Received 3 December 2023, accepted 14 December 2023, date of publication 21 December 2023,
date of current version 28 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3345448

RESEARCH ARTICLE

Network-Wide Traffic Signal Control Based on MARL With Hierarchical Nash-Stackelberg Game Model

HUI SHEN[1,2], HONGXIA ZHAO[3], (Member, IEEE), ZUNDONG ZHANG[4], XUN YANG[4], YUTONG SONG[4], AND XIAOMING LIU[4]
[1]School of Electrical and Control Engineering, North China University of Technology, Beijing 100037, China
[2]Beijing Municipal Traffic Management Bureau, Beijing 100037, China
[3]State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[4]Beijing Key Laboratory of Urban Road Traffic Intelligent Technology, North China University of Technology, Beijing 100144, China

Corresponding author: Zundong Zhang (zdzhang@ncut.edu.cn)

This work was supported in part by the National Natural Science Foundation Project under Grant U1909204, in part by the China National Railway Group Company Science and Technology Research and Development Program Project under Grant L2022X002, in part by the Open Topic of the National Railway Intelligent Transportation System Engineering Technology Research Center under Grant RITS2021KF03, and in part by the Guangdong Provincial Key Area Research and Development Program Project under Grant 2020B0909050001.

ABSTRACT Network-wide traffic signal control is an important means of relieving urban congestion, reducing traffic accidents, and improving traffic efficiency. However, solving the problem of computational complexity caused by multi-intersection games is challenging. To address this issue, we propose a Nash-Stackelberg hierarchical game model that considers the importance of different intersections in the road network and the game relationships between intersections. The model takes into account traffic control strategies between and within sub-areas of the road network, with important intersections in the two sub-areas as the game subject at the upper layer and secondary intersections as the game subject at the lower layer. Furthermore, we propose two reinforcement learning algorithms (NSHG-QL and NSHG-DQN) based on the Nash-Stackelberg hierarchical game model to realize coordinated control of traffic signals in urban areas. Experimental results show that, compared to basic game model solving algorithms, NSHG-QL and NSHG-DQN algorithms can reduce the average travel time and time loss of vehicles at intersections, increase average speed and road occupancy, and coordinate secondary intersections to make optimal strategy selections based on satisfying the upper-layer game between important intersections. Moreover, the multi-agent reinforcement learning algorithms based on this hierarchical game model can significantly improve learning performance and convergence.

INDEX TERMS Network-wide traffic signal control, hierarchical game model, multi-agent reinforcement learning.

I. INTRODUCTION

Due to rapid urbanization, congestion caused by the growth of traffic demand has become a major strategic issue for sustainable and harmonious city development. Traffic flows at intersections in the urban road network are highly correlated, and an excellent coordination mechanism is essential to controlling traffic signals and relieving traffic congestion [1]. Game theory, a mathematical model for studying the interaction of strategies among rational decision-makers, is a suitable method for solving the problem of urban traffic signal coordination and control, enabling control strategies to better adapt to the dynamic changes in traffic demand levels [2], [3]. In recent years, an increasing number of researchers have focused on the traffic signal coordination control method combined with game theory. The Nash

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng Yan.

equilibrium in game theory provides a theoretical framework for the coordination of signals at multiple intersections in the road network, but it faces challenges in expanding to more intersections due to the explosion of dimensions. Additionally, there are differences in the importance of each intersection, which can cause goal conflict problems in traffic optimization when secondary intersections sacrifice traffic capacity for important intersections [4].

Currently, game theory is mainly used in traffic guidance and management, and its application in traffic signal timing decision-making is still in its infancy [5]. Alvarez et al. [6] aimed to minimize lane queue lengths and analyzed the game between intersections as a non-cooperative game. Zhao et al [7] proposed the update of the Q-Value function based on the Nash bargaining solution, and coordination control for two adjacent intersections was achieved. Shamshirband [8] adopted a two-person non-zero-sum game model for signal coordination control between intersections. Clempner et al. [9] formulated the multi-traffic-signal-control problem as a Stackelberg game-theory approach and solved it using a Nash equilibrium based on the extra proximal method. Zhao et al. [10] proposed an algorithm based on coordination game and Pareto efficiency, and simulation results demonstrated that the algorithm is more efficient than Webster's fixed-time plan and actuated control algorithm in average queue length, average total delay, and average travel time. Zhu et al. [11] proposed a bi-hierarchical game-theoretic method using trip-based data to solve the network-wide traffic signal control problem. The self-learning and interaction of multi-agent systems are similar to the urban road network structure, which has attracted the attention of many scholars on the application of multi-agent systems and their self-learning mechanisms in urban traffic signal timing decision-making [12]. Therefore, combining Multi-Agent Reinforcement Learning (MARL) with game theory has become the research trend for coordinated control of traffic signals.

In recent years, researchers have combined MARL with game theory to obtain effective and reasonable traffic signal control strategies by using equilibrium solutions in games instead of optimal solutions [13]. To effectively control the traffic congestion problem, a two-mode agent architecture is proposed to perform independent and cooperative procedures [13]. In the cooperative mode, game theory is employed to determine how cooperation between agents can dynamically control traffic signals at multiple intersections. Wu et al. [14] proposed the Nash-A2C algorithm and Nash-A3C algorithm based on the deep reinforcement learning (DRL) algorithm and Nash equilibrium theory. Guo et al. [15] combined game theory with Q-learning methods in reinforcement learning (RL) and proposed a semi-cooperative Nash/Stackelberg Q-learning algorithm for single intersection signals. Pan et al. [4] improved the decision-making process of the IA-MARL algorithm by incorporating the concept of mixed-strategy Nash equilibrium in game theory, and presented a multi-agent reinforcement learning framework considering game (G-MARL). Zhang et al. [16] introduced a multi-agent deep reinforcement learning algorithm based on Nash equilibrium. Mondal et al. [17] demonstrated that the K-class heterogeneous cooperative MARL problem can be approximated using a related mean-field control problem. They proposed an algorithm based on natural policy gradients to efficiently approximate the optimal MARL strategy in a sample-efficient manner. The average reward model allows for more heterogeneity among agents, preserving privacy and fostering the development of decentralized MARL algorithms. In a fully competitive stochastic game, the Minimax-Q algorithm utilizes the minimax principle to compute strategies and values for stage games. The deep Q-learning method introduced by Mohammadamin Moradi et al. [18] addresses the limitations of conventional approaches, demonstrating effectiveness in enhancing the security of large-scale power grids against cyberattacks.

However, Existing coordinated control methods based on Nash equilibrium are mostly oriented to single or two intersections [13].As the expansion continues, some studies have been limited to single-objective optimization by assuming that the global objective function of the entire network is a linear summation of multiple regional objective functions, which cannot reflect the variability among intersections and obtain the global optimum [16]. The model proposed in this paper not only reflects the importance of different intersections in the road network and the game relationships between intersections but also overcomes the problem of computational complexity for the Nash equilibrium caused by the multi-intersection game.

This paper proposes a Nash-Stackelberg hierarchical game model that introduces the concepts of Nash equilibrium and Stackelberg equilibrium in game theory. The model takes into account the traffic control strategies between and within the sub-areas of the road network, with important intersections in the two sub-areas as the game subject at the upper layer and secondary intersections as the game subject at the lower layer.

Next, the proposed model is combined with reinforcement learning and deep reinforcement learning, resulting in the development of two multi-agent algorithms: NSHG-QL and NSHG-DQN. These algorithms are used to achieve coordinated control of traffic signals in urban areas. Experimental results show that, compared to basic game model solving algorithms, NSHG-QL and NSHG-DQN algorithms can significantly improve the average travel time, time loss, average speed, and road occupancy of vehicles at intersections.

In summary, this research has made significant innovations in the field of traffic signal control, offering new insights and methods for addressing urban congestion, traffic accidents, and improving traffic efficiency.

II. MULTI-AGENT REINFORCEMENT LEARNING ALGORITHMS FOR MIXED-TYPE TASKS IN TRAFFIC SIGNAL CONTROL

The concept of a multi-agent system has evolved from distributed artificial intelligence. It involves multiple agents working in an environment with characteristics like autonomy, distribution, coordination, etc. Its research objective is to solve complex and large-scale real-world problems [19]. In such scenarios, the decision-making ability of a single agent is inadequate. On the other hand, a centralized agent faces challenges due to various resource and condition constraints. It is also difficult to pre-design behaviours for agents in a multi-agent environment because of its complexity [20], [21]. Thus, agents in a multi-agent system need to learn new behaviours online over time, gradually improving their individual performance or the performance of the entire system [22], [23]. RL is a widely used framework for dealing with interaction and learning between agents and the environment. MARL, which combines RL methods and multi-agent systems, is gradually becoming a research hotspot in the RL field and is being widely used in various fields [24].

By utilizing RL algorithms in a multi-agent system, MARL enables each agent to learn its own strategy and work together towards achieving the system's goal. It tackles a sequential decision-making problem where multiple agents interact with each other in a shared system, receive reward signals, and improve their policies to maximize cumulative rewards. In recent years, MARL has gained significant attention for its remarkable success in various complex tasks that involve multiple participants, such as real-time strategy games [25], [26], card games [27], sports games [28], autonomous driving [29], and robotic control [30].

MARL applies the Markov decision process proposed by Littman as an environment framework [31] and is formalized by the tuple $\langle s, a^1, \cdots, a^n, p, r^1, \cdots, r^n \rangle$, where n denotes the number of Agents, s is the set of states, and a^i is the set of actions executed by each agent. The transition probability function can be denoted by $p{:}s{\times}a^1 \times \cdots \times a^n {\rightarrow} \Delta(s)$. $r^i{:}s{\times}a^1 \times \cdots \times a^n {\rightarrow} R$ denotes the reward function. $\Delta(s)$ is the set of probability distributions on the set s. Currently, MARL suffers from five core problems, including non-stationarity, partial observability, coordination, credit assignment allocation, and scalability issues [32].

MARL can be divided into fully cooperative, fully competitive and mixed according to the type of agents' tasks in the multi-agent system [33].

In the fully cooperative MARL algorithm, all agents receive the same reward $r = r^i = \cdots = r^N$, so each agent is incentivized to cooperate and try to avoid its failure to achieve overall optimality. Another MARL algorithm realizes the full the cooperation between agents by considering the average reward of the system as a whole [34], [35], [36]. This kind of algorithm allows each agent to have different reward functions that are kept secret from each other, and the goal of cooperation is to optimize any average reward $r\left(s, a, s'\right) = \frac{1}{N}\sum_{i \in N} r^i\left(s, a, s'\right)$. The fully competitive MARL algorithm can be regarded as a zero-sum Markov game between agents, in which the sum of rewards for any state transition is zero, that is, $r = \sum_{i=1}^{N} r^i\left(s, a, s'\right)$. In this type of algorithm, the agent strives to maximize its own reward while minimizing the rewards of other agents. In a loose sense, the agents play a competitive game and attempt to defeat their opponents while ensuring that the total system reward is non-zero.

In a mixed setting MARL algorithms, the agents play a mixed stochastic game, which is neither fully cooperative nor fully competitive. As a result, the agents' goals and reward functions are unconstrained. This kind of algorithm is most suitable for selfish agents, and the concept of equilibrium in game theory is most commonly used in this type of game [37].

The mixed MARL algorithm can be divided into static and dynamic game algorithms, with the dynamic game algorithm further divided into equilibrium-related and equilibrium-independent algorithms. The equilibrium-related algorithms include Nash Q-Learning (NQL) [38], Correlated Q-Learning (CQL) [39], Asymmetric Q-Learning (AQL) [40], Friend-or-Foe Q-Learning [41], and Negotiation-based Q-Learning [42]. NQL converges to Nash equilibrium using the reinforcement learning method after several learning iterations. CQL and AQL solve the equilibrium problem among agents by utilizing the correlation and Stackelberg equilibrium in game theory, respectively.

Single-agent reinforcement learning algorithms like Q-learning can be directly applied to fixed tasks [43]. However, this approach violates the basic assumption of reinforcement learning, which is that the environment should be stationary, and the state transition should be a Markov process. In contrast, in a multi-agent system environment, the environment of any single agent is dynamic and non-stationary due to the constantly changing policies of other agents [44].

A traffic network is a multi-agent system where each agent controls a signal light at an intersection [13]. Reinforcement learning is advantageous for signal timing control since it doesn't require modelling the traffic environment and can choose the optimal strategy in the interaction between individuals and the environment. MARL control is the extension of RL from a single intersection to a regional traffic network, aiming to approximate the optimal equilibrium strategy with the coordination of RL agents at multiple intersections [45]. The behaviour of the signal light agent in this environment affects other agents because signal control at any intersection may transfer delay to the upstream, downstream, and other intersections [11]. Therefore, mixed MARL related to equilibrium is suitable for road network signal control problems. Many researchers have combined MARL and game theory to obtain effective and reasonable traffic signal control strategies [13]. For example, Kodama et al. studied the coordination problem between two intersection signals using stochastic game theory and RL [46], and Abolghasem adopted the methods of fuzzy Q-learning and game theory, where agents make decisions based on previous experience and strategies of neighbouring agents [47]. Pan et al. proposed a MARL framework considering game theory based on mixed strategy Nash equilibrium [4], and Zhang et al. set the global objective function of the network as a linear sum of multiple regional objective functions to find the global Nash equilibrium strategy [16]. While Nash equilibrium is useful for describing the agent's optimal strategy in the road network, reflecting on the different importance and game relationships between intersections is necessary. Moreover, as the number of intersections and traffic volume increase, the complexity of traffic signal timing control rises, making Nash equilibrium difficult to calculate due to computational complexity.

III. NETWORK-WIDE TRAFFIC SIGNAL CONTROL BASED ON THE NASH-STACKELBERG HIERARCHICAL GAME MODEL

A. THE NASH-STACKELBERG HIERARCHICAL GAME MODEL

In this paper, we construct a Nash-Stackelberg hierarchical game model that takes into account the traffic control strategies both between and within the sub-areas of the road network. The game subject at the upper layer consists of the important intersections in the two sub-areas, while the game subject at the lower layer comprises the secondary intersections. The aim is to achieve network-wide traffic signal coordinated control. Fig. 1 shows the Nash-Stackelberg hierarchical game model.

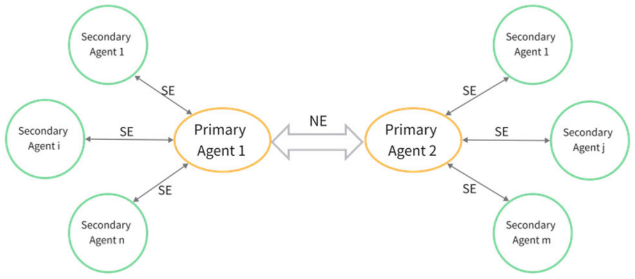


FIGURE 1. Nash-stackelberg hierarchical game model.

The goal of the upper-layer game is to find a coordinated control strategy among all the P-Agents. The strategy choice of each P-Agent is the most appropriate response to the strategy of the other P-Agent, regardless of S-Agents, and there is no need to deviate from this equilibrium point. The P-Agent's decision-making process can be described by stochastic game theory, which is referred to as Markov games, and can be represented by matrix games. The Nash equilibrium solution can be calculated using the Lemke-Howson algorithm. In the upper-layer game, a game equilibrium point is a tuple $\left(\pi^{1,*}, \pi^{2,*}, \cdots, \pi^{n}\right)$ such that for all $s \in S$, P-Agent 1and P-Agent 2, as shown in the Eq. 1.

$$v^1\left(s, \pi^1, \pi^{2,*}, \cdots, \pi^n\right) \leq v^1\left(s, \pi^{1,*}, \pi^{2,*}, \cdots, \pi^n\right) \tag{1}$$

v^1 represents the value for P-Agent v^1 when adopting the strategy combination $\left(\pi^1, \pi^2, \cdots, \pi^n\right)$ in state s, and π^i represents the strategy adopted by Agent i.

The S-Agent, which is the subject of the lower-layer game, includes the signals of secondary intersections that do not affect each other within the same sub-area. The S-Agent formulates its strategy based on the control strategy of the important intersection signal, which is formulated by the P-Agent, and it also affects the game target of the P-Agent. We denote the policy space of both P-Agents as Π^1, Π^2. Upper-layer game equilibrium solution is denoted by $\pi^{1,*}, \pi^{2,*}$. And the policy space of S-Agent j corresponding to P-Agent 1 is Π^j. If the set of S-Agent j's response policies $R_1^j\left(\pi^1\right) \in \Pi^j$, $\forall \pi^1 \in \Pi^1$, there is a mapping $T_1^j: \Pi^1 \rightarrow \Pi^j$ so that if $\pi^j \in R_1^j\left(\pi^1\right)$ then S-Agent j's response to P-Agent 1 and P-Agent 2 as the Eq. 2.

$$\pi^j = T_1^j\left(\pi^1\right) \tag{2}$$

A lower-layer game is designed to find the best response of the S-Agent to the strategy adopted by the P-Agent after the upper-layer game without knowing the strategies of the other S-Agents. At the lower-layer game, for the strategy $\pi^{1,*}$ adopted by P-Agent 1, the lower-layer game solution $\pi^j \in \Pi^j$ of S-Agent j satisfies Eq. 3 for all $s \in S$.

$$v^j_{\pi^{1,*},\pi^j}(s) \leq v^j_{\pi^{1,*},T_1^j\left(\pi^{1,*}\right)}(s) \tag{3}$$

In the Nash-Stackelberg hierarchical game model, the first step is for the P-Agent to decide on its own strategy, and then the S-Agent's policy is determined by the action taken by the P-Agent.

$$\pi^j : S \times A^1 \rightarrow A^j \tag{4}$$

In this hierarchical game model, the states are constrained by practical conditions such as traffic system delays, preventing the acquisition of complete information in a short period. As a result, each agent considers only its own local state. Additionally, the model adheres to the constraint that agents in the lower-level game should not mutually influence each other. However, the model faces limitations due to the computational complexity associated with Nash equilibrium, preventing the expansion of the number of upper-level agents beyond three. Consequently, extending the model to more complex systems becomes challenging.

B. MARL USING NASH-STACKELBERG HIERARCHICAL GAME MODEL

A multi-agent reinforcement learning algorithm based on the Nash-Stackelberg hierarchical game model (NSHG-QL) is proposed in this paper. NSHG-QL is an algorithm that uses the Q-learning algorithm in RL as the core controller and combines it with game theory. For a multi-agent environment, the Q-function for any individual agent becomes $Q^{i,*}\left(s^i, a^1, a^2, \cdots, a^n\right)$. Given the concept of a hierarchical game solution between agents, we define a game equilibrium value Val as the expected sum of discounted rewards. This value is obtained when all agents follow specified hierarchical game equilibrium strategies from the next learning period. The definition is different from the signal-Agent case, where future rewards are based only on the agent's optimal strategy. The maximum operator can be replaced by the equilibrium value of the hierarchical game calculated according to the hierarchical game model. The updated Q-functions are different for the P-Agent and S-Agent due to the hierarchical game.

In NSHG-QL, let Agent i be a P-Agent so that Agent i's Q-function is defined over $\left(s^i, a^1, \cdots, a^n\right)$. That is, Agent i's current and future rewards when all P-Agents and S-Agents follow hierarchical game equilibrium strategies, as shown in Eq. 5.

$$\begin{aligned} Q^{i,*}\left(s^i, a^1, \cdots, a^n\right) &= r^i\left(s^i, a^1, \cdots, a^n\right) \\ &\quad + \beta \sum_{s' \in S} p\left(s' \mid s^i, a^1, \cdots, a^n\right) \\ &\quad \times v^i\left(s', \pi^{1,*}, \pi^{2,*}, \cdots, \pi^n\right) \end{aligned} \tag{5}$$

where $i = 1, 2$, and $\left(\pi^{1,*}, \pi^{2,*}, \cdots, \pi^n\right)$ is the hierarchical game equilibrium strategies, $r^i\left(s^i, a^1, \cdots, a^n\right)$ is Agent i's reward in local state s^i. $v^i\left(s', \pi^{1,*}, \pi^{2,*}, \cdots, \pi^n\right)$ is Agent i's total discounted reward over infinite periods starting from state s^i given that P-Agent i follows the hierarchical game equilibrium strategies.

NSHG-QL is similar to standard single-agent Q-learning in many ways but differs in one crucial element: how to use the Q-values of the next state to update those of the current state. NSHG-QL updates with future hierarchical game equilibrium payoffs, whereas single-agent Q-learning updates are based on the agent's maximum payoff. At each time step t, Agent i observes the current state s^i_t, selects and executes action according to the action selection mechanism, and then calculates its reward r^i_t and the equilibrium solution$\left(\pi^{1,*}, \pi^{2,*}, \cdots, \pi^n\right)$ of the upper-layer game. Finally, the Q-value is calculated, and the Q table and upper-layer game matrix are updated. Q-value is determined according to Eq. 6.

$$\begin{aligned} Q^i_{t+1}\left(s^i_t, a^1, \cdots, a^n\right) &= (1-\alpha)\, Q^i_t\left(s^i_t, a^1, \cdots, a^n\right) \\ &\quad + \alpha\left[r^i_t + \gamma Val^i_t\right] \end{aligned} \tag{6}$$

where $\gamma \in 0, 1$ represents the discount factor, and $\alpha \in 0, 1$ represents the learning rate. γ is utilized to mitigate the impact of future rewards, with a larger discount factor placing greater emphasis on long-term rewards, while a smaller one prioritizes short-term rewards. On the other hand, α controls the update step size of the Q-values. A smaller learning rate results in a smaller update step size, reinforcing dependence on historical experiences, contributing to a more stable learning process, albeit at the cost of slower learning. Conversely, a larger learning rate allows for faster learning but may introduce instability into the learning process. Where hierarchical game equilibrium value Val^i_t is equilibrium value of upper-layer game.

$$Val^i_t = NashQ^i_t\left(s^i_{t+1}\right) \tag{7}$$

Calculating the upper-layer game equilibrium value of the P-Agent is based on the game process between two P-Agents using the Lemke-Howson algorithm, as shown in Eq. 8.

$$NashQ^i_t\left(s^i_{t+1}\right) = \pi^{1,*}\left(s^1_{t+1}\right) \cdot \pi^{2,*}\left(s^2_{t+1}\right) \cdot Q^i_t\left(s^i_{t+1}\right) \tag{8}$$

In NSHG-QL, let Agent j be the S-Agent corresponding to P-Agent 1. Agent j's Q-function is defined as Eq. 9.

$$\begin{aligned} &\mathrm{Q}^{j,*}\left(\mathrm{s}^j, \mathrm{a}^1, \cdots, \mathrm{a}^n\right) \\ &\quad = \mathrm{r}^j\left(\mathrm{s}^j, \mathrm{a}^1, \cdots, \mathrm{a}^n\right) \\ &\quad\quad + \beta \sum_{s' \in S} \mathrm{p}\left(\mathrm{s}' \mid \mathrm{s}^j, \mathrm{a}^1, \cdots, \mathrm{a}^n\right) \\ &\quad\quad \times \mathrm{v}^i\left(\mathrm{s}', \pi^1, \pi^2, \cdots, \mathrm{T}^j_1\left(\pi^1\right), \cdots, \pi^n\right) \end{aligned} \tag{9}$$

The basic idea is that at each time step t, S-Agent j observes the current state s^j, selects and executes action based on the action selection mechanism. Then, it calculates its reward r^j_t and the equilibrium solution $\left(\pi^1, \pi^2, \cdots, R^j_1\left(\pi^1\right), \cdots, \pi^n\right)$ of the lower-layer game. Finally, the Q-value is calculated, and the Q table and lower-layer game matrix are updated based on the Q-value. The Q-value is determined according to Eq. 10.

$$\begin{aligned} \mathrm{Q}^j_{t+1}\left(\mathrm{s}^j_t, \mathrm{a}^1, \cdots, \mathrm{a}^n\right) &= (1-\alpha)\, \mathrm{Q}^j_t\left(\mathrm{s}^j_t, \mathrm{a}^1, \cdots, \mathrm{a}^n\right) \\ &\quad + \alpha\left[\mathrm{r}^j_t + \gamma\, \mathrm{Val}^j_t\right] \end{aligned} \tag{10}$$

where the maximum operator can be replaced with the equilibrium value Val^j_t of hierarchical game.

$$Val^j_t = SeQ^j_t\left(s^j_{t+1}\right) \tag{11}$$

Calculating the lower-layer game equilibrium value of S-Agent j is based on the game process between it and the corresponding P-Agent 1. P-Agent 1 chooses its strategy $\pi^{1,*}$ based on the upper-layer game, which forces S-Agent j to select the optimal strategy that can lead to the maximum payoff. The Stackelberg solution $\left(\pi^{1,*}, T^j_1\left(\pi^{1,*}\right)\right)$ can be obtained as shown in Eq. 12.

$$SeQ^j_t\left(s^j_{t+1}\right) = \pi^{j,*}\left(s^j_{t+1}\right) \times T^j_1\left(s^j_{t+1}, \pi^{1,*}\right) \times Q^j_t\left(s_{t+1}\right) \tag{12}$$

where $\mathrm{T}^j_1\left(\mathrm{s}^j_{t+1}, \pi^{1,*}\right)$ is the optimal reactive strategy that S-Agent makes in response to the upper-layer game equilibrium strategy $\pi^{1,*}$ of P-Agent 1 at time t+1.

The learning process and formula are similar to those described above for the other P-Agent in the multi-agent system, named P-Agent 2, and the corresponding S-Agent under its leadership. In summary, the detailed steps of MARL for both P-Agent and S-Agent are shown in Algorithm 1.

Algorithm 1 NSHG-QL

```
Input Iteration time $E$, Simulation step T,
        Exploration rate $\epsilon$, Learning rate $\gamma$.
Initialize Q table
Initialize upper-layer game matrix $NT_t^i$
Initialize lower-layer game matrix $AT^j$
1. For $e = 1 \ldots E$ do
2.   For $t = 1 \ldots T$ do
3.     Observe current state $s_t^i$
4.     With probability 1-$\epsilon$ select $a_t^i = \max_a Q^*\left(s_t^i, a\right)$
5.     Calculate reward $r_t^i$
6. If Agent $i$ is a P-Agent then
7.   Calculate $Val_t^i$ using $NT_t^i$ (Eq.8)
8.   Calculate the Q value (Eq.6)
9.   Update $NT_t^i$ by $Val_t^i$
10. End
11. If Agent $j$ is an S-Agent then
12.   Calculate $Val_t^j$ using $AT^j$ (Eq.12)
13.   Calculate the Q value (Eq.10)
14.   Update $AT^j$ by $Val_t^j$
15. End
16.     Update Q table with Q values
17.   End
18. End
```

C. MADRL USING NASH-STACKELBERG HIERARCHICAL GAME MODEL

In the actual traffic signal control environment, the control task exhibits the characteristics of a high-dimensional state space and a continuous action space. Traditional reinforcement learning methods cannot compute value and policy functions for all states. As the number of intersections increases, the state of the intersection also increases, leading to an exponential increase in the agent's global/joint action space. To address this problem, deep reinforcement learning is considered in this paper as a method. By applying deep learning to supervised learning, the function fitter can obtain accurate function approximation. Additionally, the neural networks can be trained to learn optimal policies and value functions. Inspired by the Deep Q Network (DQN), this paper proposes a MADRL algorithm: Hierarchical Nash-Stackelberg Game DQN (NSHG-DQN), which is based on the deep reinforcement learning algorithm and the Nash-Stackelberg hierarchical game model.

In NSHG-DQN, a deep neural network is used as a function approximator to map states to Q values instead of estimating Q-values for each state-action pair individually. To increase convergence stability during the learning process, experience replay and target networks are also employed.

During the experience replay of P-Agent i, recent experiences are stored in the memory buffer as $\left(s_t^i, a_t^i, r_t^i, NT_t^i, AT^i\right)$ and regularly randomly selected in small batches for neural network training. In the NSHG-DQN training process, the parameter θ of the main network is updated after executing each action, and the parameter θ^i of P-Agent i's target

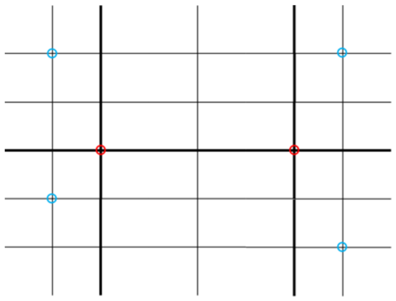


FIGURE 2. The traffic grid in SUMO.

network is updated after a period of time using the TD target y_t^i step, as shown in Eq. 13.

$$y_t^i = r_t^i + \gamma Val^i\left(s_t^i, a_t^i; \theta_t^{i,-}\right) \tag{13}$$

where $Val^i\left(s_t^i, a_t^i;\theta_t^{i,-}\right)$ is target network, using the Nash game matrix $NT \wedge i$ is calculated according to Eq. 6.

The deep neural network is essentially used to fit a function by using the output of the main network and the given label to calculate the loss function. Then, the variables of the neural network are adjusted through backpropagation. The goal is to make the neural network output closest to the given labels. In NASH-DQN, the neural network is a function of the state and the Q value, where the input of the neural network is the state observed by the Agent and the output is the Q value of the corresponding state. The label is calculated according to the game matrix of the upper or lower layer of the hierarchical game model. P-Agent i's loss function is defined in Eq. 14.

$$L\left(\theta_i\right) = E_{(s,a,r)\sim U(D)}\left[\left(r_{t+1}^i + \gamma Val^i\left(s_t^i, a_t^i; \theta_t^{i,-}\right)\right.\right.$$
$$\left.\left.-Q^i\left(s_t^i, a_t^i; \theta_t^i\right)\right)^2\right] \tag{14}$$

During the experience replay of S-Agent j, the recent experience is stored in memory buffer as $\left(s_t^j, a_t^i, d_t^j, r_t^j, AT^j\right)$, and the parameter θ^j of S-Agent j target network is updated using a TD target y_t^j step, as shown in Eq. 15.

$$y_t^j = r_t^j + \gamma Val^j\left(s_t^j, a_t^j; \theta_t^{j,-}\right) \tag{15}$$

where $Val^j\left(s_t^j, a_t^j;\theta_t^{j,-}\right)$ is target network which is calculated by Eq. 10 using SE game matrix AT^j.

The loss function of S-Agent is defined in Eq. 16.

$$L\left(\theta_j\right) = E_{(s,a,r)\sim U(D)}\left[\left(r + \gamma Val^j\left(s_t^j, d_t^j; \theta_t^{i,-}\right)\right.\right.$$
$$\left.\left.-Q^i\left(s_t^j, a_t^j; \theta_t^j\right)\right)^2\right] \tag{16}$$

In summary, the detailed steps of MARL for the P-Agent and S-Agent are shown in Algorithm 2.

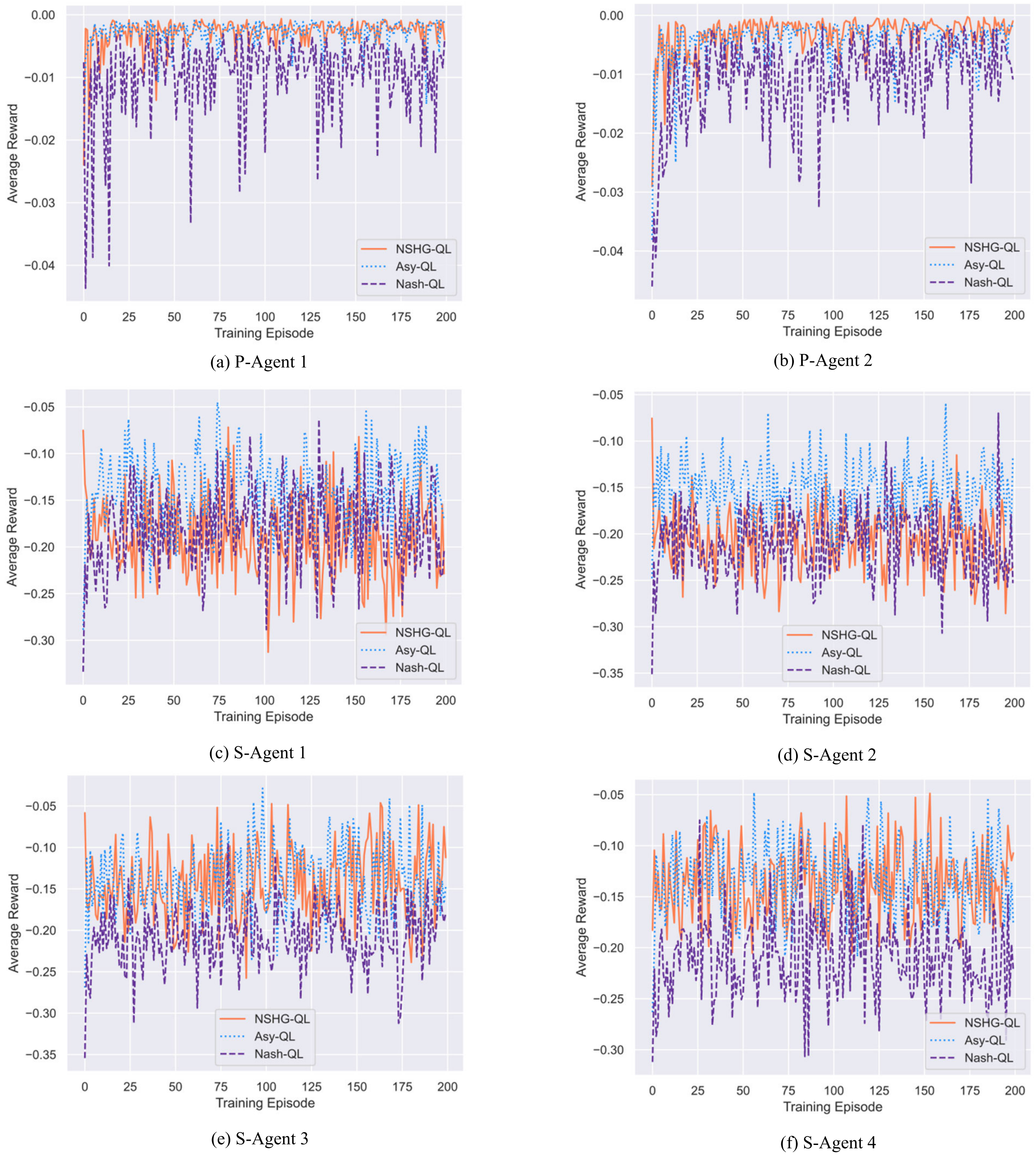


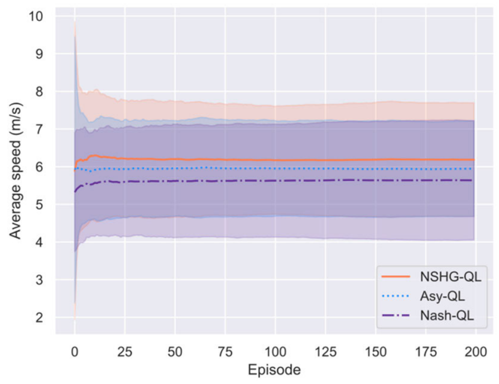
FIGURE 3. **The average reward for each intersection agent.**

IV. EXPERIMENTS AND RESULTS

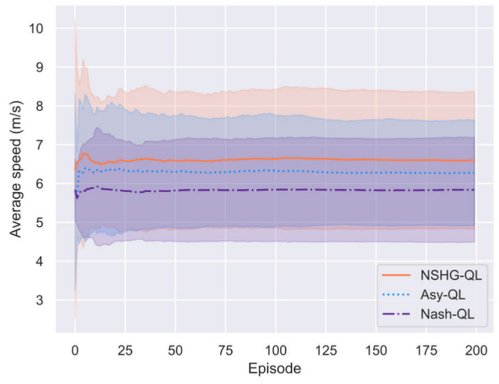
A. EXPERIMENTAL SETTINGS

In the experiment, this paper uses the road network shown in Fig. 2 to evaluate the proposed algorithm and employs the SUMO simulation software for simulation. The longer edge in the simulation has a length of 400m, and the shorter edge has a length of 200m. The critical intersection is represented as a two-way 6-lane road, with a turning probability of 0.2 for left-turn, 0.6 for right-turn, and 0.2 for straight-going in the north-south edge, while the turning probability in the east-west edge 0.3 for left-turn, 0.4 for right-turn, and 0.3 for straight-going. The initial phase of the intersection signal light is two-phase, and the period is 46s. The north-south and the east-west phases have an initial duration of 20s, and the yellow phase for each phase has a duration of 3s.

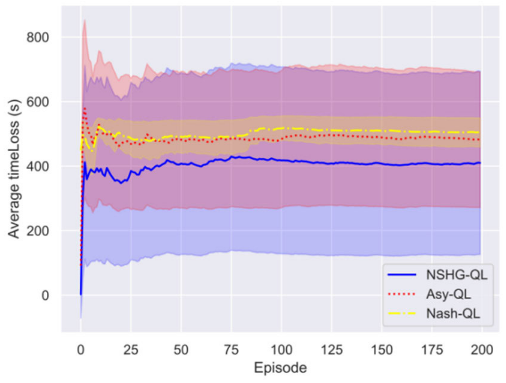
To verify the performance of the NSHG-QL and NSHG-DQN, we compare the two algorithms with Nash-Q learning (Nash-QL), Asy-Q learning algorithm (Asy-QL),



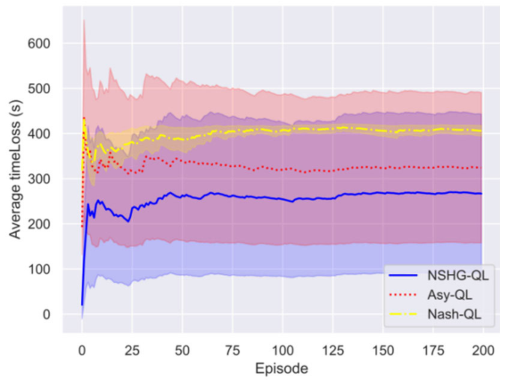
(a) P-Agent 1



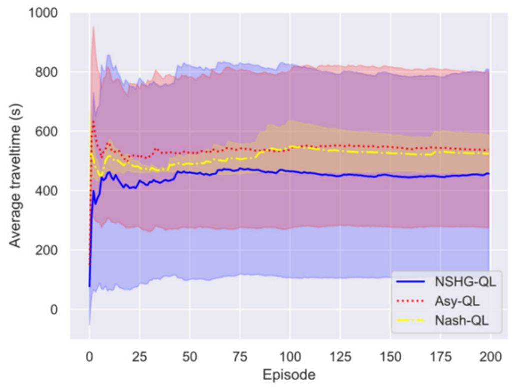
(b) P-Agent 2



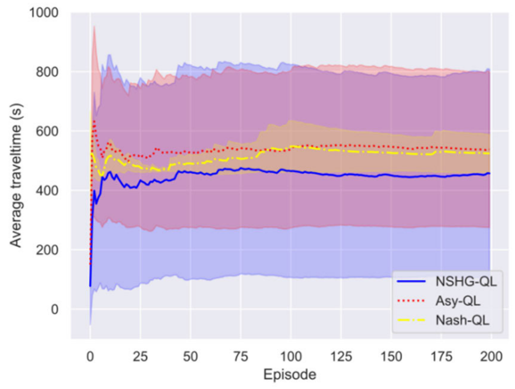
(c) S-Agent 1



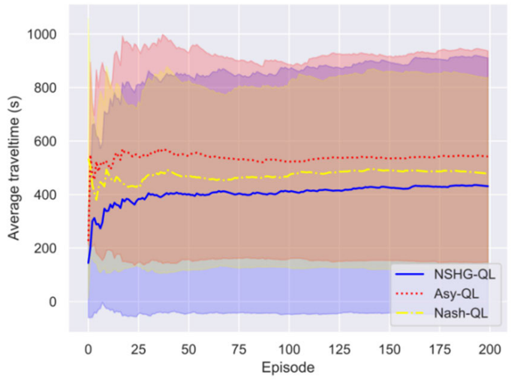
(d) S-Agent 2



(e) S-Agent 3

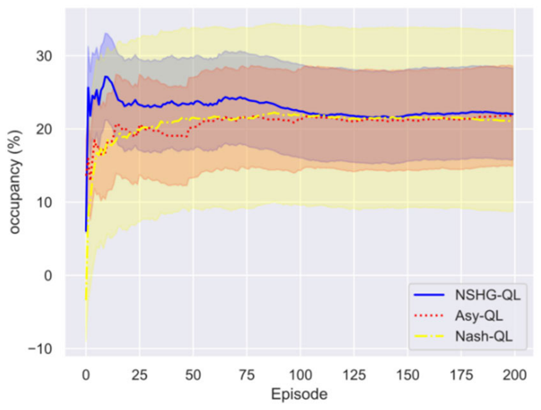


(e) S-Agent 3

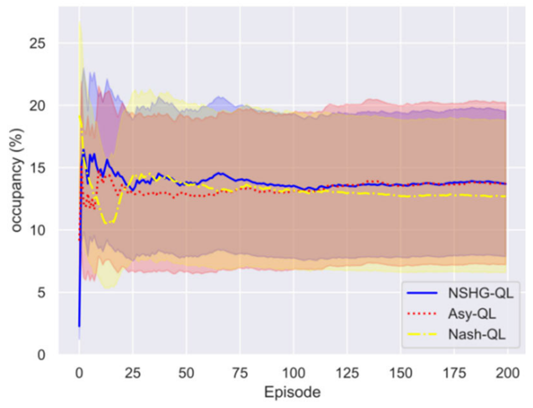


(f) S-Agent 4

FIGURE 4. The average speed (a, b), the average time loss (c, d), and the average travel time (e, f) data.



(a) P-Agent 1



(b) P-Agent 2

FIGURE 5. The occupancy data.

Algorithm 2 NSHG-DQN

```
Input Iteration time $E$, Simulation step T, Experience
    replay $D$, Exploration rate $\epsilon$, Learning rate $\gamma$.
Initialize Q network weight $\theta^i$, target network weight $\theta^{i,-}$
Initialize upper-layer game matrix $NT_t^i$,
Initialize lower-layer game matrix $AT^j$
1. For $e = 1 \ldots E$ do
2.   Observe state sequence $s_1^i = \{x_1^i\}$ and $\phi_1^i = \phi\left(s_1^i\right)$
3.    For $t = 1 \ldots T$ do
4.   Select $a_t^i = \max_a Q^*\left(\phi\left(s_t^i\right), a; \theta^i\right)$ with 1-$\epsilon$
5.   Calculate reward $r_t^i$
6.   Store $\left(s_t^i, a_t^i, r_t^i, \phi_t^i, AT^i\right)$ in $D$
7.   Sample random minibatch $m$ from $D$
8. If Agent $i$ is a P-Agent then
9.   Calculate $Val^i\left(s_t^i, a_t^i; \theta_t^{i,-}\right)$ (Eq.7)
10. End
11. If Agent $j$ is an S-Agent then
12.   Calculate $Val^j\left(s_t^j, a_t^j; \theta_t^{j,-}\right)$ (Eq.11)
13. End
14. $y_t^i = \begin{cases} r_t^i, & t = T \\ r_t^i + \gamma Val^i\left(s_t^i, a_t^i; \theta_t^{i,-}\right), & \text{otherwise} \end{cases}$
15. Update $NT_t^i$ or $AT^j$ using $y_t^i$
16. Update $\theta^i$ using $\left(y_t^i - Q^{i,*}\left(\phi_t, a_t; \theta^i\right)\right)^2$
17. Update $\theta^{i,-} = \theta^i$ every $N$ steps
18. End
```

Nash-DQN, and Asy-DQN algorithms. Nash-QL and Nash-DQN coordinate the control strategies of important intersection signals in the two areas using the Nash equilibrium. Asy-QL and Asy-DQN algorithm coordinate the control strategies of important intersection lights and other minor intersection lights in the two regions by employing Stackelberg equilibrium. Four evaluation indicators are used in the experiment: mean speed, which is the average speed of vehicles passing the area; mean time loss, which is the average time loss for all vehicles passing the area; mean travel time, which is the time required for vehicles to pass the area; and occupancy, which is the percentage (0-100%) of the time a vehicle was detected by the detector.

1) STATES

Vehicle information at intersections is a crucial factor in traffic signal control, particularly the number of queued vehicles at the intersection. In a multi-agent system, the joint state space exponentially increases, and each agent cannot entirely access the entire state space. A joint state space may also present the issue of challenging information transmission between intersections. Thus, it is necessary to simplify the state space of each intersection [48]. To better describe the traffic state of the intersection, in this article, Agent i is only able to partially observe the system feature information in its own intersection area at time t. To limit the status space and facilitate the system, the queue length of the four entrance edges at each intersection is selected to form a state together, where s^i is the state of intersection i, l_{que}^k is the length of the queue for the edge k of the interchange i, as shown in Eq. 17.

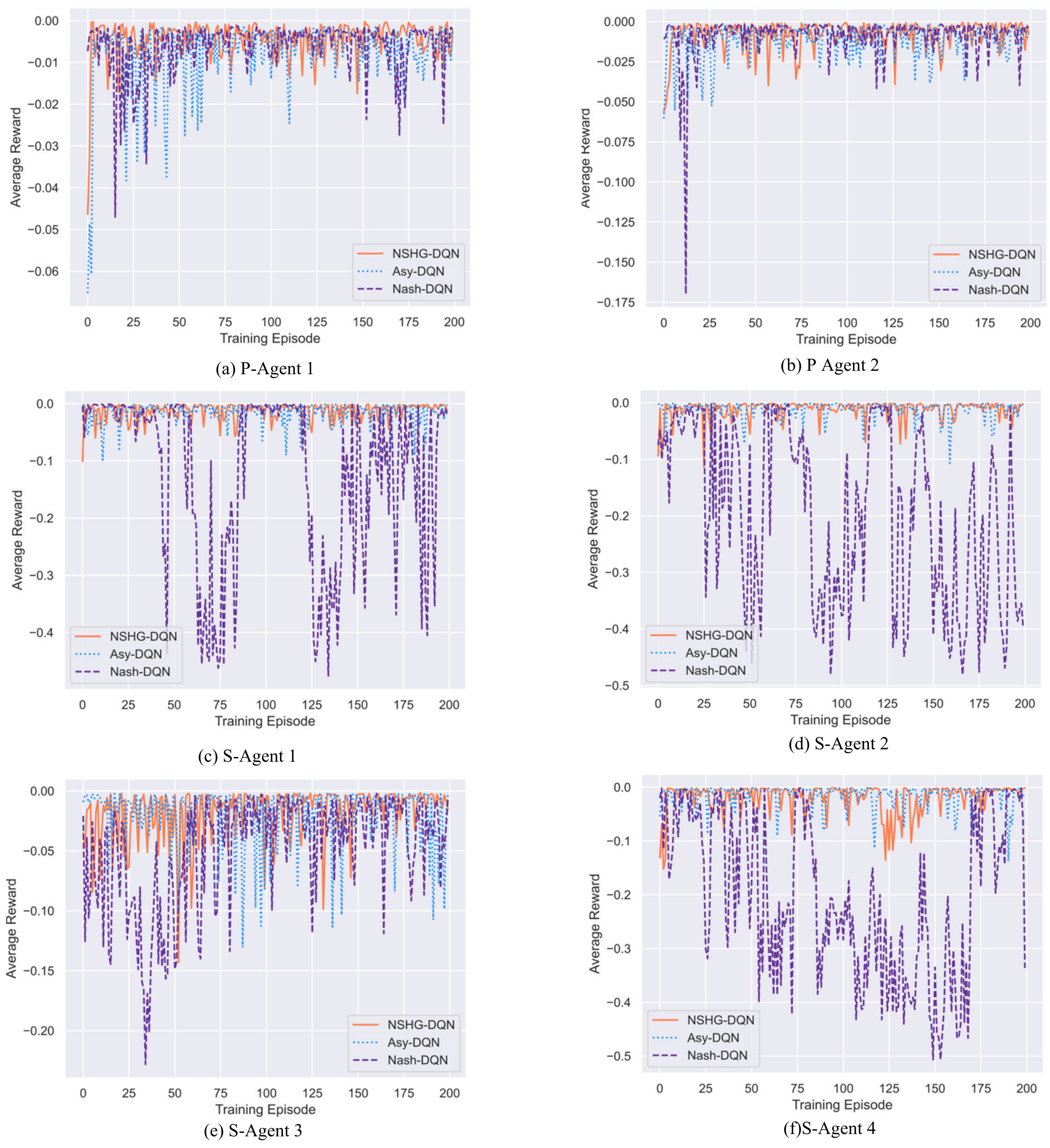
$$s^i = [l_{que}^1, l_{que}^2, l_{que}^3, l_{que}^4] \tag{17}$$

2) ACTIONS

The action space refers to the set of possible actions ($a^i \in A^i$) that the agent i can choose after observing the intersection state at time step t. The set of all actions that the agent can take is denoted by A^i, and the agent executes the selected action. In this paper, we consider the possibility of switching phases for different durations. The duration of each action is the length p of the signal phase configuration. At time step $t + p$, the agent observes the updated state affected by the latest action and selects the next action. It is possible for the agent to take the same action at time step $t + p$ and t.

In this article, actions are selected using the ϵ-greedy mechanism. At time t, agents take $\epsilon \in [0, 1]$ as the exploration probability. When the probability is 1-ϵ, agents perform the optimal action according to the highest Q value. Otherwise, agents select an action randomly.

$$a_t^i = \begin{cases} \underset{a}{\text{argmax}}\, Q\left(s_t, a\right) 1-\epsilon \\ Random\ Action\ \epsilon \end{cases} \tag{18}$$



(a) P-Agent 1

(b) P Agent 2

(c) S-Agent 1

(d) S-Agent 2

(e) S-Agent 3

(f)S-Agent 4

FIGURE 6. The average reward data.

3) REWARD

The reward function is to evaluate how well the agent's actions affect the environment. The environment provides feedback on the effect of the executed action on itself, which is calculated according to the reward function and provided to the agent. The agent seeks strategies to maximize rewards. There are many reward mechanisms in traffic signal control, such as the change in queued vehicles, the duration of the green time, and the cumulative delay time of the vehicle. In this article, the reciprocal of the queue waiting time at the intersection is used as the reward function, defined as follows.

$$r_t^i = \frac{1}{w_t^1 + w_t^2 + w_t^3 + w_t^4} \tag{19}$$

where w_t^k is the waiting time for the edge k of intersection i after the action is performed at each time t. It can be seen from the function expression that if the reciprocal of the queue waiting time at the intersection i decreases after the action a_t^i is taken at time t, the reward value r_t^i increases. This means that the action has a positive impact on the current traffic

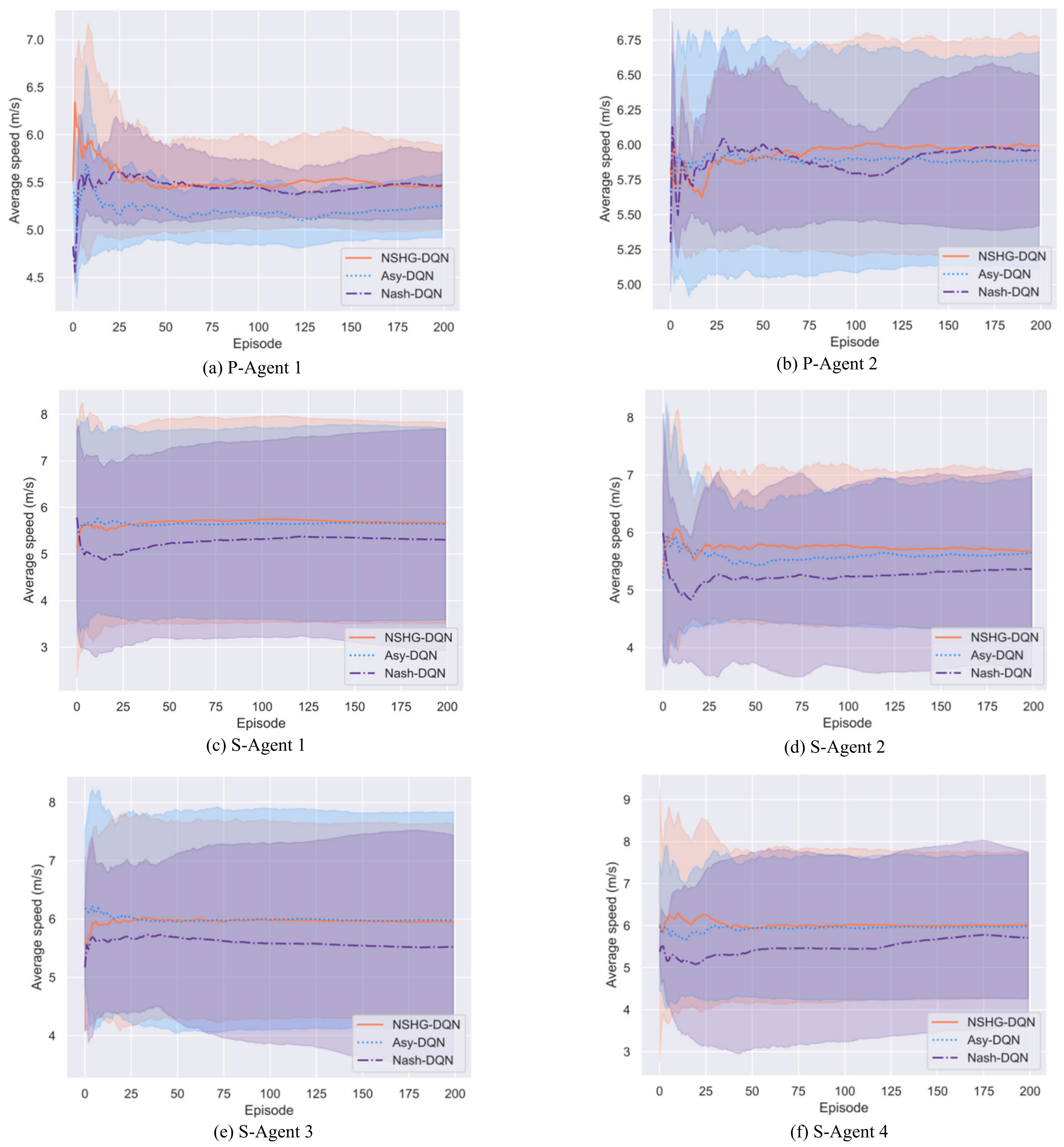


FIGURE 7. **The average speed data.**

state. Therefore, a larger reward value indicates that the action optimizes and positively affects the current intersection traffic state.

B. EXPERIMENTAL RESULTS FOR NSHG-QL

Fig. 3 plots the average rewards per episode for each intersection agent in the road network under the control of three algorithms that use the Q-learning algorithm as the core controller. Taking P-Agent 1 in Fig. 3 (a) as an example, the average reward of the NSHG-QL gradually increases during the learning process. After about 125 episodes of training, it reaches a stable state. However, the results of Asy-QL and Nash-QL increase slowly and fluctuate all the time, failing to converge. The learning trend of P-Agent 2 is consistent with that of P-Agent 1. In Fig. 3 (c), (d), (e), and (f), the average reward trends of S-Agents under the control of the three algorithms are similar, oscillating continuously and not reaching a stable state.

The occurrence of oscillations does not necessarily imply that the algorithm is not converging. Subsequent analysis of the algorithm data proves that the algorithm is indeed

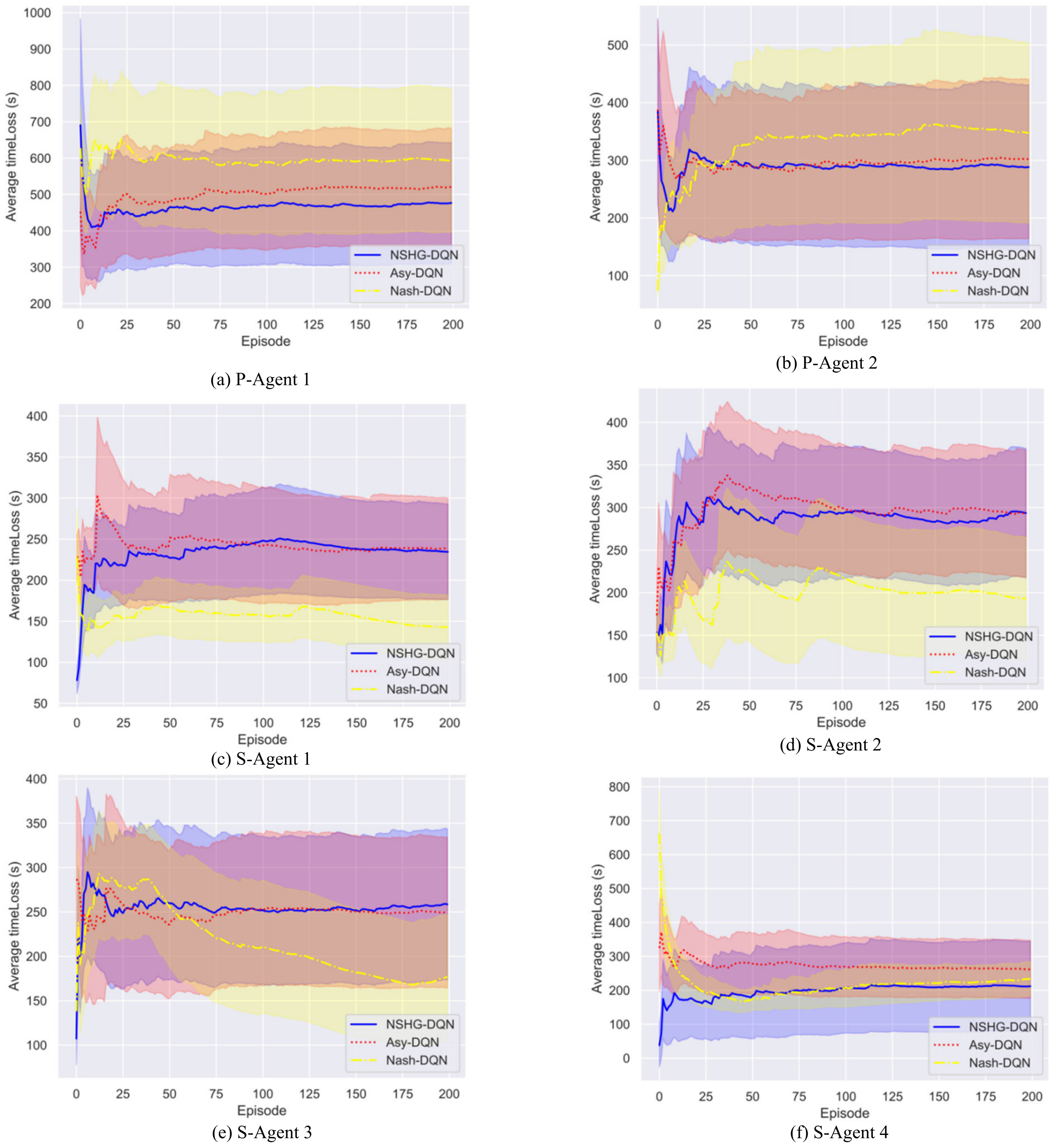


FIGURE 8. The average time loss data.

converging. The reason for the oscillations is that the two algorithms selected for the control experiment in the hierarchical game model are constrained by practical conditions such as traffic system delays, making it impossible to obtain complete information in a short period. Therefore, each agent only considers its own local state and satisfies the constraint that agents should not influence each other in the lower-level game. This can be attributed to the complexity of computing Nash equilibrium, but overall, the algorithm converges.

Fig. 4 shows the traffic data of the areas controlled by the P-Agents, which includes average speed, average time loss and average travel time. The control effect of S-Agents is not included in the evaluation since their policies are meaningless. The two P-Agents controlled by NSHG-QL exhibit superior control effects compared to the comparison algorithms. Taking P-Agent 1 in Fig. 4 (a)(b)(c) as an example, after several training episodes, the average speed of NSHG-QL stabilizes at around 6.3m/s, which is 5% and 12.5% faster than Asy-QL and Nash-QL, respectively. The

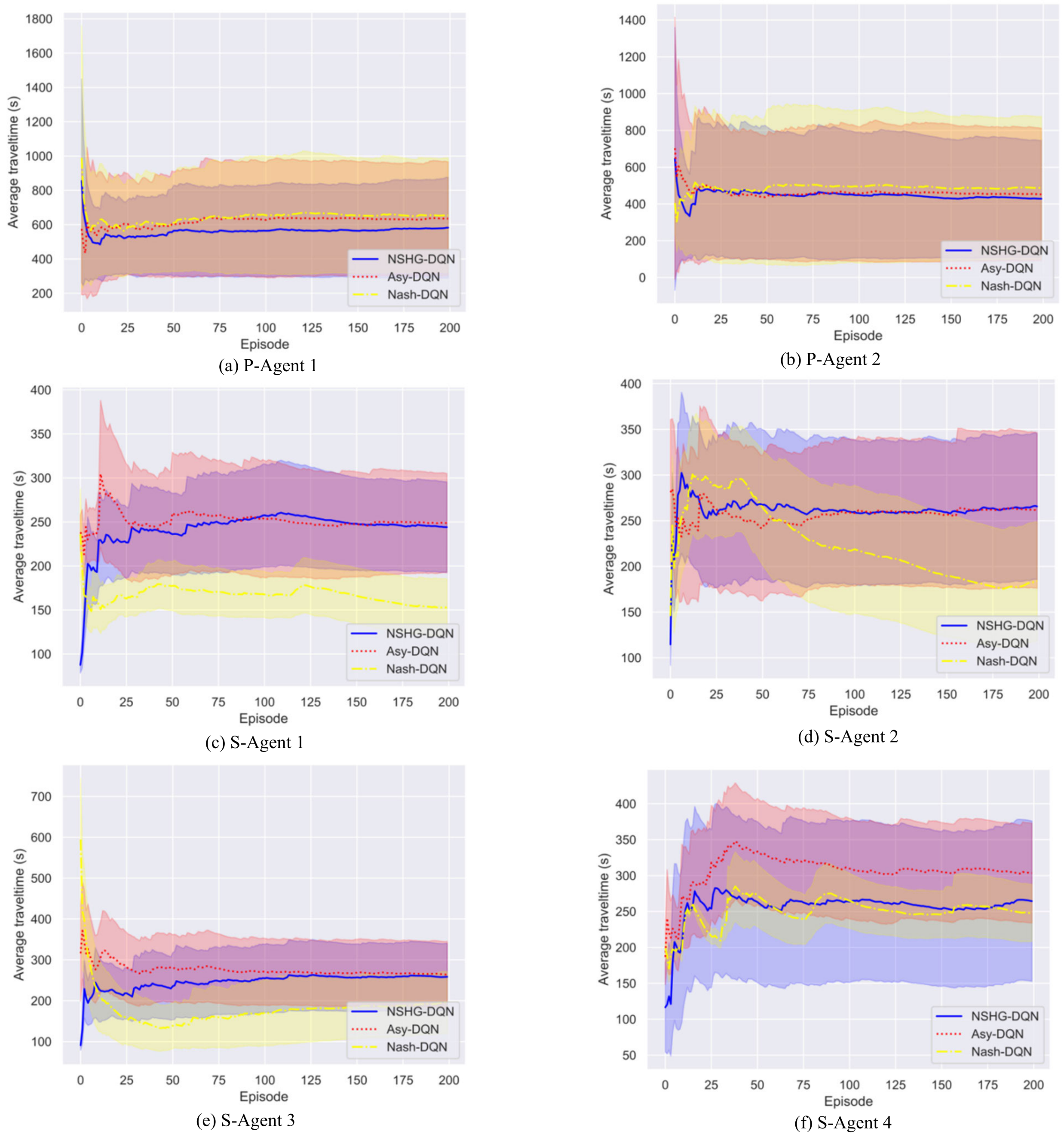
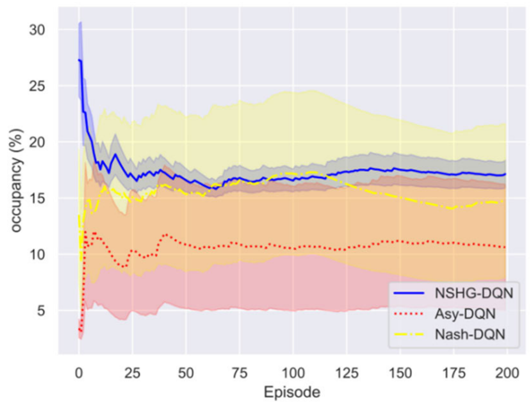


FIGURE 9. **The average travel time data.**

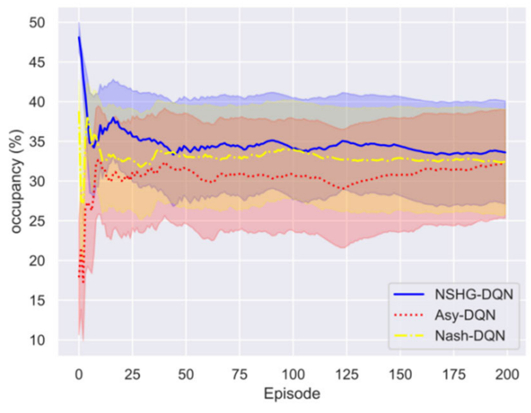
average time loss gradually stabilizes at about 460s, which is about 6% and 10% less than Asy-QL and Nash-QL, respectively. The average travel time loss gradually stabilizes at about 510s, which is about 7% and 3% less than Asy-QL and Nash-QL. The control effect of P-Agent 2 is consistent with the trend of P-Agent 1.

Fig. 5 shows the occupancy of the road segments that connect the two P-Agent intersections. Under the control of NSHG-QL, the occupancy of the east entrance edge at P-Agent 1 and the west entrance edge at P-Agent 2 gradually stabilize at around 22% and 14%, respectively, which are both higher than the compared algorithms.

The experimental results demonstrate that NSHG-QL effectively coordinates the strategy selection between agents through hierarchical games, resulting in better control optimization of P-Agent compared to control algorithms that only considers the single game relationship. As a result of the learning process, P-Agent can converge to the optimal joint strategy and achieve better control.



(a) P-Agent 1



(b) P-Agent 2

FIGURE 10. **The average travel time data.**

C. EXPERIMENTAL RESULTS FOR NSHG-DQN

Fig. 6 plots agents' average rewards under the control of three algorithms that use the DQN algorithm as the core controller. Fig. 6 (a)(b) shows that, after several training episodes, the average reward trends of P-Agents under the control of NSHG-DQN are similar to those of P-Agents under the control of NSHG-QL, which quickly reach a stable state. However, the learning process and stability of NSHG-DQN are stronger than NSHG-QL. The NSHG-DQN can solve the dimensional explosion problem by using deep neural networks for function approximation. Fig. 6 (c)(d)(e)(f) show that, after several episodes of training, the average reward of the S-Agent under the control of Nash-DQN gradually reaches a stable state, while that of the S-Agent under the control of the comparison algorithms is still oscillating and unable to converge. This means that the Nash-Stackelberg hierarchical game model can coordinate the strategy selection between P-Agents and S-Agents to select optimal actions based on satisfying the upper-layer game between P-Agents. Compared with game models that only consider one game relationship in the comparison algorithms, the Nash-Stackelberg hierarchical game model can obtain the global optimal joint strategy.

Fig. 7 shows the average speed data of the P-Agents' and the S-Agents' areas in the road network under the control of three algorithms that use the DQN algorithm as the core controller. As shown in Fig. 7 (a), after several training episodes, the average speed of P-Agent 1 under the control of NSHG-DQN gradually stabilizes at about 5.5m/s, which is 4.7% faster than Asy-DQN and is similar to Nash-DQN. Fig. 7 (d) shows that the average speed of S-Agent 2 under the control of NSHG-DQN gradually stabilizes at about 5.7m/s, which is 1.7% and 7.5% faster than Asy-DQN and Nash-DQN, respectively. The control effect of P-agent 2 is consistent with the trend of P-Agent 1. P-Agent 2 and other S-Agents show similar trends to P-Agent 1 and S-Agent 2.

In addition to the average speed, each agent's average time loss and travel time are shown in Fig. 8 and 9. And Fig. 10 shows the occupancy of the road segments connecting the two P-Agent intersections. In a similar manner to the control effect of average speed, the average time loss and average travel time of P-Agents and S-Agents are reduced after the training becomes stable, and occupancy is improved as a result. Experimental results illustrate that NSHG-DQN significantly improves traffic levels in comparison with MARL which only consider one game relationship.

V. CONCLUSION

This paper proposes a Nash-Stackelberg hierarchical game model for coordinated traffic signal control in a road network. The proposed model takes into account the traffic control strategies both between and within the sub-areas of the road network. In this model, important intersections in two sub-areas act as game subjects at the upper layer, while secondary intersections serve as game subjects at the lower layer. Additionally, this paper proposes the NSHG-QL and NSHG-DQN algorithms, both of which are based on the Nash-Stackelberg hierarchical game model. The algorithms are verified through experiments conducted in the environment constructed by SUMO.

In comparison to the algorithms that consider one game relationship, NSHG-QL enables P-Agent to achieve stable learning and prioritizes the optimization of its control. On the other hand, NSHG-DQN differs from NSHG-QL in that it utilizes deep neural networks to address dimensional explosion, improves the learning rate and stability of P-Agent, and ultimately leads to the convergence S-Agent's learning. As a result, it significantly improves the traffic level at each intersection in the road network. The experimental results demonstrate that the proposed Nash-Stackelberg hierarchical game model can obtain the global optimal joint strategy, reflect the importance of different intersections and game relationships in the road network, and overcome the computational complexity of Nash equilibrium caused by multi-intersection games.

Future work could involve integrating the calculation method of intersection importance with the Nash-Stackelberg hierarchical game model proposed in this study to enhance the performance and adaptability of the algorithms. Additionally, the model's application could be expanded to more intricate systems by considering the game relationship between S-Agents that have a mutual impact on each other.

In larger and more complex traffic networks, the algorithm based on the Nash-Stakelberg hierarchical game model proposed herein demonstrates excellent scalability. Experimental results indicate that the proposed Nash-Stakelberg hierarchical game model can achieve globally optimal joint strategies. This implies that the algorithm can effectively find optimal traffic signal control strategies in larger-scale road networks, enhancing the overall efficiency of the entire traffic network. Simultaneously, it overcomes the computational complexity associated with Nash equilibrium in multi-intersection games, indicating efficient operation of the algorithm in handling complex intersections and game relationships. Future work could explore the integration of methods for calculating intersection importance with the Nash-Stakelberg hierarchical game model to enhance the algorithm's performance and adaptability. Overall, the proposed algorithm exhibits outstanding scalability in larger and more complex traffic networks, providing a robust tool and method for addressing challenges in real-world urban traffic management. Future research directions may further explore how to integrate methods for calculating intersection importance and how to generalize and improve the algorithm in broader systems.

REFERENCES

[1] A. L. C. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," *Auton. Agents Multi-Agent Syst.*, vol. 18, no. 3, pp. 342–375, Sep. 2008.

[2] K.-H. N. Bui and J. J. Jung, "Cooperative game-theoretic approach to traffic flow optimization for multiple intersections," *Comput. Electr. Eng.*, vol. 71, pp. 1012–1024, Oct. 2018.

[3] R. C. González, J. B. Clempner, and A. S. Poznyak, "Solving traffic queues at controlled-signalized intersections in continuous-time Markov games," *Math. Comput. Simul.*, vol. 166, pp. 283–297, Dec. 2019.

[4] Z. Pan, Z. Qu, Y. Chen, H. Li, and X. Wang, "A distributed assignment method for dynamic traffic assignment using heterogeneous-adviser based multi-agent reinforcement learning," *IEEE Access*, vol. 8, pp. 154237–154255, 2020.

[5] H. M. Abdelghaffar and H. A. Rakha, "A novel decentralized game-theoretic adaptive traffic signal controller: Large-scale testing," *Sensors*, vol. 19, no. 10, p. 2282, May 2019.

[6] I. A. Villalobos, A. S. Poznyak, and A. M. Tamayo, "Urban traffic control problem: A game theory approach," *IFAC Proc. Volumes*, vol. 41, no. 2, pp. 7154–7159, Jul. 2008.

[7] X. H. Zhao, Z. L. Li, Q. Yu, and Y. C. Li, "Traffic signal coordination control for two adjacent intersections based on NashCC-Q learning," *Int. J. Simul. Model*, vol. 20, no. 17, p. 4, 2008.

[8] S. Shamshirband, "A distributed approach for coordination between traffic lights based on game theory," *Int. Arab. J. Inf. Techn.*, vol. 2, no. 2, pp. 148–153, 2012.

[9] J. B. Clempner and A. S. Poznyak, "Modeling the multi-traffic signal-control synchronization: A Markov chains game theory approach," *Eng. Appl. Artif. Intell.*, vol. 43, pp. 147–156, Aug. 2015.

[10] Y. Zhao et al., "Traffic signal control for isolated intersection based on coordination game and Pareto efficiency," *Int. Arab. J. Inf. Techn.*, pp. 3508–3513, Oct. 2019.

[11] Y. Zhu, Z. He, and G. Li, "A bi-hierarchical game-theoretic approach for network-wide traffic signal control using trip-based data," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15408–15419, Sep. 2022.

[12] C. Chen, H. Wei, N. Xu, G. Zheng, M. Yang, Y. Xiong, K. Xu, and Z. Li, "Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control," in *Proc. AAAI*, Apr. 2020, vol. 34, no. 4, pp. 3414–3421.

[13] M. Abdoos, "A cooperative multiagent system for traffic signal control using game theory and reinforcement learning," *IEEE Intell. Transp. Syst. Mag.*, vol. 13, no. 4, pp. 6–16, Winter 2021.

[14] Q. Wu, J. Wu, J. Shen, B. Yong, and Q. Zhou, "An edge based multi-agent auto communication method for traffic light control," *Sensors*, vol. 20, no. 15, p. 4291, Jul. 2020.

[15] J. Guo and I. Harmati, "Evaluating semi-cooperative Nash/Stackelberg Q-learning for traffic routes plan in a single intersection," *Control Eng. Pract.*, vol. 102, Sep. 2020, Art. no. 104525.

[16] Z. Zhang, J. Qian, C. Fang, G. Liu, and Q. Su, "Coordinated control of distributed traffic signal based on multiagent cooperative game," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–13, Jun. 2021.

[17] W. U. Mondal, M. Agarwal, V. Aggarwal, and S. V. Ukkusuri, "On the approximation of cooperative heterogeneous multi-agent reinforcement learning (MARL) using mean field control (MFC)," *J. Mach. Learn. Res.*, vol. 23, no. 129, pp. 5614–5659, 2022.

[18] M. Moradi, Y. Weng, and Y.-C. Lai, "Defending smart electrical power grids against cyberattacks with deep Q-learning," *PRX Energy*, vol. 1, no. 3, Nov. 2022, Art. no. 033005.

[19] W. Du and S. F. Ding, "Overview on multi-agent reinforcement learning," *Comput. Sci.*, vol. 46, no. 8, pp. 1–8, Aug. 2019.

[20] Z. Zhao, Y. Gao, B. Luo, and S. Chen, "Reinforcement learning technology in multi-agent system," *Comput. Sci.*, vol. 31, no. 3, pp. 23–27, 2004.

[21] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Applications*, vol. 310. Berlin, Germany: Springer, 2010, pp. 183–221.

[22] G. Weiss, "Distributed problem solving and planning," in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 2000, ch. 3, pp. 121–161.

[23] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Auton. Robots*, vol. 8, no. 3, pp. 345–383, Jun. 2000.

[24] W. Yang, L. Zhang, and F. Zhu, "Multi-agent reinforcement learning based traffic signal control for integrated urban network: Survey of state of art," *Appl. Res. Comput.*, vol. 35, no. 3, pp. 1613–1618, 2018.

[25] K. Arulkumaran, A. Cully, and J. Togelius, "AlphaStar: An evolutionary computation perspective," in *Proc. GECCO*, Jul. 2019, pp. 314–315.

[26] D. Ye et al., "Mastering complex control in MOBA games with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 6672–6679.

[27] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, I. Dunning, S. Mourad, H. Larochelle, M. G. Bellemare, and M. Bowling, "The hanabi challenge: A new frontier for AI research," *Artif. Intell.*, vol. 280, Mar. 2020, Art. no. 103216.

[28] K. Kurach et al., "Google research football: A novel reinforcement learning environment," in *Proc. AAAI*, vol. 34, no. 4, pp. 4501–4510, Apr. 2020.

[29] M. Zhou et al., "Smarts: An open-source scalable multi-agent RL training school for autonomous driving," in *Proc. Mach. Learn. Res.*, vol. 155, 2020, pp. 264–285.

[30] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 6252–6259.

[31] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings*, 1994, pp. 157–163.

[32] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[33] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[34] S. Kar, J. M. F. Moura, and H. V. Poor, "QD-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus + innovations," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1848–1862, Apr. 2013.

[35] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5872–5881.

[36] T. Doan, M. Siva, and R. Justin, "Finite-time analysis of distributed TD(0) with linear function approximation on multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1626–1635.

[37] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: A survey," *Artif. Intell. Rev.*, vol. 55, no. 2, pp. 895–943, Feb. 2022.

[38] J. Hu and M. P. Wellman, "Multi agent reinforcement learning: Theoretical framework and an algorithm," in *Proc. Int. Conf. Mach. Learn.*, 1998, pp. 242–250.

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

[40] V. Könönen, "Asymmetric multiagent reinforcement learning," *Int. J. Intell. Syst. Int. J.*, vol. 2, no. 2, pp. 105–121, 2004.

[41] M. L. Littman, "Friend-or-foe Q-learning in general-sum games," in *Proc. Int. Conf. Mach. Learn.*, vol. 1, 2001, pp. 322–328.

[42] R. A. C. Bianchi and A. L. C. Bazzan, "Combining independent and joint learning: A negotiation based approach," in *Proc. 11th Int. Conf. Auton. Agent. Multiagent Agent. Syst.*, vol. 3, 2012, pp. 1395–1396.

[43] Y. J. Park, Y. S. Cho, and S. B. Kim, "Multi-agent reinforcement learning with approximate model learning for competitive games," *PLoS ONE*, vol. 14, no. 9, Sep. 2019, Art. no. e0222215.

[44] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.

[45] L.-H. Xu, X.-H. Xia, and Q. Luo, "The study of reinforcement learning for traffic self-adaptive control under multiagent Markov game environment," *Math. Problems Eng.*, vol. 2013, pp. 1–10, Jan. 2013.

[46] N. Kodama, T. Harada, and K. Miyazaki, "Traffic signal control system using deep reinforcement learning with emphasis on reinforcing successful experiences," *IEEE Access*, vol. 10, pp. 128943–128950, 2022.

[47] A. Daeichian and A. Haghani, "Fuzzy Q-learning-based multi-agent system for intelligent traffic control by a game theory approach," *Arabian J. Sci. Eng.*, vol. 43, no. 6, pp. 3241–3247, Jun. 2018.

[48] M. Al-Turki, N. T. Ratrout, S. M. Rahman, and K. J. Assi, "Signalized intersection control in mixed autonomous and regular vehicles traffic environment—A critical review focusing on future control," *IEEE Access*, vol. 10, pp. 16942–16951, 2022.

HUI SHEN was born in Changping, Beijing. He received the M.S. degree in communication science and technology from Beijing Jiaotong University, in 2007. He is currently pursuing the Ph.D. degree with the North China University of Technology. He is also a Traffic Management Officer with the Beijing Municipal Traffic Management Bureau, working on traffic signal control and management. His research interests include traffic detection and traffic signal control.

HONGXIA ZHAO (Member, IEEE) received the Ph.D. degree in control theory and control engineering from the Chinese Academy of Sciences, Beijing, China, in 2009. She is currently an Assistant Professor with the State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences. Her research interests include traffic data analysis, traffic information processing, traffic modeling, and prediction.

ZUNDONG ZHANG received the Ph.D. degree in system engineering from Beijing Jiaotong University, Beijing, China, in 2010. He has been a Lecturer with the School of Electrical and Control Engineering, North China University of Technology, since 2013. He is currently teaching courses in transportation engineering, traffic simulation technology, traffic big-data analysis, and deep reinforcement learning. His research interests include traffic simulation, traffic signal control, multi agent systems, complex systems, deep learning, reinforcement learning, and game theory.

XUN YANG received the B.S. degree in traffic equipment and control engineering from the North China University of Technology, in 2023, where she is currently pursuing the master's degree. Her research interests include traffic simulation, signal control, deep learning, reinforcement learning, and evolutionary games.

YUTONG SONG received the bachelor's degree in traffic equipment and control engineering from North Industrial University, in 2023, where she is currently pursuing the master's degree. Her primary research interests include the study of deep reinforcement learning algorithms based on graph neural networks, the optimization of multi-objective traffic tasks, and research on signal control methods using intelligent connected vehicles.

XIAOMING LIU received the Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, in 2004. He is currently a Professor and the Head of the Department of Transportation Information and Control Engineering, School of Electrical and Control Engineering, North China University of Technology.