{\text{Q-NFG}}$. Each population contains M "atom-agents" with pure strategies available to them. Fig. 1 illustrates the constitution of a population game with N populations.
- *Strategies:* Atom-agents in a population can choose their strategies from a set of pure strategies available to them. The set of pure strategies available to the atom-agents in population $i \in \mathcal{P}$ is $\mathcal{A}_i$. In the population game corresponding to state s, the atom-agents in the N populations match with each other to play the normal-form game $\Gamma_s^{\text{Q-NFG}}$. Each atom-agent from one population picks up a pure strategy available to it.
- *Population State:* For a given state $s \in \mathcal{S}$, we define the population state of population i corresponding to all atom-agents of agent $i \in \mathcal{N}$ as the probability distribution of atom-agents choosing among all the pure strategies available to agent i:

$$\boldsymbol{x}_i(s) = \{x_i^k(s) | k = 1, 2, \ldots, |\mathcal{A}_i|\}, \tag{4}$$

where $x_i^k(s)$ represents the proportion of atom-agents in population i choosing the kth strategy in $\mathcal{A}_i$. The set of population states for agent i is

$$\mathbb{X}_i(s) = \left\{ \boldsymbol{x}_i(s) \in \mathbb{R}_+^{|\mathcal{A}_i|} : \sum_{k \in \mathcal{A}_i} x_i^k(s) = 1 \right\}. \tag{5}$$

Note that a state of population i is mathematically equivalent to a mixed-strategy of agent i in the normal-form game $\Gamma_s^{\text{Q-NFG}}$.

- *Social State:* The state of a society (a.k.a. social state) is a combination of the states of all the populations of all N agents. Let $\boldsymbol{x}(s)$ be the social state which describes the behavior of atom-agents in all the N populations, formally defined as, $\boldsymbol{x}(s) = \{\boldsymbol{x}_i(s) | i \in \mathcal{N}\}$. Note that a social state $\boldsymbol{x}(s)$ is mathematically equivalent to a mixed-strategy profile of all the agents in the normal-form game $\Gamma_s^{\text{Q-NFG}}$.
- *Payoff Function:* For social state $\boldsymbol{x}(s)$, we define a payoff function for the atom-agents choosing the same strategy in the same population. Denote the pure strategy profile of the N atom-agents (one from each population) matched to play the game $\Gamma_s^{\text{Q-NFG}}$ as $\boldsymbol{a} = (a_i, \boldsymbol{a}_{-i})$. The notation $\boldsymbol{a}_{-i} = \{a_j | j \in \mathcal{N}, j \neq i\}$ represents the collection of pure strategies chosen by all agents except agent i, where j represents any agent different from agent i and and $a_j \in \mathcal{A}_j$ represents a pure strategy available to agent j.

1) The payoff for the atom-agents choosing the kth strategy in $\mathcal{A}_i$ at social state $\boldsymbol{x}(s)$ is:

$$F_i^k(\boldsymbol{x}(s)) = \sum_{\boldsymbol{a}_{-i} \in \mathcal{A}_{-i}} \left(Q_i(s, k, \boldsymbol{a}_{-i}) \prod_{j \in \mathcal{N}, j \neq i} x_j^{a_j}(s) \right) \tag{6}$$

where $x_j^{a_j}(s)$ denotes the proportion of the atom-agents in population j choosing strategy a_j.

2) We express the average payoff obtained by the atom-agents in population i at social state $\boldsymbol{x}(s)$ as:

$$\bar{F}_i(\boldsymbol{x}(s)) = \sum_{k \in \mathcal{A}_i} x_i^k(s) F_i^k(\boldsymbol{x}(s)). \tag{7}$$

3) We define the excess payoff of the kth strategy in $\mathcal{A}_i$ as the difference between strategy k's payoff and the average payoff of population i:

$$\widetilde{F}_i^k(\boldsymbol{x}(s)) = F_i^k(\boldsymbol{x}(s)) - \bar{F}_i(\boldsymbol{x}(s)). \tag{8}$$

- *Stochastic evolutionary process:* The payoff function $F_i^k(\boldsymbol{x}(s))$ describes the strategic environment. A revision protocol will decide when an atom-agent in a population switches its strategy and to which strategy he should switch [21]. Starting from an initial condition $\boldsymbol{\xi}(s)$, a revision protocol will lead to a stochastic evolutionary process $\{\boldsymbol{x}_\tau^M(s)\}$, where τ is the iteration counter.

Fig. 1 illustrates the relationships among the four games in this paper. Game 1 (Γ_{SG}) represents a stochastic game formulation that captures the sequential, state-based interactions among multiple agents in non-cooperative environments. Game 2 (Γ_s^{NFG}) focuses on a single state within Game 1, presenting a standard normal-form (stateless) game that describes agent interactions at a particular point in the broader stochastic game. Game 3 ($\Gamma_s^{\text{Q-NFG}}$) maintains the normal-form structure of Game 2 but redefines the payoff function for each agent using their respective Q-functions ($\boldsymbol{Q}_i(s)$). This critical transformation allows us to integrate Q-learning principles into the equilibrium-finding process for Game 1. However, this approach still faces computational complexity challenges when determining equilibrium policies. To address these computational limitations, we reformulate Game 3 as Game 4 ($\Gamma_s^{\text{POP-G}}$) - a population game that models the strategic interactions among agents. This final transformation enables us to leverage population dynamics as an efficient mechanism for finding approximate equilibria, significantly reducing computational complexity while maintaining solution quality. Fig. 2 illustrates how the population state evolution is related to the Q-values in a two-agent environment. In Sections IV-D and IV-F, we design an inheritance-based population dynamic to allow the population states to evolve into an approximate mixed-strategy NE for all the populations.

D. Proposed Pop-Q Algorithm

Given the population game formulated at each state, we adopt the Brown-von Neumann-Nash (BNN) dynamic [40] as the basis to adapt the atom-agents' behavior to the population game. It has been proved that the stationary points of the BNN dynamic correspond to the Nash equilibria of the underlying population game, and that the stationary point is unique from a given initial social state [22].

BNN dynamic: We describe the BNN dynamic as follows: At each time instant, the dynamic randomly selects an atom-agent who receives a revision opportunity to change its strategy. The rate at which revision opportunities arrive at the atom-agents and the probabilities with which they change to each strategy are both functions of current excess payoffs. Let τ denote the time counter

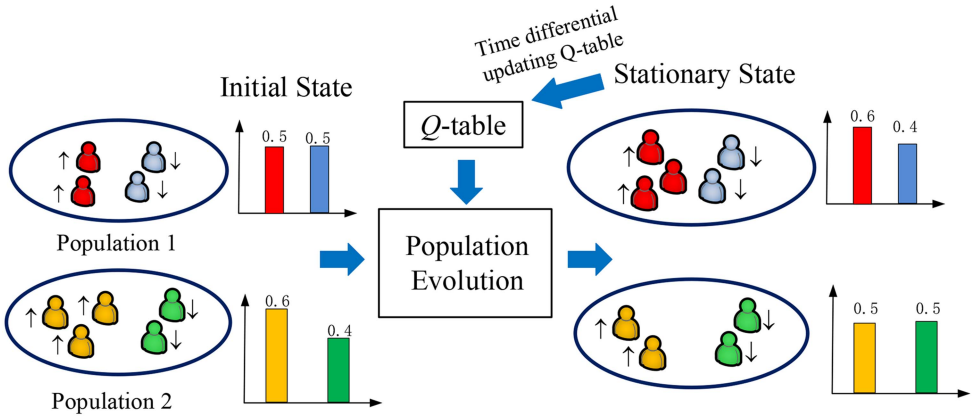


Fig. 2. Illustration of updating the Q-tables based on population state evolution. The number of atom-agents in each population choosing different strategies in the figure is for an illustrative purpose. The population size M needs to go to infinity to allow the use of ODE to describe the evolutionary process.

Algorithm 1: Proposed Pop-Q Algorithm.

1 **Input:** Learning rate α, discount factor γ, exploration factor ϵ, agent set $\mathcal{N}$
2 Set $Q_i(s, \boldsymbol{a}) \leftarrow 0, \forall s \in \mathcal{S}, \forall \boldsymbol{a} \in \mathcal{A}, \forall i \in \mathcal{N}$
3 **for** $e = 1$ ***to*** N_{eps} **do**
4 Initialize the system state $s \leftarrow s_{\text{ini}}$
5 **while** $s! = s_{term}$ **do**
6 Choose the action profile $\boldsymbol{a}$ for the agents according to $\boldsymbol{x}^*(s)$ with ϵ-greedy policy
7 **for** *agent* $i \in \mathcal{N}$ **do**
8 Take the chosen action
9 Observe the experience $(s, \boldsymbol{a}, R_i, s')$
10 Form the normal-form game:
11 $\Gamma_{s'}^{\text{Q-NFG}} = (\mathcal{N}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{Q_i(s')\}_{i\in\mathcal{N}})$
12 Formulate the population game identified by the payoff functions (6) and (7)
13 Compute the mixed-strategy NE at s':
14 $\boldsymbol{x}^*(s') \leftarrow$ $\text{BNN}(s', F_1^1(\boldsymbol{x}(s')), \ldots, F_N^{|\mathcal{A}_N|}(\boldsymbol{x}(s')))$
15 **for** *agent* $i \in \mathcal{N}$ **do**
16 $\Theta_i(s') \leftarrow \bar{F}_i(\boldsymbol{x}^*(s))$
17 $Q_i(s, \boldsymbol{a}) \leftarrow$ $(1-\alpha)Q_i(s, \boldsymbol{a}) + \alpha(R_i + \gamma\Theta_i(s'))$
18 Update the state $s \leftarrow s'$

in the evolutionary process[5]. Following standard population-game practice, we model each population as a continuum of atom-agents. This continuum assumption turns discrete updates into differential equations and eliminates finite-size artefacts such as random drift [21]. When the population size goes to infinity (i.e., $M \to \infty$), the following ODEs mathematically describes the evolutionary process ($\forall i \in \mathcal{N}, \forall k \in \mathcal{A}_i$) [41]:

$$\frac{d}{d\tau}x_i^k(s) = [\widetilde{F}_i^k(\boldsymbol{x}(s))]_+ - x_i^k(s)\sum_{l\in\mathcal{A}_i}[\widetilde{F}_i^l(\boldsymbol{x}(s))]_+, \tag{9}$$

where $[\widetilde{F}_i^k(\boldsymbol{x}(s))]_+ = \max\{0, \widetilde{F}_i^k(\boldsymbol{x}(s))\}$. The social state is stationary if we have $dx_i^k(s)/d\tau = 0$, $\forall i \in \mathcal{N}$, $\forall k \in \mathcal{A}_i$. The ODE-based dynamics in (9) provide a tractable mathematical framework while still capturing the essential evolutionary behavior of the system. Numerous iterative methods can be applied to solve the above ODE and we adopt the Runge-Kutta method [42] in this study. Given an initial condition, the rest point of the BNN dynamic represented by the ODE in (9) is an NE of the underlying population game, where no agent benefits from changing its mixed-strategy unilaterally [22].

Once the mixed-strategy NE policies are computed, we use the average payoffs for the populations at the rest point of the dynamic to update the Q-values of the corresponding agents. Denote the expected payoff of agent i at the NE point as $\Theta_i(s)$, which is given by

$$\Theta_i(s) = \bar{F}_i(\boldsymbol{x}^*(s)). \tag{10}$$

Substituting (10) into (3), we obtain the update rule of the Pop-Q algorithm:

$$Q_i(s, \boldsymbol{a}) = (1-\alpha)Q_i(s, \boldsymbol{a}) + \alpha\left[r_i + \gamma\Theta_i(s')\right]. \tag{11}$$

We summarize the proposed Pop-Q algorithm in Algorithm 1. As can be seen from the algorithm, we first initialize the elements of Q-table for each agent as zero (Line 2). Then, for each episode during the learning process, we first initialize the system state s and then choose a joint action according to the computed equilibrium at s with probability $(1-\epsilon_2)$ and choose a random joint action with probability ϵ_2 (Line 6). The agent takes the chosen action (as part of the joint action) and then observes the experience $(s, \boldsymbol{a}, R_i, s')$ (Line 7 - Line 9).

Next, we form an N-player normal-form game $\Gamma_{s'}^{\text{Q-NFG}}$ with the Q-values at the next state s' (Line 10 - Line 11). We then formulate a population game corresponding to the normal-form

[5]In the proposed algorithm, the Q-values and population states are updated in two different time scales. The symbol t denotes the iteration counter for Q-learning, while the symbol τ represents the iteration counter for population state evolutionary at each system state in MARL.

game where the agents' Q-values are analytically related to the payoffs of the "atom-agents" (Line 12). We then exploit the BNN dynamic to find the mixed-strategy NE of the formulated population game (Line 13 - Line 14). The BNN dynamic enables the population state to evolve into a stationary state corresponding to an mixed-strategy NE at state s' in the stochastic game.

Lastly, we update the Q-table for the state-joint action pair $(s, \boldsymbol{a})$ for each agent using the expected payoff $\Theta_i(s')$ at the computed approximate equilibrium (Line 15 - Line 17). Once the Q-tables are updated after sufficiently many episodes, one can derive the equilibrium policies for the agents at each stage of the game with the Q-values as inputs.

E. Convergence Analysis

We provide a brief convergence analysis of the Pop-Q algorithm here and defer the detailed proofs to the appendix. In our algorithm, updating Q-values and population states with a different number of steps in each iteration creates two different time scales. Therefore, in the limit of small learning rates, the sequence of population state and Q-values $\{Q(t), x(t)\}$ could be approximated by the following system of ordinary differential equations:

$$\begin{aligned} \epsilon \dot{\boldsymbol{x}}_\epsilon(t) &= H(\boldsymbol{x}_\epsilon(t), Q_\epsilon(t)), \\ \dot{Q}_\epsilon(t) &= G(\boldsymbol{x}_\epsilon(t), Q_\epsilon(t)), \end{aligned} \tag{12}$$

where $\epsilon \ll 1$ controls the difference in the time-scale between $\boldsymbol{x}_t$ and Q_t and

$$H^k_{a_k,s}(x, Q) = [\widetilde{F}^k_{a_k}(\boldsymbol{x}(s))]_+ - x^k_{a_k}(s) \sum_{a'_k \in \mathcal{A}_k} [\widetilde{F}^k_{a'_k}(\boldsymbol{x}(s))]_+$$

$$G^k_{\mathbf{a},s}(x, Q) = r^k(\mathbf{a}, s) + \beta \sum_{s' \in \mathbb{S}} p(s') \sum_{\mathbf{a}'} Q^k_{s',\mathbf{a}'} \Pi^n_{k'=1} x^{k'}(s', \mathbf{a}'_{k'})$$

The intuition behind the two time-scales is as follows: As the evolution of the population state is much faster than that of the Q-values, population state $\boldsymbol{x}_t$ treats Q-values Q_t as quasi-constant, and Q-values Q_t treat population state $\boldsymbol{x}_t$ as quasi-equilibrated. More precisely, in the time-scale of $\boldsymbol{x}(t)$, its dynamic is approximated by

$$\dot{\boldsymbol{x}}(t) = H(\boldsymbol{x}(t), Q),$$

for some fixed constant $Q \in \mathbb{Q}$. While in the time-scale of $Q(t)$, its dynamic is approximated by

$$\dot{Q}(t) = G(\lambda(Q(t)), Q(t)),$$

where $\lambda : \mathbb{Q} \to \mathbb{X}$ satisfying $H(\lambda(Q), Q) = \mathbf{0}$ for all $Q \in \mathbb{Q}$ is a fixed point of the fast dynamic, which means that for each given $Q \in \mathbb{Q}$, $x(t)$ will quickly converge to the corresponding equilibrium state $\lambda((Q))$.

To be more precise, $\boldsymbol{x}_\epsilon$ and Q_ϵ in this system of differential equations quickly converges to a stable slow manifold

$$\mathcal{M} = \{(\boldsymbol{x}, Q) : Q \in \mathbb{Q}, H(\boldsymbol{x}, Q) = 0\} \tag{13}$$

in exponential rate, as $\epsilon \to 0$,

$$\|\boldsymbol{x}_\epsilon(t) - \lambda(Q_\epsilon(t))\| \leq M \|\boldsymbol{x}_\epsilon(0) - \lambda(Q_\epsilon(0))\| \exp(-kt) \tag{14}$$

as shown by Theorem A.1. Then the effective dynamics of $Q_\epsilon(t)$ on the stable slow manifold is reduced to

$$\dot{Q}(t) = G(\lambda(Q(t)), Q(t))$$

Recall that we conclude the convergence of $(\boldsymbol{x}(t), Q(t))$ to the slow manifold and derive the effective dynamics of $Q(t)$ on the slow manifold. This is done by showing that the sequence of population state and Q-values $\{\boldsymbol{x}_t, Q_t\}$ could be approximated by a system of singular perturbed ordinary differential equations. More precisely, the approximation is in the sense that the sequence $\{\boldsymbol{x}_t, Q_t\}$ is a (T, δ)-perturbation of the ODE system.

Definition IV.1 ((T, δ)-perturbation): For an ODE

$$\dot{x} = h(x(t)), \quad x \in \mathbb{R}^d, \tag{15}$$

given $T, \delta > 0$, its (T, δ)-perturbation is a bounded measurable function $y : [0, \infty) \to \mathbb{R}^d$, such that there exists $0 = T_0 < T_1 < \cdots < T_n \to \infty$ and solutions $x^j(t), t \in [T'_j, T'_{j+1}], j \geq 0$ of ODE (27), such that $T'_{j+1} - T'_j = T_{j+1} - T_j \geq T$ for $j \geq 0$ and $\|y(t) - x^j(T'_j + t - T_j)\| \leq \delta$.

In the Appendix B, we prove that the sequence $\{x(t), Q(t)\}$ is indeed the (T, δ)-perturbation of the system of singular perturbed ODEs by the following steps:

1) In the fast time scale of x_t, it is a (T, δ)-perturbation of
$$\dot{x}(t) = H(x(t), Q) \quad \text{for some } Q \in \mathbb{Q};$$
2) The mapping $\lambda : \mathbb{Q} \to \mathbb{X}$ is a well defined injective and Lipschitz continuous function under some technical conditions;
3) In the slow time scale of Q_t, it is a (T, δ)-perturbation of
$$\dot{Q}(t) = G(\lambda(Q_t), Q_t).$$

One of the most important properties of a sequence which is the (T, δ)-perturbation of an ODE system is that the sequence will converge to the same limit set as the ODE system.

Lemma IV.2: Suppose that the ODE (27) has an asymptotically stable attractor J, then its any (T, δ)-perturbation converges to an ϵ-neighborhood of J, that is, $y(t) \to J^\epsilon$.

To prove the convergence the sequence $\{x(t), Q(t)\}$ in our algorithm, it remains to show that the effective dynamic of $Q(t)$ on the stable slow manifold has an unique asymptotically stable fixed point Q_*, such that $G(\lambda(Q_*), Q_*) = \mathbf{0}$ and Q_* coincides with the Nash Q-values. The former argument follows directly from the fact that $G(\lambda(Q(t)), Q(t))$ is a linear function of $Q(t)$, while the later argument could be proved by verifying the definition of Nash Q-values. Therefore, we conclude that Pop-Q converges to NE policy and Nash-Q values. The formal statements, corresponding technical assumptions and detailed proof of our convergence result are available in Appendix A (Theorem A.10).

Algorithm 2: Inheritance-Based BNN Dynamic for Approximate NE Computation.

Input: State s, $F_i^k(\boldsymbol{x}(s))$, $\bar{F}_i(\boldsymbol{x}(s))$, $\boldsymbol{x}^*(s)$, δ
$\boldsymbol{x}^*_{\text{pre}}(s) \leftarrow \boldsymbol{x}^*(s)$
$\eta \leftarrow rand(0,1)$
if $\eta >= \varepsilon_1$ **then**
  Inherit the previously computed equilibrium as the initial condition: $\boldsymbol{\xi}(s) \leftarrow \boldsymbol{x}^*_{\text{pre}}(s)$
else
  Generate a random distribution over the strategy space for each population i as the initial condition:
  $\xi_i(s) \leftarrow randdist(\mathcal{A}_i), \forall i \in \mathcal{P}$
Solve the ODE with the chosen initial condition using iterative methods, e.g., the Runge-Kutta method:
$\boldsymbol{x}^*(s) \leftarrow SolODE(\boldsymbol{\xi}(s), F_i^k(\boldsymbol{x}(s)), \bar{F}_i(\boldsymbol{x}(s)), \delta)$
Output: Newly computed equilibrium policies $\boldsymbol{x}^*(s)$

F. Inheritance-Based Population Dynamic

Since the most computationally expensive part in the equilibrium-based Q-learning framework is the equilibrium computation involved at each state, the rate at which the BNN dynamic reaches the NE becomes essential. Define $||\dot{\boldsymbol{x}}(s)|| = ||d\boldsymbol{x}(s)/d\tau||$, where $||\cdot||$ denotes the first-order norm of a vector. Let $\delta(s)$ and $\boldsymbol{\xi}(s)$ denote the maximum allowed threshold of $||\dot{\boldsymbol{x}}(s)||$ and the initial condition supplied to solving the ODE for state s, respectively. In general, a smaller $\delta(s)$ indicates a larger number of iterations and thus a longer time to solve the corresponding ODE. Still, the solution will be closer to the exact NE. On the other hand, an initial condition closer to the exact NE will help accelerate the convergence of solving the ODE. Therefore, both $\delta(s)$ and $\boldsymbol{\xi}(s)$ are key factors in determining the computation efficiency of the approximate NE.

In this section, we design an algorithm based on the BNN dynamic to find the approximate equilibrium efficiently. We design our algorithm based on the equilibrium similarity for the normal-form games corresponding to two successive visits to the same state in the stochastic game. Specifically, for two successive visits to the same state, we propose to inherit the equilibrium computed at the first visit as the initial condition for the population state evolution at the second visit. Moreover, we design another threshold-based mechanism to dynamically adjust the number of evolution iterations to accelerate the approximate equilibrium computation.

1) Inheritance-Based Initial Condition: Empirical experiments show that the chance of two normal-form games corresponding to two successive visits to the same state having similar equilibria is high [43]. Thus, in this study, we reuse the previously computed NE as the initial condition to solve the ODE corresponding to the second visit to the same state. In this way, we can reduce the iteration number to find the approximate equilibrium.

We propose an ε-greedy algorithm to inherit the precomputed equilibrium policies, which serves as the initial condition for solving the ODE corresponding to the same state. Our purpose is to accelerate the convergence rate of the BNN dynamic in finding the equilibrium of the population games formulated for each state s. Let $\Gamma_s^{\text{Q-NFG}}$ and $\bar{\Gamma}_s^{\text{Q-NFG}}$ denote the normal-form games formulated under two successive visits to the state $s \in \mathcal{S}$. Let $\boldsymbol{x}^*_{\text{pre}}(s)$ denote the approximate equilibrium obtained from solving the ODE in (9) corresponding to game $\Gamma_s^{\text{Q-NFG}}$, i.e.,

$$\boldsymbol{x}^*_{\text{pre}}(s) = \{\boldsymbol{x}_i^*(s) | i \in \mathcal{N}\}. \tag{16}$$

Let $\boldsymbol{\xi}(s)$ represent the initial condition for solving the ODE corresponding to the normal-form game $\bar{\Gamma}_s^{\text{Q-NFG}}$, with probability $1-\varepsilon_1$ (where $0 < \varepsilon_1 < 1$), we inherit $\boldsymbol{\xi}(s)$ from $\boldsymbol{x}^*_{\text{pre}}(s)$ as:

$$\boldsymbol{\xi}(s) = \boldsymbol{x}^*_{\text{pre}}(s). \tag{17}$$

With probability ε_1, we generate a random distribution over the strategy space $\mathcal{A}_i$ as population i's initial condition. Algorithm 2 summarizes the inheritance-based BNN dynamic to compute the approximate NE at each state.

2) Threshold-Based Mechanism: The threshold value $\delta(s)$ used in solving the ODE also affects the computation efficiency of approximate equilibrium. As the threshold value $\delta(s)$ becomes smaller, the solution from the ODE will become closer to the exact NE, but the time for solving the ODE also increases. For the approximate equilibria considered in this paper, we use

Algorithm 3: Proposed Pop-Q Algorithm with Inheritance-Based BNN Dynamic.

Input: Learning rate α, discount factor γ, exploration factor ϵ, agent set $\mathcal{N}$
Set $Q_i(s, \boldsymbol{a}) \leftarrow 0, \forall s \in \mathcal{S}, \forall \boldsymbol{a} \in \mathcal{A}, \forall i \in \mathcal{N}$
for $e = 1$ ***to*** N_{eps} **do**
  if $e == 1$ **then**
    $\delta_e \leftarrow \delta_{ini}$
  else
    $\delta_e \leftarrow \delta_{e-1}\kappa$
  Initialize the system state $s \leftarrow s_{\text{ini}}$
  while $s! = s_{term}$ **do**
    Choose the action profile $\boldsymbol{a}$ for the agents according to $\boldsymbol{x}^*(s)$ with ϵ-greedy policy
    for *agent* $i \in \mathcal{N}$ **do**
      Take the chosen action
      Observe the experience $(s, \boldsymbol{a}, R_i, s')$
    Form the normal-form game:
    $\Gamma_{s'}^{\text{Q-NFG}} = (\mathcal{N}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{Q_i(s')\}_{i\in\mathcal{N}})$
    Formulate the population game identified by the payoff functions (6) and (7)
    Compute the approximate NE at s':
    $\boldsymbol{x}^*(s') \leftarrow$ BNN$(s', F_1^1(\boldsymbol{x}(s')), \ldots, F_N^{|\mathcal{A}_N|}(\boldsymbol{x}(s')), \delta_e)$
    for *agent* $i \in \mathcal{N}$ **do**
      $\Theta_i(s') \leftarrow \bar{F}_i(\boldsymbol{x}^*(s))$
      $Q_i(s, \boldsymbol{a}) \leftarrow (1-\alpha)Q_i(s, \boldsymbol{a}) + \alpha(R_i + \gamma\Theta_i(s'))$
    Update the state $s \leftarrow s'$

a larger value for $\delta(s)$ to reduce ODE solving time, thereby shortening the overall running time of the proposed algorithm. It should be noted that the best response (BR) algorithm can also achieve an NE (or approximate NE) with fast convergence. The BR algorithm converges quickly in potential games [44]; outside that class, convergence is not guaranteed. In most cases, the N-player normal-form game Γ_s^{NFG} is not a potential game. Therefore, the BR algorithm cannot be applied directly to obtain the approximate NE.

Let $e = 1, \ldots, N_{eps}$ denote the episode counter in the whole learning process, where N_{eps} is the maximum number of episodes. We set up the threshold δ_e to terminate the process of solving the ODE for all the states in episode e when the following condition is satisfied:

$$||\dot{\boldsymbol{x}}(s)|| \leq \delta_e, \forall s \in \mathcal{S}_e, \tag{18}$$

where $\mathcal{S}_e$ represents the set containing all the states visited in the eth episode. Alternatively, the ODE solving accuracy can also be controlled by setting the maximum allowed number of iterations (denoted by N_{iter}). To improve the performance, the value of δ_e should shrink as the learning process proceeds. Therefore, we propose to use $\delta_{ini} = 0.1$ as the initial value for δ and let it decay over the episode as follows:

$$\delta_{e+1} = \delta_e \kappa, \forall e = 1, \ldots, N_{eps} - 1 \tag{19}$$

where κ is the shrinking rate for δ, which is set as 0.99in this study. We summarize the proposed Pop-Q algorithm with inheritance-based BNN dynamic in Algorithm 3. The practical implementation of this algorithm benefits from a logically centralized control architecture that facilitates network-wide state collection and coordination. For instance, the next-generation communication networks can serve as an implementation platform through software-defined networking (SDN) architectures [45], [46]. In this context, an SDN controller provides the centralized intelligence needed to gather global network state information while making coordinated decisions. A scheduling agent embedded within the SDN control module can execute the Pop-Q algorithm to optimize resource allocation policies across multiple base stations [47], [48].

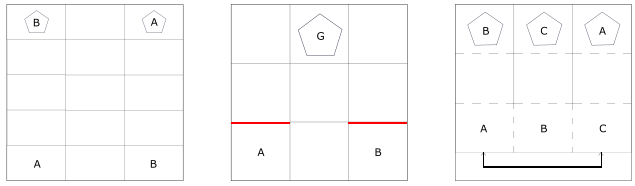
By setting $\boldsymbol{\xi}(s)$ and δ_e at suitable values, we enable the population state to evolve into a stationary state which represents an approximate NE at each state in the stochastic game. This helps increase the overall convergence rate of the Pop-Q algorithm.

V. Experiments

In this section, we evaluate the performance of the proposed Pop-Q algorithm using three grid-world games, which are widely used in evaluating the performance of equilibrium-based MARL algorithms [18], [19].

A. Test Case #1: Grid-World Game I

Baselines: We empirically compared the performance of our Pop-Q with Nash-Q [17] and CE-Q [16]. The Nash-Q algorithm solves the exact NE policy for the agents at each system state using the Lemke-Howson method [17]. The expected payoffs at the computed NE are used to update the agents' Q-functions.



(a) (b) (c)

Fig. 3. (a) Grid game I; (b) Grid game II; and (c) grid game III. The symbol(s) A, B, C, and G at the bottom indicates the positions where the agents are initially located, while the symbols in the pentagons on the top indicates the locations of the agents' goals. The two red lines in (b) represent the two barriers. In (c), agents can directly move between the two grids connected by a double arrow line. In addition, agents cannot move across a solid line but can move across a dotted line.

The CE-Q algorithm finds the CE policy by solving linear programming and updates the agents' Q-functions based on the computed CE.

Settings: As shown in Fig. 3(a), we consider a grid game with two agents and two distinct goals. At the initial state, we allocate two agents at the corners of the bottom, whose goals are located at the corners of the top, respectively. At each stage of the game, each agent can move up, left, right, down, or stay at the same place but cannot move out of the 3×5 world. If the two agents try to move to the same grid, then the moves are not valid, and they will stay where they are. If anyone of the two agents reaches its goal, the game ends. An agent receives a negative reward of -1 at each valid move, so each agent minimizes the number of movements before reaching his goal. If an agent tries to move out of the world, it receives a -5 reward. If two agents try to move to the same grid, they both receive a -5 reward. An agent who reaches his goal at the end of the game gets a 100 reward.

The proposed Pop-Q algorithm and the CE-Q algorithm use parameter settings $\alpha = 0.99$, $\epsilon = 0.99$, and $\gamma = 0.9$. The Nash-Q algorithm adopts the parameter settings $\alpha = 0.5$, $\gamma = 0.9$, and $\epsilon = 0.1$. These parameter values are determined by experiences. For all three algorithms, we decay the learning rate α and exploration rate ϵ geometrically by $\eta = 0.999$ after each episode.

Metrics: We use four metrics to evaluate the performance of the proposed Pop-Q algorithm: (1) Average ODE solving time; (2) Maximum loss; (3) Number of steps per episode; and (4) Average reward per step. They are defined as follows:

We define $\bar{T}_e^{\text{ODE}}$ as the average ODE solving time in episode e (where $e = 1, 2, \ldots, N_{eps}$) as follows:

$$\bar{T}_e^{\text{ODE}} = \frac{\sum_{s \in \mathcal{S}_e} T_e^{\text{ODE}}(s)}{|\mathcal{S}_e|}, \tag{20}$$

where $T_e^{\text{ODE}}(s)$ denotes the ODE solving time corresponding to state s in episode e, and $\mathcal{S}_e$ represents the set containing all the states experienced by the agents in episode e.

We define the maximum loss of agent i as

$$\mathcal{L}_i^e = \max_{(s,\boldsymbol{a}) \in \Omega_e} |Q_i^e(s, \boldsymbol{a}) - Q_i^{e-1}(s, \boldsymbol{a})|, \tag{21}$$

where Ω_e represents the set containing all the state-action pairs experienced by all the agents in episode e. The value of $\mathcal{L}_i^e$ shows the maximum difference of the Q-values between two successive

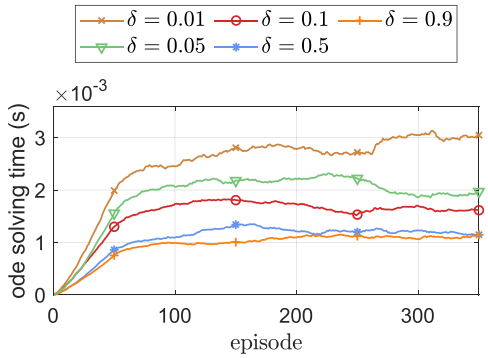


Fig. 4. Running time per episode of the proposed Pop-Q algorithm with different threshold δ values.

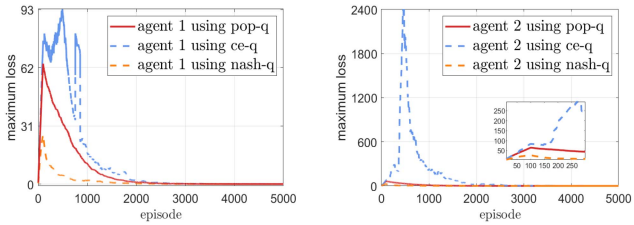


Fig. 5. Maximum loss of Q-functions in grid game I: (a) Loss function of agent 1. (b) Loss function of agent 2.

episodes for agent i, which is an indicator for the convergence property of the three learning algorithms.

The number of steps per episode for agent i in episode e is simply the number of steps taken by agent i in episode e. The average reward per step for agent i in episode e is defined as the sum of rewards obtained by agent i in episode e over the number of steps taken by agent i in episode e.

All performance metrics are calculated as the average of 10 experiments to avoid randomness in the results.

Results: We first study how the running time of solving ODE changes with the threshold value δ for the proposed Pop-Q algorithm. We choose the threshold value from the set of $\{0.01, 0.05, 0.1, 0.5, 0.9\}$ and keep δ as a constant through the whole learning process. For each choice, we run the proposed Pop-Q algorithm with 350 episodes (i.e., $N_{eps} = 350$). For each episode, we record the running time for solving the ODE per state and calculate $\bar{T}_e^{\text{ODE}}$ based on (22), as shown in Fig. 4. From the figure, we observe that the average running time for solving the ODE decreases as the value of δ increases, as a larger δ indicates a smaller number of iterations. This allows us to achieve the proper trade-off between system performance and convergence speed through adjusting the value of δ.

Next, we evaluate the convergence performance of the proposed algorithm over several metrics, i.e., maximum loss, number of steps per episode, and average reward per step. We choose the maximum number of episodes to be 5000 for all three algorithms and set the value of δ to 0.1. Fig. 5 shows the value of the maximum loss $\mathcal{L}_i^e$ defined in (21) of the two agents' for the three algorithms. We observe that $\mathcal{L}_i^e$ converges to zero within about 2000 episodes for all three algorithms. This demonstrates

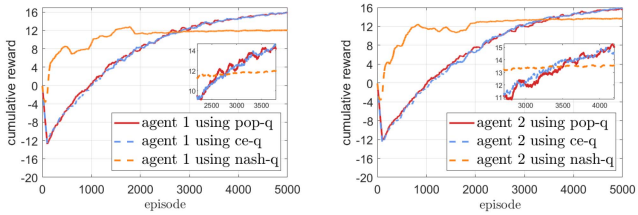


Fig. 6. Convergence performance of the three algorithms in grid game I: (a) Average reward per step of agent 1. (b) Average reward per step of agent 2.

that the proposed Pop-Q algorithm converges well when applied to this grid game. For this two-agent setting, the convergence rate of the Pop-Q algorithm is similar to the two baseline algorithms.

The agents' policy choices affect the total rewards agents obtain in each episode. For this two-agent grid game, there are several NE policies in which the agents coordinate their behavior to yield an average reward per step of (100-6)/6=15.67. To illustrate agents' policy choices after convergence, we show the agents' average rewards per step of the three algorithms in Fig. 6. We have two observations. First, the average reward per step increases as the learning proceeds for all three algorithms. Second, both the proposed algorithm and the CE-Q algorithm achieve the NE policies with an average reward per step of 15.67. The Nash-Q algorithm, however, achieves a lower average reward for both agents. This implies that although the Pop-Q algorithm only computes the approximate NE at each state during the learning process, it eventually still achieves the NE policies for the agents in this grid game. Although Pop-Q and CE-Q have a similar performance in this game, the advantages of the Pop-Q algorithm over the CE-Q algorithm will be more apparent in a three-agent setting in Section V-C.

B. Test Case #2: Grid-World Game II

Baselines and metrics: We use the same baselines and metrics as the first grid-game experiment.

Settings: We consider a 3×3 square grid as shown in Fig. 3(b). In this grid game, there are two agents, one goal, and two barriers (red lines). The two agents are allocated at the left and right corners of the bottom at the initial state, respectively. Their common goal is located in the middle of the top. If an agent attempts to move across a barrier, the agent will fail with a probability of 0.5. The reward settings of this game are the same as that in Grid Game I. Note that this game has probabilistic transitions due to the barriers, while Game I only has deterministic moves. We intend to use this experiment to demonstrate that the proposed Pop-Q algorithm can achieve the equilibrium policies in a game with probabilistic transitions. We use the same hyperparameter settings as the first experiment.

Results: We first demonstrate the convergence of the proposed Pop-Q algorithm with the maximum loss defined in (21). The maximum number of episodes is 5000 for all three algorithms. Fig. 7 shows the maximum loss of the two agents' Q-functions. We can observe that all three algorithms converge after about 4000 episodes.

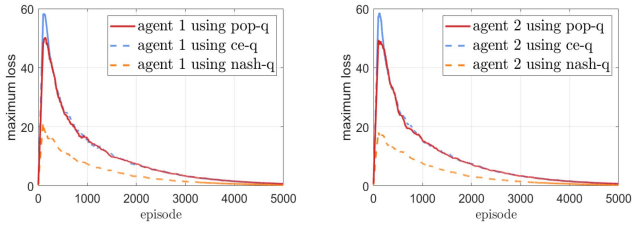


Fig. 7. Maximum loss of Q-functions in grid game II: (a) Loss function of agent 1. (b) Loss function of agent 2.

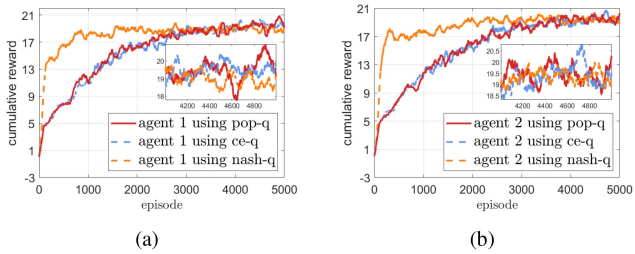


Fig. 8. Convergence performance of the three algorithms in grid game II: (a) Average reward per step of agent 1. (b) Average reward per step of agent 2.

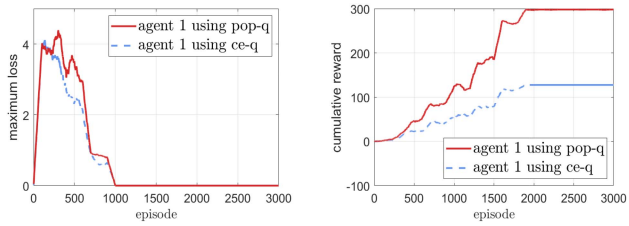


Fig. 9. Convergence performance comparison between Pop-Q and CE-Q in grid game III: (a) Loss function of agent 1 with the two algorithms. (b) Cumulative reward of agent 1 with the two algorithms.

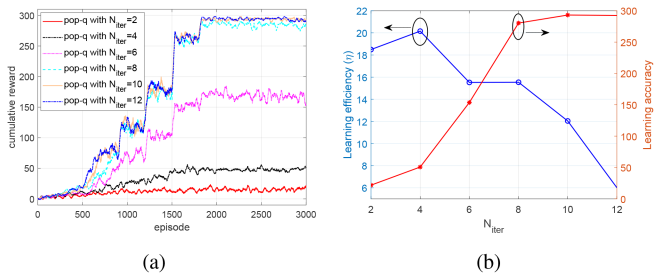


Fig. 10. (a) Cumulative reward of agent 1 with different values of N_{iter}. (b) The trade-off between learning efficiency and accuracy controlled by N_{iter}.

We next show the two agents' average rewards per step using the three algorithms in Fig. 8. We have two observations. First, the average reward per step increases as the learning proceeds for all three algorithms. Second, the reward obtained by the proposed Pop-Q learning algorithm and that by the CE-Q algorithm are quite similar, higher than that of the Nash-Q algorithm. This implies that with computing the approximate NE at each state only, the proposed algorithm achieves similar performance compared with the CE-Q algorithm (which computes the exact CE) while outperforming the Nash-Q algorithm (which computes the exact NE).

C. Test Case #3: Grid-World Game III

Baselines and metrics: We use the CE-Q algorithm as the baseline and the cumulative reward as the comparison metric for this game. The cumulative reward for agent i in episode e is the sum of rewards obtained by agent i over all the state-action pairs in episode e.

Settings: As shown in Fig. 3(c), we consider a three-agent traffic game, where three agents exist in a 3×3 grid world. At each non-ending state (no agent reaches his goal), agents respectively receive a negative reward of -1. If any agent moves across his initial location, he receives 50 bonus points. If any agent reaches his goal, he receives 100 bonus points, and the game ends. Any grid can hold an arbitrary number of agents. If two agents move to the same grid, they both receive a negative reward of -2. If the three agents move to the same grid, they all receive a negative reward of -3.

For the CE-Q algorithm and the proposed Pop-Q algorithm, the parameter settings are $\alpha = 0.3$, $\epsilon = 0.5$, and $\gamma = 0.3$. We let the learning rate α and the exploration rate ϵ decay by 0.08 after each episode according to experience.

Results: Fig. 9(a) compares the performance between the CE-Q algorithm and the proposed Pop-Q algorithm in terms of the maximum loss function of agent 1. We observe that both algorithms converge after around 1000 episodes. Fig. 9(b) shows the cumulative rewards of agent 1 for the two algorithms (the cumulative rewards of the other two agents exhibit highly similar tendencies). We observe that the proposed Pop-Q algorithm achieves the NE policies (at different states) with a reward of 300 for all the agents. The CE-Q algorithm, however, is trapped into a locally optimal policy for each agent with a reward lower than 150. This is because the objective of CE-Q is to maximize the sum of the two agents' rewards, which is not well-suited in this non-cooperative setting.

To examine how the number of ODE solving iterations (N_{iter}) influences learning complexity and accuracy in our proposed Pop-Q algorithm, we conducted additional experiments on Grid-World Game III using various N_{iter} values (2, 4, 6, 8, 10, and 12). For each value, we executed the Pop-Q algorithm ten times, calculating the average cumulative reward per episode across all three agents while measuring the total learning time per experiment. We define the average learning time across ten experiments as $\bar{T}_{\text{learn}}$ and introduce learning efficiency (η) as inversely proportional to $\bar{T}_{\text{learn}}$ as follows (with $K = 1000$):

$$\eta = \frac{K}{\bar{T}_{\text{learn}}}. \tag{22}$$

We define learning accuracy as the final cumulative reward achieved by Agent 1 upon convergence. Fig. 10(a) displays Agent 1's cumulative reward during Pop-Q learning with different N_{iter} values (results for Agents 2 and 3 are similar and omitted for brevity). Fig. 10(b) demonstrates the trade-off between learning efficiency and accuracy as modulated by N_{iter}. Our analysis reveals two key findings: First, Fig. 10(a) shows that

Agent 1's final cumulative reward (learning accuracy) increases in N_{iter} but plateaus at $N_{\text{iter}} = 10$. Second, Fig. 10(b) indicates that as N_{iter} increases, learning efficiency generally decreases while learning accuracy improves, with accuracy stabilizing at $N_{\text{iter}} = 10$. Based on these results, we determine that $N_{\text{iter}} = 10$ represents the optimal value for balancing learning complexity and accuracy in this grid game scenario.

VI. Conclusion

We have proposed a novel population game-based MARL algorithm to allow agents to learn equilibrium policies in non-cooperative environments. By formulating the normal-form game at each state into a population game and devising an efficient population dynamic, we have achieved a better trade-off between performance and complexity than existing equilibrium-based MARL frameworks. The proposed Pop-Q algorithm holds great potential to promote the development of modern systems/networks requiring decentralized decision-making and coordination, including the coordination of self-driving cars and traffic light optimization in traffic systems, multi-robot control in warehouse automation systems, trajectory planning and resource allocation in multi-AAV-aided communication systems, and dynamic resource management and load balancing in Vehicle-to-Everything communication networks, among others. The potential performance gains can be reduced traffic congestion, faster task completion, higher bandwidth utilization, and improved transmission latency, etc.

In this study, we develop our proposed Pop-Q algorithm based on tabular Q-learning, as it is more suitable for simple environments with discrete states/actions like grid-games. Real-world 6 G applications often have raw sensory inputs and therefore face high-dimensional state/action spaces. For future work, we will consider incorporating value function approximators (i.e., neural networks) into the proposed Pop-Q algorithm to handle these complex application scenarios. By replacing the Q-table in Pop-Q by a deep neural network, Pop-Q can be applied to complex environments. For instance, convolutional neural networks (CNNs) can be used for spatial data processing in grid-based environments, and recurrent neural networks (RNNs) or transformers can be employed for handling sequential decision processes with temporal dependencies. Nevertheless, advanced training strategies (such as experience replay and target networks) may be needed to address convergence issues when population ODEs interact with fluctuating neural network outputs during training, and the computational complexity of tracking population distributions across high-dimensional action spaces. Furthermore, we will also consider using concrete network examples (such as traffic systems and multi-AAV-aided communication systems) to demonstrate the performance of the proposed Pop-Q algorithm in future studies.

Appendix
Convergence Analysis of Pop-Q Learning

In this appendix, we provide the convergence analysis of the Pop-Q learning algorithm by showing that it has the same asymptomatic behavior as a singular ODE system and then show the ODE system converges to the equilibrium policy and Nash-Q values under suitable assumptions.

A. Slow-Fast Motion by Singular ODE System

In this subsection, we show that the proposed Pop-Q algorithmic framework can be approximated by a slow-fast motion system.

During the training process, the population state proceeds according to the deterministic approximation of the stochastic evolutionary process in the limit $M \to \infty$ to an approximate equilibrium of the ODE system for fixed Q-values. Q-values get updated using an approximately equilibrated population state. Thus, we may expect that the iteration of $\boldsymbol{x}_t$ and Q_t to track a singular ODE system in the form of

$$\begin{aligned} \epsilon \dot{\boldsymbol{x}}_\epsilon(t) &= H(\boldsymbol{x}_\epsilon(t), Q_\epsilon(t)) \\ \dot{Q}_\epsilon(t) &= G(\boldsymbol{x}_\epsilon(t), Q_\epsilon(t)) \end{aligned} \tag{23}$$

in the limit $\epsilon \to 0$, where $\epsilon \ll 1$ controls the difference in the time-scale between $\boldsymbol{x}_t$ and Q_t and

$$H^k_{a_k,s}(x, Q) = [\widetilde{F}^k_{a_k}(\boldsymbol{x}(s))]_+ - x^k_{a_k}(s) \sum_{a'_k \in \mathcal{A}_k} [\widetilde{F}^k_{a'_k}(\boldsymbol{x}(s))]_+$$

$$\begin{aligned} G^k_{\mathbf{a},s}(x, Q) &= r^k(\mathbf{a}, s) \\ &\quad + \beta \sum_{s' \in \mathbb{S}} p(s') \sum_{\mathbf{a}'} Q^k_{s',\mathbf{a}'} \Pi^n_{k'=1} x^{k'}(s', \mathbf{a}'_{k'}) \end{aligned}$$

Intuitively, the fast motion $\boldsymbol{x}_\epsilon(t)$ see the slow motion $Q_\epsilon(t)$ as quasi constant, while $Q_\epsilon(t)$ see $\boldsymbol{x}_\epsilon(t)$ as quasi equilibrated. More precisely, under sufficient regularity conditions, $(\boldsymbol{x}_\epsilon(t), Q_\epsilon(t))$ converges to a slow manifold

$$\mathcal{M} = \{(\boldsymbol{x}, Q) : Q \in \mathbb{Q}, H(\boldsymbol{x}, Q) = 0\} \tag{24}$$

and the effective dynamic of the slow variable $Q_\epsilon(t)$ is

$$\dot{Q}(t) = G(\lambda(Q(t)), Q(t)) \tag{25}$$

where the mapping $\lambda : \mathbb{Q} \to \mathbb{X}$ satisfies $H(\lambda(Q), Q) = 0, \forall Q \in \mathbb{Q}$. As $\epsilon \to 0$, $(\boldsymbol{x}_\epsilon(t), Q_\epsilon(t))$ converges to the slow manifold exponentially, starting from an arbitrary initial condition, that is,

$$\|\boldsymbol{x}_\epsilon(t) - \lambda(Q_\epsilon(t))\| \le M \|\boldsymbol{x}_\epsilon(0) - \lambda(Q_\epsilon(0))\| \exp(-kt) \tag{26}$$

as shown by Theorem A.1.

Theorem A.1 (Convergence towards a slow manifold): Let $\mathcal{M} = \{(x, y) : x = x^*(y), y \in D_0\}$ be an uniformly asymptotically stable slow manifold of the slow-fast system (23) and let $f, g \in C^2_b$ in a neighbourhood $\mathcal{N}$ of $\mathcal{M}$. Then there exist positive constant $\epsilon_0, c_0, c_1, k, M$, such that, for $0 < \epsilon \le \epsilon_0$ and any initial condition $(x_0, y_0) \in \mathcal{N}$ satisfying $\|x_0 - x^*(y_0)\| \le c_0$, the bounded

$$\|x_t - x^*(y_t)\| \le M \|x_0 - x^*(y_0)\| \exp(-kt) + c_1 \epsilon$$

holds as long as $y_t \in D_0$.

B. Asymptotic Behavior of Pop-Q

Then, we show that the Pop-Q learning algorithm has the same asymptomatic behavior as the ODE system (23) by showing that it is a (T, δ)-perturbation of the ODE system. We need to state following definitions and lemmas first.

Definition A.2 ((T, δ)-perturbation): For an ODE

$$\dot{x} = h(x(t)), \quad x \in \mathbb{R}^d \tag{27}$$

given $T, \delta > 0$, its (T, δ)-perturbation is a bounded measurable function $y : [0, \infty) \to \mathbb{R}^d$ such that there exists $0 = T_0 < T_1 < \cdots < T_n \to \infty$ and solutions $x^j(t), t \in [T'_j, T'_{j+1}], j \geq 0$ of ODE (27) such that $T'_{j+1} - T'_j = T_{j+1} - T_j \geq T$ for $j \geq 0$ and $\|y(t) - x^j(T'_j + t - T_j)\| \leq \delta$.

In addition,

Lemma A.3: Suppose that the ODE (27) has an asymptotically stable attractor J, then its any (T, δ)-perturbation converges to an ϵ-neighborhood of J, that is, $y(t) \to J^\epsilon$.

Proof: See [49].

Then we are ready to state our first lemma about the asymptotic behavior of the Pop-Q learning algorithm.

Lemma A.4: For any $T, \delta > 0, \exists t(\delta) > 0$, such that $(X(t(\delta) + \cdot), Q(t(\delta) + \cdot))$ is a (T, δ)-perturbation of the following ODE system

$$\begin{aligned} \epsilon \dot{X}_\epsilon(t) &= H(X_\epsilon(t), Q_\epsilon(t)) \\ \dot{Q}_\epsilon(t) &= 0 \end{aligned}$$

Proof: Let $\tilde{x}(\tau) = X(t(\delta) + \tau)$ and $\tilde{q}(\tau) = Q(t(\delta) + \tau)$, and

$$\tilde{x}(0) = X_\epsilon(0), \quad \tilde{q}(0) = Q_\epsilon(0)$$

Following the proof of Lemma 2.2 in [50], we have $\exists t(\delta) > 0$ such that $\|q(t) - q(0)\| = \|q(t) - Q_\epsilon(t)\| \to 0$ for t in finite time interval $[0, T]$ as $t(\delta) \to \infty$. Then the lipschitz continuity of F guarantee that

$$\|H(\tilde{x}_t, q(t)) - H(\tilde{x}_t, Q_\epsilon(t))\| \to 0$$

Then the desired result follows.

A direct corollary is given below

Corollary A.5: $(X_t, Q_t) \to \mathcal{M} = \{(\lambda(Q), Q), Q \in \mathbb{Q}\}$ almost surely as $t \to \infty$.

Proof: Immediate from Lemmas A.3 and A.4.

Furthermore, we assume that the mapping $\lambda(Q) : \mathbb{Q} \to \mathbb{X}$ is Lipschitz continuous and an unique global asymptotically stable equilibrium of

$$\dot{X}_\epsilon(t) = G(X_\epsilon(t), Q), \quad \forall Q \in \mathbb{Q}$$

and $\lambda(Q)$ is either a global optimum or a saddle point of the stage game defined by Q.

Then, we show that the asymptotic behavior of Q values in Pop-Q algorithms tracks the effective dynamic of the singular ODE system (23).

Lemma A.6: For any $T, \delta > 0, \exists t(\delta) > 0$, such that $X(t(\delta) + \cdot)$ is a (T, δ)-perturbation of the effective dynamic described by the following ODE

$$\dot{Q}_\epsilon(t) = G(\lambda(Q_\epsilon(t)), Q_\epsilon(t)) \tag{28}$$

Proof: As Lemma A.4.

A similar corollary immediately follows

Corollary A.7: If the effective dynamic (28) has a global asymptotically stable attractor J, the population state and Q values in Pop-Q learning algorithm $(X_t, Q_t) \to (\lambda(Q_*), Q_*)$ for some $Q_* \in J$.

In addition, with the help of the assumptions on $\lambda : \mathbb{Q} \to \mathbb{X}$, we can prove the following lemma about effective dynamic described by the (28).

Lemma A.8: The ODE

$$\dot{Q}_\epsilon(t) = G(\lambda(Q_\epsilon(t)), Q_\epsilon(t))$$

has an unique global asymptotically stable equilibrium Q_* and Q_* coincides with the Nash-Q values.

Proof: For $\boldsymbol{x} = \lambda(Q_\epsilon(t))$ as a feasible population state, we have

$$\begin{aligned} &G(\boldsymbol{x}, Q_\epsilon(t)) \\ &\quad = r^k(s, a_1, \ldots, a_n) + \beta \left(\sum_{s' \in S} p(s') \operatorname{pop}_t^k(s') \right. \\ &\qquad \left. - Q^k(s, a_1, \ldots, a_n) \right) \\ &\quad = \beta \sum_{s' \in S} p(s') \left(\mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t r^k(s_t, a_1, \ldots, a_n) | s_0 = s', \boldsymbol{x} \right] \right. \\ &\qquad \left. - \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t r^k(s_t, a_1, \ldots, a_n) | s_0 = s', \boldsymbol{x} \right] \right) \\ &\quad = 0 \end{aligned}$$

where

$$\operatorname{pop}_t^k(s_t) = \sum_{(a_1, \ldots, a_n) \in A} \prod_{k=1}^{n} \boldsymbol{x}(s_t, a_k) \cdot Q_t^k(s, a_1, \ldots, a_n)$$

and the Jacobian matrix of $g(\boldsymbol{x}, \lambda(\boldsymbol{x}))$ given by

$$\begin{aligned} &J^*(\boldsymbol{x})_{Q^k(s, a_1, \ldots, a_n), Q^k(s^*, a_1^*, \ldots, a_n^*)} \\ &\quad = \partial_{Q_t^k(s^*, a_1^*, \ldots, a_n^*)} g(\boldsymbol{x}, \lambda(\boldsymbol{x}))_{Q^k(s, a_1, \ldots, a_n)} \\ &\quad = \beta p(s^*) \prod_{k=1}^{n} \boldsymbol{x}(s, a_k) - I_{s^* = s, a_1 = a_1^*, \ldots, a_n = a_n^*} \end{aligned}$$

The matrix is strictly negative for all diagonal elements and strictly positive for all off-diagonal elements. In addition, the sum of all of its elements is equal to

$$n \times |\mathbb{S}| \times |\mathbb{A}_1| \times \cdots \times |\mathbb{A}_n| (\beta - 1) < 0$$

Therefor, the sum of absolute values of diagonal elements is strictly larger than the sum of absolute values of off-diagonal elements. Then, by the property of diagonally dominant matrix, all its eigenvalues have strictly negative real parts. Then we conclude that Nash Q-values Q_* is asymptotically stale. The proof is completed. ■

C. ODE Approximation Error

In each episode, for fixed Q-values $Q \in \mathbb{Q}$ and O.D.E of population state

$$\dot{X}_\epsilon(t) = H(X_\epsilon(t), Q_\epsilon)$$

the algorithm doesn't solve above ODE exactly, instead it converges to an ϵ-neighborhood of the fixed point of above ODE, that is

$$X_\epsilon(t) \to \mathcal{N}_\epsilon(X^*), \quad H(X^*, Q) = 0$$

Therefore, pair of population state and Q-values (X_t, Q_t) doesn't exactly converges to the slow manifold $\mathcal{M}$ defined above, instead it converges to an adiabatic manifold $\mathcal{M}_\epsilon$ defined by

$$\mathcal{M}_\epsilon = \{(X, Q) : X = \lambda(Q, \epsilon), Q \in \mathbb{Q}\}$$

where $\lambda(Q, \epsilon) = \lambda(Q) + O(\epsilon)$. And the existence of such adiabatic manifold is guaranteed [51]. The effective dynamic of Q_t on the adiabatic manifold $\mathcal{M}_\epsilon$ is governed by

$$\dot{Q}_\epsilon(t) = G(Q_\epsilon(t), \lambda(Q_\epsilon(t), \epsilon))$$

and it is guaranteed that [51]

$$\|(X(t), Q(t)) - (X_\epsilon(t), Q_\epsilon(t))\| \leq M\|(X(0), Q(0)) - (X_\epsilon(0), Q_\epsilon(0))\| \exp\left(-\frac{kt}{\epsilon}\right)$$

where $(X(t), Q(t))$ is the solution of the ODE system (23) in the limit $\epsilon \to 0$. As the effective dynamic of $(X(t), Q(t))$ on the slow manifold $\mathcal{M}$ is governed by

$$\dot{Q}(t) = G(\lambda(Q(t)), Q(t)), \quad X(t) = \lambda(Q(t))$$

and converges to the Nash equilibrium and the Nash Q-values (π^*, Q_*). Therefore, a direct corollary follows.

Corollary A.9: $\exists T > 0$, such that, $\forall t \geq T$, we have

$$\|(X_\epsilon(t), Q_\epsilon(t)) - (\pi^*, Q_*)\| \leq \bar{M} \exp\left(-\frac{kt}{\epsilon}\right)$$

where $\bar{M}$ is a positive constant depending on T but independent of ϵ.

D. Main Result

Now, we are ready to state our theorem on the convergence of Pop-Q learning algorithm.

Theorem A.10: Suppose that the following conditions hold

1) All states and actions are visited infinitely often;
2) $$\sum_{t=0}^{\infty} \alpha(t) = \infty, \qquad \sum_{t=0}^{\infty} \alpha^2(t) < \infty;$$
3) The Lipschitz continuous mapping $\lambda(Q) : \mathbb{Q} \to \mathbb{X}$ is an unique global asymptotically stable equilibrium of
$$\dot{\boldsymbol{x}}_\epsilon(t) = G(\boldsymbol{x}_\epsilon(t), Q), \quad \forall Q \in \mathbb{Q}$$
and $\lambda(Q)$ is either a global optimum or a saddle point of the stage game defined by Q.

Then, the population state and Q values in the Pop-Q learning algorithm $(\boldsymbol{x}(t), Q(t))$ converges to equilibrium policy of the stochastic game and Nash-Q values (π_*, Q_*) almost surely.

REFERENCES

[1] W. Zhang, T. Zhao, Z. Zhao, Y. Wang, and F. Liu, "An intelligent strategy decision method for collaborative jamming based on hierarchical multi-agent reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 10, no. 4, pp. 1467–1480, Aug. 2024.

[2] Z. Chang, H. Deng, L. You, G. Min, S. Garg, and G. Kaddoum, "Trajectory design and resource allocation for multi-UAV networks: Deep reinforcement learning approaches," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 2940–2951, Sep./Oct. 2023.

[3] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, Mar. 2021.

[4] B. Lu, Y. Wu, L. Qian, S. Zhou, H. Zhang, and R. Lu, "Multi-agent DRL-based two-timescale resource allocation for network slicing in V2X communications," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 6, pp. 6744–6758, Dec. 2024.

[5] K. K. Nguyen, T. Q. Duong, T. Do-Duy, H. Claussen, and L. Hanzo, "3D UAV trajectory and data collection optimisation via deep reinforcement learning," *IEEE Trans. Commun.*, vol. 70, no. 4, pp. 2358–2371, Apr. 2022.

[6] J. Ji, K. Zhu, and L. Cai, "Trajectory and communication design for cache-enabled UAVs in cellular networks: A deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 6190–6204, Oct. 2023.

[7] X. He, S. Pang, H. Gui, K. Zhang, N. Wang, and X. Zhai, "Multi-agent DRL-based large-scale heterogeneous task offloading for dynamic IoT systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 12, no. 2, pp. 982–996, Mar./Apr. 2025.

[8] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 729–743, Feb. 2020.

[9] S. Cheng, X. Lin, X. Li, and J. Wang, "Joint UAV trajectory and RadCom task schedule for IVNs: A game-embedding multi-agent deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 24, no. 1, pp. 181–196, Jan. 2025.

[10] J. Wu et al., "Resource allocation for delay-sensitive vehicle-to-multi-edges (V2Es) communications in vehicular networks: A multi-agent deep reinforcement learning approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1873–1886, Apr.–Jun. 2021.

[11] H. Peng and X. Shen, "Multi-agent reinforcement learning based resource management in MEC-and UAV-assisted vehicular networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 131–141, Jan. 2021.

[12] K. Griparic, M. Polic, M. Krizmancic, and S. Bogdan, "Consensus-based distributed connectivity control in multi-agent systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 3, pp. 1264–1281, May/Jun. 2022.

[13] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A very condensed survey and critique of multiagent deep reinforcement learning," in *Proc. 19th Int. Conf. Auton. Agents MultiAgent Syst.*, 2020, pp. 2146–2148.

[14] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," in *Handbook of Reinforcement Learning and Control*. Cham, Switzerland: Springer, 2021, pp. 321–384.

[15] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6201–6213, Jul. 2020.

[16] A. Greenwald, K. Hall, and R. Serrano, "Correlated Q-learning," in *Proc. Int. Conf. Mach. Learn.*, 2003, vol. 3, pp. 242–249.

[17] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Nov. 2003.

[18] Y. Hu, Y. Gao, and B. An, "Multiagent reinforcement learning with unshared value functions," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 647–662, Apr. 2015.

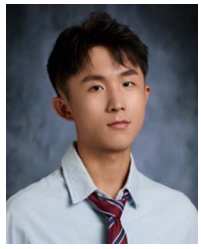
[19] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. Mach. Learn. Proc. 1994*, 1994, pp. 157–163.

[20] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a Nash equilibrium," *SIAM J. Comput.*, vol. 39, no. 1, pp. 195–259, 2009.

[21] W. H. Sandholm, *Population Games and Evolutionary Dynamics*. Cambridge, MA, USA: MIT Press, 2010.
[22] W. H. Sandholm, "Excess payoff dynamics and other well-behaved evolutionary dynamics," *J. Econ. Theory*, vol. 124, no. 2, pp. 149–170, Oct. 2005.
[23] W. Shi, J. Li, H. Wu, C. Zhou, N. Cheng, and X. Shen, "Drone-cell trajectory planning and resource allocation for highly mobile networks: A hierarchical DRL approach," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9800–9813, Jun. 2021.
[24] T. Cai et al., "Cooperative data sensing and computation offloading in UAV-assisted crowdsensing with multi-agent deep reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3197–3211, Sep./Oct. 2022.
[25] X. Xu, R. Li, Z. Zhao, and H. Zhang, "The gradient convergence bound of federated multi-agent reinforcement learning with efficient communication," *IEEE Trans. Wireless Commun.*, vol. 23, no. 1, pp. 507–528, Jan. 2024.
[26] T. Wu et al., "Joint traffic control and multi-channel reassignment for core backbone network in SDN-IoT: A multi-agent deep reinforcement learning approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 231–245, Jan.–Mar. 2021.
[27] J. Li, H. Wu, X. Huang, Q. Huang, J. Huang, and X. S. Shen, "Toward reinforcement-learning-based intelligent network control in 6G networks," *IEEE Netw.*, vol. 37, no. 4, pp. 104–111, Jul./Aug. 2023.
[28] M. L. Littman, "Friend-or-foe Q-learning in general-sum games," in *Proc. Int. Conf. Mach. Learn.*, 2001, vol. 1, pp. 322–328.
[29] Y. Jiang, S. Jiang, and X. Wang, "A model-based reinforcement learning algorithm for multi-agent cooperation Nash equilibrium with unstable communication," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 71, no. 11, pp. 4743–4747, Nov. 2024.
[30] Z. Lv, L. Xiao, Y. Du, Y. Zhu, S. Han, and Y. -J. Liu, "Efficient communications in multi-agent reinforcement learning for mobile applications," *IEEE Trans. Wireless Commun.*, vol. 23, no. 9, pp. 12440–12454, Sep. 2024.
[31] T. Yu, H. Z. Wang, B. Zhou, K. W. Chan, and J. Tang, "Multi-agent correlated equilibrium Q (λ) learning for coordinated smart generation control of interconnected power grids," *IEEE Trans. Power Syst.*, vol. 30, no. 4, pp. 1669–1679, Jul. 2015.
[32] K. Tuyls, P. J. Hoen, and B. Vanschoenwinkel, "An evolutionary dynamical analysis of multi-agent learning in iterated games," *Auton. Agents Multi-Agent Syst.*, vol. 12, no. 1, pp. 115–153, 2006.
[33] S. Tan, Y. Wang, and A. V. Vasilakos, "Distributed population dynamics for searching generalized Nash equilibria of population games with graphical strategy interactions," *IEEE Trans. Syst. Man Cybern., Syst.*, vol. 52, no. 5, pp. 3263–3272, May 2022.
[34] Y. Ming, J. Chen, Y. Dong, and Z. Wang, "Evolutionary game based strategy selection for hybrid V2V communications," *IEEE Trans. Veh. Technol.*, vol. 71, no. 2, pp. 2128–2133, Feb. 2022.
[35] S. Tan, Z. Fang, Y. Wang, and J. Lü, "Consensus-based multipopulation game dynamics for distributed Nash equilibria seeking and optimization," *IEEE Trans. Syst. Man Cybern., Syst.*, vol. 53, no. 2, pp. 813–823, Feb. 2023.
[36] Y. Ma, K. Liu, Y. Liu, X. Wang, and Z. Zhao, "An intelligent game-based anti-jamming solution using adversarial populations for aerial communication networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 11, no. 3, pp. 1981–1995, Jun. 2025.
[37] L. Matignon, G. J. Laurent, and N. L. Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," *Knowl. Eng. Rev.*, vol. 27, no. 1, pp. 1–31, Feb. 2012.
[38] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 279–292, May 1992.
[39] G. Perelli, "Enforcing equilibria in multi-agent systems," in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2019, pp. 188–196.
[40] J. Hofbauer, J. Oechssler, and F. Riedel, "Brown–von Neumann–Nash dynamics: The continuous strategy case," *Games Econ. Behav.*, vol. 65, no. 2, pp. 406–429, 2009.
[41] U. Berger and J. Hofbauer, "Irrational behavior in the Brown–Von Neumann–Nash dynamics," *Games Econ. Behav.*, vol. 56, no. 1, pp. 1–6, 2006.
[42] E. Suli and D. F. Mayers, *An Introduction to Numerical Analysis*, 2nd ed. Hoboken, NJ, USA: Wiley, 1989.
[43] Y. Hu, Y. Gao, and B. An, "Accelerating multiagent reinforcement learning by equilibrium transfer," *IEEE Trans. Cybern.*, vol. 45, no. 7, pp. 1289–1302, Jul. 2015.
[44] J. Li, W. Shi, Q. Ye, N. Zhang, W. Zhuang, and X. Shen, "Multiservice function chain embedding with delay guarantee: A game-theoretical approach," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11219–11232, Jul. 2021.
[45] S. -C. Lin, K. -C. Chen, and A. Karimoddini, "SDVEC: Software-defined vehicular edge computing with ultra-low latency," *IEEE Commun. Mag.*, vol. 59, no. 12, pp. 66–72, Dec. 2021.
[46] A. Papa et al., "MARC: On modeling and analysis of software-defined radio access network controllers," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4602–4615, Dec. 2021.
[47] J. Li, W. Shi, N. Zhang, and X. Shen, "Delay-aware VNF scheduling: A reinforcement learning approach with variable action set," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 304–318, Mar. 2021.
[48] H. Wu et al., "Resource management in space-air-ground integrated vehicular networks: SDN control and AI algorithm design," *IEEE Wireless Commun.*, vol. 27, no. 6, pp. 52–60, Dec. 2020.
[49] V. R. Konda and V. S. Borkar, "Actor-critic-type learning algorithms for Markov decision processes," *SIAM J. Control Optim.*, vol. 38, no. 1, pp. 94–123, 1999.
[50] V. S. Borkar, "Stochastic approximation with two time scales," *Syst. Control Lett.*, vol. 29, no. 5, pp. 291–294, 1997.
[51] N. Berglund and B. Gentz, *Noise-induced phenomena in slow-fast dynamical systems: A sample-paths approach*. Berlin, Germany: Springer, 2006.

Junling Li (Member, IEEE) received the B.S. degree from Tianjin University, Tianjin, China, in 2013, the M.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2016, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2020. She was a Joint Postdoctoral Research Fellow with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, University of Waterloo, and The Chinese University of Hong Kong, Shenzhen, China, from 2020 to 2022. She is currently an Associate Professor with National Mobile Communications Research Laboratory, Southeast University, Nanjing, China. Her research interests include digital twin, channel modeling, game theory, and machine learning-based channel prediction. She was the recipient of Best Paper Awards at the IEEE ICCC'19, ICCT'23, and ICC'25.

Hao Zhang received the B.Sc. degree in computational finance from The City University of Hong Kong, Hong Kong. He is currently a Quantitative Researcher specializing in algorithmic trading. His research interests include reinforcement learning, optimal control, and their applications in financial markets, with a particular focus on advancing techniques for algorithmic trading.



Shuqi Ke received the B.Sc. degree in mathematics from The Chinese University of Hong Kong, Shenzhen, China. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include self-supervised learning, generative models, game theory, and privacy-preserving machine learning. He was a reviewer of ICLR, NeurIPS, AISTATS, and IEEE TRANSACTIONS ON SIGNAL PROCESSING.

Jianwei Huang (Fellow, IEEE) is a currently the Presidential Chair Professor and Associate Vice President with The Chinese University of Hong Kong, Shenzhen, China. He is also the Associate Director with the Shenzhen Institute of Artificial Intelligence and Robotics for Society. He was the Editor-in-Chief of IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING and Associate Editor-in-Chief for IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY. He is an IEEE ComSoc Distinguished Lecturer, Clarivate Web of Science Highly Cited Researcher, and Elsevier Most Cited Chinese Researcher.

Nan Chen is currently a Professor of systems engineering and engineering management with The Chinese University of Hong Kong (CUHK), Hong Kong. He is also the Director with the Centre for Financial Engineering, CUHK. His research interests include financial engineering and FinTech, stochastic control, monte carlo simulation, and applied probability. He is also an Associate Editor for the journals of *Operations Research*, *Mathematical Finance*, and *Digital Finance*.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks. Dr. Shen is also a registered Professional Engineer with Ontario, Canada, Engineering Institute of Canada Fellow, Canadian Academy of Engineering Fellow, Royal Society of Canada Fellow, Chinese Academy of Engineering Foreign Member, and Engineering Academy of Japan International Fellow. Dr. Shen was the recipient of "West Lake Friendship Award" from Zhejiang Province in 2023, Canadian Award for Telecommunications Research from the Canadian Society of Information Theory in 2021, R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015, and Education Award in 2017 from the IEEE Communications Society (ComSoc), Technical Recognition Award from Wireless Communications Technical Committee in 2019, AHSN Technical Committee in 2013, Excellent Graduate Supervision Award in 2006 from the University of Waterloo, and the Premier's Research Excellence Award in 2003 from the Province of Ontario, Canada. He is/was the General Chair of 6 G Global Conference'23 and ACM Mobihoc'15, Technical Program Committee Chair/Co-Chair of IEEE Globecom'24, '16 and '07, IEEE Infocom'14, and IEEE VTC'10 Fall. Dr. Shen is also the Past President of IEEE ComSoc, Vice President of Technical & Educational Activities, Vice President of Publications, Memberat-Large on the Board of Governors, Chair of Distinguished Lecturer Selection Committee, and member of IEEE Fellow Selection Committee of ComSoc. Dr. Shen was the Editor-in-Chief of IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and *IET Communications*.