Expert Systems With Applications 235 (2024) 121242

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Lane-changing system based on deep Q-learning with a request–respond mechanism

Jian Guo [*], Istvan Harmati

Budapest University of Technology and Economics, H-1117 Magyar Tudosok krt 2, I building, Budapest, Hungary

ARTICLE INFO

Keywords:
Lane-changing
Deep Q-learning
Request–respond mechanism
Congested traffic
Computational complexity

ABSTRACT

Lane-changing control is a crucial task to regulate the traffic flow in the intelligent transportation system efficiently, and manipulating the lane-changing maneuvers for many vehicles optimally and simultaneously in a congested traffic environment is a challenge. This study formulates a lane-changing model where the traffic lanes are discretized into cells, considers both mandatory lane-changing and discretionary lane-changing maneuvers and proposes a novel approach based on a deep Q-network with a request–respond mechanism. The modeled lane-changing system is divided into isolated and symmetry groups, then independent Q-learning and the concept "central agent" are integrated to simplify the training process. In the proposed approach, these groups can be classified into two categories: request group and respond group. The request group and respond group are trained separately during the training process. Specifically, the request group trains agents with only considering the states of the group, while the respond group also evaluates the superimposing actions (i.e., the request message) from the request group besides the states of the group. The execution process is also treated the same decentralized way as the training process. Then, the baselines such as a rule-based method, a game theory-based decomposition algorithm and a simple deep Q-learning method are compared with the proposed approach. Results reveal that the proposed approach performs as well as the game theory-based decomposition algorithm and outperforms the other two in terms of lane-changing efficiency. However, the computational time of the proposed approach in the execution process is far less than the game theory-based decomposition algorithm, especially in a congested traffic environment.

1. Introduction

The mobility demand for vehicles has been increasing significantly, aggravating traffic congestion and causing unpredicted emergencies and accidents (Steger-Vonmetz, 2005). Developing intelligent transportation systems is an irresistible trend to improve the efficiency of regulating traffic flow. The rapid development of transportation management approaches, and models have been witnessed recently. Traffic light management system (Guo & Harmati, 2020), which optimally adjusts traffic lights, is one of the most critical fundamental tasks in the intelligent transportation system to reduce traffic congestion. Additionally, lane-changing models (Guo & Harmati, 2022) which address the need for lane-changing are also essential to ease traffic congestion and improve mobile safety.

Lane-changing is the primary driving task that affects the traffic flow characteristics and traffic safety significantly, e.g., traffic congestion (Arai & Sentinuwo, 2012), lane-changing crashes (Sen et al., 2003), traffic breakdown (Wall & Hounsell, 2005), etc. According to the motivations of lane-changing, lane-changing is classified as Mandatory Lane-changing (MLC) and Discretionary Lane-changing (DLC) (Halati, Lieu, & Walker, 1997). MLC is typically carried out to arrive at the planned destination, while DLC is adopted for better driving conditions. For safety reasons, the crash risk can be reduced by DLC models due to the higher complexity, compared with MLC (Vechione, Balal, & Cheu, 2018). In this study, both types of lane-changing (i.e., MLC and DLC) are combined to make the lane-changing system more realistic in a connected environment. Road condition detection (Subramani, Sattar, de Prado, Girirajan, & Wozniak, 2021; Woźniak, Zielonka, & Sikora, 2022) is not considered in the lane-changing system, assuming that all the vehicles are in a connected environment with shared road conditions information. It is also the trend in the future, which may be achieved due to the remote sensor network, communication technology and computer science technologies.

1.1. Literature review

Lane-changing models and approaches have received more attention in the literature to improve the problems mentioned above, and some

[*] Corresponding author.
E-mail addresses: guojian@iit.bme.hu (J. Guo), harmati@iit.bme.hu (I. Harmati).

https://doi.org/10.1016/j.eswa.2023.121242
Received 26 April 2022; Received in revised form 14 August 2023; Accepted 16 August 2023
Available online 19 August 2023

linear and nonlinear control approaches provide necessary enlightenment and significance for lane-changing system (Das, Tiwari, Sinha, & Khanzode, 2023; Preitl, Precup, Tar, & Takács, 2006; Rigatos, Siano, Selisteanu, & Precup, 2017). The typical lane-changing model applied in the freeway is the rule-based model proposed by Gipps (1986), which produces the lane-changing decisions with a set of sequential and deterministic rules. Whether the drivers intend to change a lane depends on the gap size of the adjacent lanes. It was reported that Gipps' model was examined in Australia. However, some assumptions of rule-based models are unrealistic when the traffic is congested. E.g., the lane-changing maneuver only occurs when the gap size is available so that the vehicles cannot change the lane without sufficient gaps in congested traffic. Then this model was improved for both urban streets and freeway (Hidas, 2005). The follower on the target lane will slow down to create an accepted gap for the subject vehicle (i.e., the lane changer) to insert. Kesting, Treiber, and Helbing (2007) proposed a MOBIL (i.e., minimizing overall braking induced by lane changes) model governed by safety and desirability rules, which simplifies the lane-changing process significantly with the single-lane accelerations. Other rule-based lane-changing models can be found in details (Zheng, 2014).

Integrating artificial intelligence into the lane-changing system is also a popular direction in recent research. For instance, the fuzzy logic-based model (Pan et al., 2020) converts the lane-changing rules into fuzzy rules as IF-THEN rules, which better mimics an actual decision process of a driver. The problem is that it is a challenge to define fuzzy sets and associated membership functions (Ross, 2005). Another primary example is combining game theory into the lane-changing models, where the lane-changing process is formulated as a non-zero-sum non-cooperative or cooperative game. In the initial stage, (Kita, 1999) proposed a game theory-based model, where a two-person non-zero-sum non-cooperative game is formulated, and actions such as merging and giveway are selected based on the payoffs. However, some limitations, such as constant speeds, small-range interaction, and multiple equilibria, may cause unrealistic consequences. Then an improved model is proposed in Liu, Xin, Adam, and Ban (2007), where the speed is a variable rather than a constant, and the equilibriums are searched more efficiently using a bi-level estimation method. Still, it is a two-person non-zero-sum non-cooperative game. The lane-changing model was extended to a three-player Stackelberg game by Yu, Tseng, and Langari (2018), miming human behavior and outperforming the rule-based method. Guo and Harmati (2022) proposed a novel game theory-based decomposition algorithm to solve a problem where large-scale players exist. The computational complexity is dramatically reduced by decomposing the lane-changing system into several minor games. The optimal or close-to-optimal solutions can be calculated for all the players by selecting consistent equilibrium according to different cases. Nevertheless, the computational time increases exponentially as the traffic density increases.

With the rapid development of machine learning, techniques such as Reinforcement Learning (RL), Deep Learning (DL), and Deep Reinforcement Learning (DRL) are combined into the lane-changing system. Compared with the game theory-based approaches, machine learning-based methods have the great advantage that the time of execution process is fixed and short after the training process. DRL method integrates RL and DL has the most potential to be applied in such a complex and dynamic system. As shown in Table 1, different DRL algorithms are designed to fit in the lane-changing system to enhance the lane-changing policy that ensures safety and efficiency. For instance, (Sharifzadeh, Chiotellis, Triebel, & Cremers, 2016) proposed an inverse reinforcement learning approach to extract the rewards with large state spaces. A single agent is constructed in a simulation-based autonomous driving scenario with full observation, and only the lateral actions are considered to train the model in the learning process. A hierarchical reinforcement learning with a policy gradient method and an "Option Graph" concept was formulated to generate long-term driving strategies (Shalev-Shwartz, Shammah, & Shashua, 2016), where the problem is decomposed into a series of policies for desires and trajectory planning to enable the comfort guarantee the safety of driving. Mukadam, Cosgun, Nakhaei, and Fujimura (2017) developed a single-agent framework using DRL solely to obtain a high-level policy for tactical decision-making while maintaining tight integration with the low-level controller. A technique called Q-masking is applied to simplify the reward function and improve the data efficiency, and five basic actions are taken to break down the high-level decisions. Similarly, (Chen et al., 2018) proposed a new DRL architecture to improve the training efficiency, and Jiang, Chen, and Shen (2019) used DRL to solve a Partial Observed Markov Decision Process (POMDP) and quintic polynomials-based motion planning algorithm to obtain both optimal lateral and longitudinal trajectory. A DRL framework using Monte Carlo tree search and an AlphaGo Zero-based algorithm extending to a domain with a continuous state space were applied in two highway driving cases with partial observation (Hoel, Driggs-Campbell, Wolff, Laine, & Kochenderfer, 2019). Double deep Q-network was also applied with the proposed driving policy to minimize or eliminate assumptions of the environment, and a single-agent framework was implemented to compare the optimal policy derived via Dynamic Programming with the proposed policy (Makantasis, Kontorinaki, & Nikolos, 2019). An and Jung (2019) developed a lane change system based on a deep deterministic policy gradient learning algorithm, where the approach determines the high-level action for a host vehicle trying to change lanes from the state information. However, it only considers a straight road and stable steering when the host vehicle is running, and only one remote vehicle is considered. Ye, Cheng, Wang, Chan, and Zhang (2020) proposed an automated lane change strategy based on DRL using proximal policy optimization to improve the learning efficiency while maintaining stable performance, where a single agent is trained to learn a smooth, safe, and efficient driving policy. A decentralized cooperative lane-changing controller and a multi-agent DRL paradigm were developed using proximal policy optimization to improve the performance (Hou & Graf, 2021). Meanwhile, each agent can access global state information and evaluate policy learning and action reward locally. Dong et al. (2021) integrated the data collected through sensing and connectivity capabilities from other vehicles and those located further downstream to optimize the lane-changing strategies and implemented a double deep Q-network to enhance safety and mobility in a given traffic environment. The information such as absolute or relative position, speed, and distance between adjacent vehicles constitutes the state space, and whether the state is full or partial observable depends on the type of the DRL algorithms. In general, the action space comprises three basic actions known as lateral actions, i.e., keep the current lane, turn to the left lane and turn to the right lane. Some lane-changing models add longitudinal actions such as acceleration and deceleration to make the system more realistic, as shown in Table 1. Moreover, only a few studies model the lane-changing process as a multi-agent system, while most existing lane-changing literature considers a single-agent system.

1.2. Research gaps

In the existing lane-changing literature, the rule-based approaches regulate the traffic flow with a series of sequential and deterministic rules. The lane changer only changes the lane when the target lane has an available gap, reducing lane-changing efficiency in congested traffic. In other words, the traffic system could not provide sufficient gaps for changing a lane efficiently when the traffic density grows (i.e., more vehicles enter the lane-changing system in a specific period). Thus, the rule-based approach cannot fit the lane-changing system in a congested traffic scenario.

The game theory-based models are much more efficient regarding lane-changing maneuvers than rule-based lane-changing systems.

Table 1
Existing lane-changing models based on deep reinforcement learning.

DRL-based models	Algorithm types	Single or multi-agent	States observability	Actions design (Lateral or longitudinal)
Sharifzadeh et al. (2016)	IRL	Single-agent	Full	Only lateral
Shalev-Shwartz et al. (2016)	HRL	Multi-agent	Full	Both
Mukadam et al. (2017)	DQN	Single-agent	Full	Both
Chen et al. (2018)	DQN	Multi-agent	Full	Only lateral
Jiang et al. (2019)	DQN	Single-agent	Partial	Only lateral
Hoel et al. (2019)	AlphaGo policy-based	Single-agent	Partial	Both
Makantasis et al. (2019)	Double DQN	Single-agent	Full	Both
An and Jung (2019)	DDPG	Single-agent	Full	Only lateral
Ye et al. (2020)	PPO	Single-agent	Full	Both
Hou and Graf (2021)	PPO	Multi-agent	Full	Both
Dong et al. (2021)	Double DQN	Single-agent	Full	Only lateral

IRL: Inverse reinforcement learning; HRL: Hierarchical reinforcement learning; DQN: Deep Q-network; DDPG: Deep deterministic policy gradient; PPO: Proximal policy optimization.

The lane-changing decision process is typically constructed as a non-zero-sum non-cooperative, or cooperative game, which generates the optimal action corresponding to the calculated equilibria. However, these models have a common disadvantage, i.e., the computational complexity of the system increases exponentially as the number of players increases. For instance, Guo and Harmati (2022) analyzes the space complexity of the system when the number of players grows. In contrast, the time complexity is tricky to formulate since finding an equilibrium solution is unknown (Daskalakis, Goldberg, & Papadimitriou, 2009). Although the computational complexity is reduced dramatically in Guo and Harmati (2022) by decomposing a large game into small games, the computational time still increases significantly as the traffic density increases. Therefore, finding a suitable lane-changing approach is necessary to improve lane-changing efficiency and reduce computational time.

The recent DRL-based lane-changing models enumerated in Table 1 overcome the problem of game theory-based lane-changing models (i.e., long computational time with a large number of players). Even though the DRL-based models face a similar problem with many agents in the training process, the time of execution process is fixed and short after the training process. Overall, the main advantages and benefits of learning-based methods are improving lane-changing efficiency and reducing computational time compared with the general non-learning control approaches. However, almost existing studies only consider a single agent, which is unrealistic in a scenario such as urban streets where many vehicles must change lanes before arriving at the traffic light. Although (Chen et al., 2018; Hou & Graf, 2021) model the lane-changing system as a multi-agent system, it is executed in a decentralized way. That is unsuitable in congested traffic where the agent has a strong connection with other agents, and the action made by the agent directly affects other agents. To fill this gap, a novel hierarchical approach based on deep Q-learning with a request and respond mechanism is proposed to fit in such a lane-changing system with a significant scale agent. The details are presented in Section 3.

1.3. Main contribution

The main contribution of this study is (a) a lane-changing model is discretized into cells, and both MLC and DLC maneuvers are considered to maximize the utilization of available gaps so that the traffic congestion could be eased. (b) the lane-changing system can be formulated as several sequential and symmetry groups, so independent deep Q-learning and the concept "central agent" could be integrated to handle such a large-scale problem. (c) A "virtual agent" concept is introduced to train the group consistently with the shared network parameters since the number of agents in each group is dynamic in this lane-changing system. (d) a novel control approach is proposed based on deep Q-learning with a request and respond mechanism. The novelty of this approach is mainly shown in a request and respond mechanism. Specifically, two groups, the request group and the respond group, are trained separately, and the execution process is decentralized. During the training process, the request group sends a message to the respond group to speed up the lane-changing process and provide agents with more opportunities to change lanes. The proposed approach reflects high lane-changing efficiency in congested traffic scenarios compared with the rule-based and typical deep Q-learning methods and much less computational time in the execution process than game theory-based and other approaches. (e) some baselines are compared to the proposed approach, and the results suggest that the proposed method has great potential.

The remainder of the paper is organized as follows: Section 1 reviews the rule-based, game theory-based, and DRL-based lane-changing models, analyzes the major issue of existing models and introduces the main contribution. Section 2 formulates a lane-changing model with basic formulas and assumptions. Section 3 presents the details of the proposed approach and shows some examples. Section 4 demonstrates and compares the simulated results between the baselines and the proposed method. Section 5 discusses the reasons behind those results, the potential applications and the proposed approach's limitations. Section 6 summarizes the main findings and discusses some future research directions.

2. Model formulation

A regular traffic lane-changing system contains five traffic lanes, which can be discretized into cells shown in Fig. 1. The position of the cells can be defined as $P(i,j)$, where $i \in N = \{1,2,\dots,n\}$ is the x-axis corresponding to the index of the traffic lanes and $j \in M = \{1,2,\dots,m\}$ is the y-axis corresponding to the number of cells in each traffic lane. It is clear for $n = 5$ since the number of traffic lanes in this lane-changing system is five, and the traffic lane's width is the same. The number of cells in each traffic lane m depends on the factors such as limited speed, safety distance and driving rules so that it can be specified as a different value. Each cell can only have one vehicle; otherwise, the collision happens. The states of the cell can be represented as follows:

$$P(i,j) = \begin{cases} 0, & \mathit{Empty} \quad \text{(a)} \\ 1, & \mathit{One\ vehicle} \quad \text{(b)} \\ > 1, & \mathit{Collision} \quad \text{(c)} \end{cases} \tag{1}$$

As shown in Fig. 1, traffic lanes have different turning functions, and the vehicles must complete the lane-changing maneuver to participate in the next turning (e.g., arrive at a traffic light and pass through a traffic intersection). Specifically, there are four turning functions for the traffic lanes, i.e., lane 1 represents U-turn and left-turn, lane 2 represents left-turn, lane 3 and lane 4 represents go-straight, lane 5 represents right-turn. The red mark in Fig. 1 attached to vehicles represents the target lane, which corresponds to the specific lane where the vehicle is going (i.e., U: U-turn, L: left-turn, S: go-straight, R: right-turn). Therefore, the value of the target lane carried by a vehicle at the

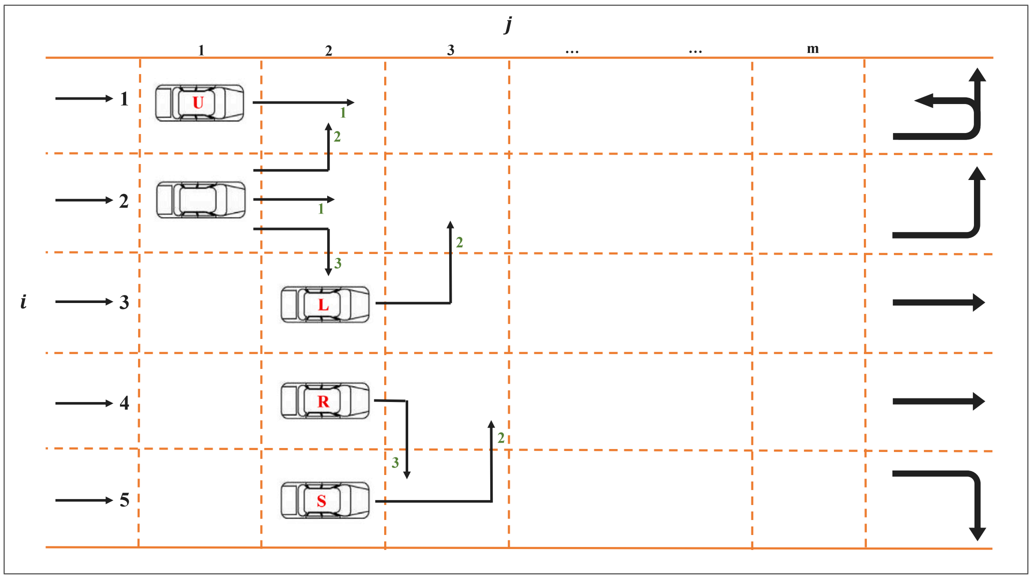

Fig. 1. The schematic of the lane-changing system.

position $T(i,j)$ can be expressed as follows:

$$T(i,j) = \begin{cases} \text{U} \in \{1\} & \text{(a)} \\ \text{L} \in \{1,2\} & \text{(b)} \\ \text{S} \in \{3,4\} & \text{(c)} \\ \text{R} \in \{5\} & \text{(d)} \end{cases} \tag{2}$$

where elements of the sets in (2) represent the target lane that matches the turning function, and it can be converted to a digit corresponding to the index of the traffic lane. For instance, the element S can be value 3 or 4 since lane 3 and lane 4 have the same turning function (i.e., go-straight). Thus, the vehicle with an element S can either go lane 3 or lane 4 to complete a MLC before arriving at a traffic light. Then this vehicle will consider a better driving condition according to the surrounding traffic information. It may stay at the current lane or change to the adjacent lane with the same turning function (i.e., a DLC). Additionally, the actions of the vehicles are also important in the lane-changing system, which determines how the vehicles arrive at the target lane. The basic actions are shown in Table 2, they are $a[1]$-forward, $a[2]$-turn left, $a[3]$-turn right, respectively. Each action has the corresponding lateral and longitudinal velocity, which are also specified in Table 2. Fig. 1 shows examples of the process of executing an action by the vehicle, e.g., the vehicle with an element L at the position $P(3,2)$ intends to change to the target lane 1 or 2 with the action-turn left. The vehicle will arrive at the position $P(2,3)$ at the next step, and then a DLC is considered whether it needs to change to lane 1 or stay on the current lane 2 based on the surrounding data. Similarly, the vehicle at position $P(1,1)$ executes the action-forward and will arrive at position $P(1,2)$ at the next step. After executing the action, the new position of the vehicle and the target lane's value can be updated:

$$P_{t+1}(i',j') = f(P_t(i,j), a_{i,j}), \forall i \in N, \forall j \in M \tag{3}$$

$$T_{t+1}(i',j') = g(T_t(i,j), a_{i,j}), \forall i \in N, \forall j \in M \tag{4}$$

where $f(\cdot)$ and $g(\cdot)$ are functions that maps the previous portion $P_t(i,j), T_t(i,j)$ to the new position $P_{t+1}(i',j'), T_{t+1}(i',j')$ with the actions $a_{i,j}$, respectively.

The position of each vehicle can be updated based on the executed action from (3) and (4) at each step, so the position of all vehicles in the lane-changing system can be figured out. Thus, a new variable q can be defined, i.e., queues. The queues represent the total number of vehicles staying on the traffic lane, and q_i is the queue of lane i, $\forall i \in N$. It can be represented as follows:

$$q_i = \sum_{j=1}^{m} P(i,j), \forall i \in N, \forall j \in M \tag{5}$$

This lane-changing system cannot avoid the existing constraints issue, i.e., the collision problem. Generally, the collision cases can be considered in two categories: (1) path collision. The collision could happen on the trajectory where vehicles are driving from different traffic lanes. (2) position collision. The collision occurs at the same position where vehicles from separate traffic lanes arrive. In this lane-changing problem, the case of path collision is neglected to simplify the collision problem. It is supposed that the vehicles could pass across to each other smoothly even with parts of the overlapped trajectory (e.g., the vehicle in the position $P(4,2)$ can turn right and the vehicle in the position $P(5,2)$ can turn left simultaneously, shown in Fig. 1). The path planning and collision avoidance algorithms have been proposed and developed in recent years (Li, Luo, & Wu, 2019; Liu, Lee, Varnhagen, & Tseng, 2017; Wang et al., 2019), which is not the scope of this study. Therefore, the position collision is only considered in this lane-changing system, and the collision state is represented in (1) shown above.

3. Proposed method framework

In this section, the deep Q-network approach is designed to optimize the lane-changing problem (i.e., generate the optimal strategies to organize the driving vehicles to arrive at the target lanes as soon as possible) with a short computational time in a multi-agent system. Furthermore, a novel multi-agent reinforcement learning (MARL) method based on Deep Q-network with a Request-respond Mechanism (DQNRR) with a request and respond mechanism is proposed to improve the efficiency of the lane-changing system, where more lane-changing demands could be satisfied in a specific period with the congested traffic environment.

3.1. Q learning

The Q-learning algorithm is a common model-free approach that learns the value of an action a in a particular state s. The Q-function called the state–action value function, estimates a policy π which maps to action from the current state, and it can be represented as follows:

$$Q^{\pi}(s,a) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} | s_t = s, a_t = a] \tag{6}$$

Table 2
Basic actions of vehicles.

Index (ℓ)	Actions ($a[\ell]$)	Lateral velocity (cells/step)	Longitudinal velocity (cells/step)
1	Forward	0	+1
2	Turn left	−1	+1
3	Turn right	+1	+1

where k is the iterations, and r_{t+1} is a reward at the time step $t+1$ received by taking action a_t in the state s_t at the time step t; γ is a discount factor, and $\sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$ is the sum of future discounted rewards. The discount factor $\gamma \in [0,1]$ indicates that future rewards are weighted less than the immediate reward. It is considered short-sighted when $\gamma = 0$ and far-sighted as γ closes to 1.

The Q-function can be updated as in (7):

$$Q_{t+1}(s_t, a_t) = (1-\alpha)Q_t(s_t, a_t) + \alpha[r_t + \gamma \max_a Q_t(s_{t+1}, a)] \tag{7}$$

where $\alpha \in [0,1]$ is the learning rate that determines to what extent newly acquired information overrides old information. A factor of 0 makes the agent learn nothing, while one makes the agent consider only the most recent information. Watkins and Dayan (1992) proved that the Q-function would converge to an optimum defined as $Q^*(s,a)$ after the states and actions are visited infinitely with some other assumption.

The action selection mechanism during the learning process is based on the $\varepsilon - greedy$ algorithm, which is shown as follows:

$$a_t = \begin{cases} \arg\max_a Q(s_t, a), & 1-\varepsilon \quad \text{(a)} \\ RandomAction, & \varepsilon \quad \text{(b)} \end{cases} \tag{8}$$

$$\varepsilon = e^{-\beta t} \tag{9}$$

where $\varepsilon \in [0,1]$ is the exploration probability, determining how fast the agent explores the external environment. An action corresponding to the maximum Q-value is exploited by the agent with the probability of $1-\varepsilon$, while a random action is selected to explore with probability ε. As shown in (9), the probability of exploration decreases as the time step increases since the Q-values tend to be convergent. The attenuation rate β is a constant.

Generally, the Q-function is replaced by a table that stores the Q-values corresponding to the state and action. The action corresponding to the maximum Q-value can be selected based on the input state by looking up a tabular Q-function. However, creating such a table with large state and action spaces is unrealistic, which consumes too much storage and computation resources.

3.2. Deep Q-network

Generally, the method of Q-learning can solve the problem where the state and action spaces are quite small. Otherwise, a "curse of dimensionality" problem shows as the space of states and actions expands, and it is impossible to create such a large Q-value table for storing and searching the corresponding data. Deep Q-network (DQN) method is developed based on Q-learning, which uses multi-layer neural networks to approximate the Q function instead of using a tabular Q function in Q learning. Thus, DQN can ease this problem when the states and actions space is too large. Specifically, the action-value function $Q(s,a)$ is parameterized as $Q(s,a;\theta_t)$ with using the deep convolutional neural network, where θ_t is the weight of the neural network at iteration t. DQN uses two separate neural networks: the target and the online neural network, for the Q function updating instead of using a single Q function in (7). The states are the input for training, and the output is the optimized Q-value corresponding to the current pair of state and action. These two neural networks can calculate the target and estimation value, respectively. Then the loss function is updated as follows:

$$\begin{aligned} L_t(\theta_t) &= E_{s_t,a_t,r_t}[y_t - Q(s_t, a_t; \theta_t)]^2 \\ &= E_{s_t,a_t,r_t,s_{t+1}}[r_t + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; \hat{\theta}_t) - Q(s_t, a_t; \theta_t)]^2 \end{aligned} \tag{10}$$

where $y_t = r_t + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; \hat{\theta}_t)$ is the target value depending on the weights of the neural network, γ is the discount factor which is mentioned before, $\hat{Q}(s_{t+1}, a_{t+1}; \hat{\theta}_t)$ is the target Q-network, and $\hat{\theta}_t$ are the parameters used to compute the target Q-network at iteration t. The target network parameters $\hat{\theta}_t$ are only updated with the Q-network parameters θ_t every $\mathbb{X}$ step and are fixed between individual updates. The loss function is minimized to update the Q-network by using the stochastic gradient descent method.

Moreover, a replay buffer that stores the experiences is used to update the online network in DQN. Specifically, mini-batches (e.g., samples) are selected randomly and uniformly from the buffer of stored experiences to update the Q-network with the loss function in (10). That is different from the Q function updating process in Q-learning that only uses the last obtained experience. The target network is updated periodically instead of every step to stabilize and speed up the training process. It can alleviate the high correlation between consecutive experiences and overcome neural networks' "forgetfulness" problem during the learning process. Besides, the experiences stored in the replay buffer can be relearned multiple times, making data utilization more efficient.

3.3. The request–respond mechanism

The lane-changing system can be modeled as a cooperative multi-agent system where the agents would take optimal joint actions to maximize a global and long-term reward in a dynamic environment for the whole system. In this lane-changing system, some components are defined and described as follows:

- **Agent:** Each independent vehicle driving on the lanes is regarded as an agent.
- **State:** A state s_i of agent i consists of the value of the current lane where the agent is driving and the target lane where the agent is going. All the states of agents constitute the state space $S =< s_1, s_2, \dots, s_n >$.
- **Action:** An action a_i of agent i from Table 2 is selected to decide if the agent would stay on the current lane or change to another lane at each time step. The action space $A =< a_1, a_2, \dots, a_n >$ consists of the joint actions of all agents.
- **Policy:** A policy $\pi_i = \mathbb{P}(a_i|s_i)$ indicates a probability of taking action a_i of agent i, given state s_i.
- **Reward:** A reward r_i is donated to agent i after the agent i take the action a_i, given in the state s_i. The reward function can be defined as a negative value of the absolute distance from the current lane to the target lane. The global reward is the sum of the reward of all agents, i.e., $\sum_i r_i$.

There are many vehicles (i.e., agents) in this lane-changing system, and the actions taken by the agents affect each other. Thus, the independent Q-learning (IQL) (Tan, 1993), where each agent learns its policy and ignores the influence of others, cannot be applied in this system despite its simplicity. For a fully observable (i.e., the state space $S =< s_1, s_2, \dots, s_n >$) and cooperative multi-agent problem, the "central agent" who selects the joint actions $A =< a_1, a_2, \dots, a_n >$ for all the agents are introduced. The central agent learns Q-function $Q(S,A) = Q(s_1, s_2, \dots, s_n, a_1, a_2, \dots, a_n)$ according to the surrounding environment. However, the state space and action space expand dramatically as the number of agents increases, which directly results in exponentially growing the computational complexity of the system.

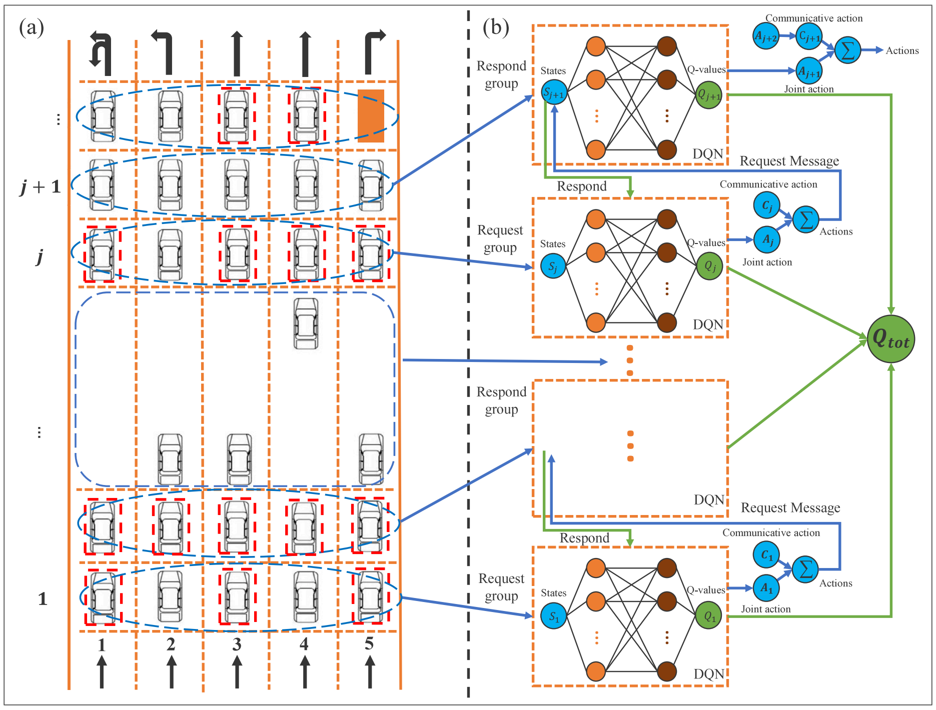

Fig. 2. Schematic of deep Q-network with request–respond mechanism.

Independent deep Q-learning and the “central agent” concept can be combined to solve such a scalable problem. In specific, as shown in Fig. 2(a), the whole lane-changing system can be decomposed into groups defined as $G = \{g_1, g_2, \dots, g_j, \dots, g_m\}, j \in M = \{1, 2, \dots, m\}$, each row is regarded as a group and the vehicles in the same row can be considered as the agents in the same group. Therefore, the agents in the same group take joint actions like a central agent since the maximum number of the agents is not large (e.g., 5). Generally, the group g_j close to the intersection has higher priority, which means the agents with the higher priority take the joint actions first. The group that follows the front one will take the corresponding actions based on both the states of the environment and the actions taken by the previous group. Thus, the actions of the groups comply with the independent deep Q-network since the connections between the groups are independent due to the basic actions which cannot affect each other between groups. DQN is implemented for each group g_j, and the parameters are shared between different groups since every group has the same properties. Thus, the loss function of DQN for each group can be extended from (10) and converted into:

$$L_{j,t}(\theta_{j,t}) = E_{d_{j,t}\backsim D}[R_{j,t} + \gamma \max_{A_{j,t+1}} \hat{Q}(S_{j,t+1}, A_{j,t+1}; \hat{\theta}_{j,t}) - Q(S_{j,t}, A_{j,t}; \theta_{j,t})]^2 \tag{11}$$

where $d_{j,t} = (S_{j,t}, A_{j,t}, R_{j,t}, S_{j,t+1})$ is a tuple which stores the experience in the replay buffer D, j is the sequence of the group, $S_{j,t} =< s_{1,j}, s_{2,j}, \dots, s_{i,j}, \dots, s_{n,j} >$ is the state space of all the agents in the same group j at the time step t and $A_{j,t} =< a_{1,j}, a_{2,j}, \dots, a_{i,j}, \dots, a_{n,j} >$ is the joint actions of all the agents in the same group j at the time step t. As mentioned previously, $\hat{\theta}_{j,t}$ and $\theta_{j,t}$ are the parameters of the target network $\hat{Q}$ and online network Q, respectively. The reward of group j is the sum of the distance from the current lane to the target lane for all the agents, which can be represented as:

$$R_{j,t} = \begin{cases} \sum_{i=1}^{n} |T'(i,j) - i|, & No\ collision \quad (a) \\ u_{min}, & Collision \quad (b) \end{cases} \tag{12}$$

where $T'(i,j)$ represents the converted value of target lane with considering MLC and DLC for the vehicle at the position (i,j). The absolute difference between the target lane $T'(i,j)$ and the current lane i indicates how much the vehicle desires a MLC to narrow the distance between the current lane and target lane. The reward value increases as the vehicle approach the target lane, and the vehicle stays on the target lane if the difference is zero (i.e., maximum). However, there are multi-choices for vehicles to choose a target lane since there are multiple lanes for the same turning function (e.g., lanes 1 and 2 for turning left, lanes 3 and 4 for going straight shown in Fig. 1). That means DLC can be considered for changing to the lane where fewer vehicles are in front of the current vehicle. In such a cooperative environment, the value of target lane $T(i,j)$ can be converted into a new value $T'(i,j)$, i.e.,

$$T'(i,j) = \begin{cases} T(i,j), & if \quad T(i,j) = \mathrm{U}\{1\}, \mathrm{R}\{5\} \quad (a) \\ \arg\min_{i} \sum_{j=n-j}^{m} P(i,j), & Otherwise \quad (b) \end{cases} \tag{13}$$

where U{1} represents U-turn with the value one and R{5} represents right-turn with the value 5; $\sum_{j=n-j}^{m} P(i,j)$ is the queue in front of the vehicle at the position $P(i,j)$. The target value remains for the traffic lanes with the unique turning function (i.e., U-turn and right-turn). Otherwise, the new target value is updated to the value of the traffic lane corresponding to the minimal queue, which indicates a DLC is considered for drivers to have a better driving condition.

Another problem of the lane-changing system is that the number of agents for each group is dynamic, resulting in the training process of the DQN being changed and adapted in each group. That means different Q-networks are utilized to cope with different situations where the groups have different agents. There are 5 cases with the number of agents 1–5 in this lane-changing system, so the corresponding Q-networks should also be five. It is inconvenient and inefficient to train all the agents in the lane-changing system with such several Q-networks, so the “virtual agent” concept is introduced into the training process. As shown in Fig. 2(a), the vehicles in the red-dotted rectangle are virtual agents. The virtual agents ensure that the number of agents is constant during training and do not influence the environment. After the training process, the method with a simple DQN is implemented in the later chapter, and the result is compared with other methods.

Table 3
Extended actions of vehicles.

Index (ℓ)	Action ($a[\ell]$)	Horizontal velocity (cells/step)	Vertical velocity (cells/step)
4	Accelerate ($a[1]+a[1]$)	0	+2
5	Accelerate and Turn Left ($a[2]+a[1]$)	−1	+2
6	Accelerate and Turn Right ($a[3]+a[1]$)	+1	+2

However, the lane-changing efficiency may get affected since the independent DQN cuts the entire connections between groups by taking the basic actions in Table 2. The basic actions are extended by superimposing the action $a[1]$ (i.e., from Tables 2 to 3), and the request–respond mechanism is built based on DQN so that a connection between groups could be built. As shown in Fig. 2, each group is trained by a Q-network, and there are two types of Q-networks (i.e., defined as request Q-network and respond Q-network) which correspond to the request group and the respond group, respectively. The details and examples are described as follows:

- **Request group:** If agents still do not arrive at the target lane in a group after executing the basic joint actions generated from the request Q-network, then the group regarded as a request group (e.g., group j in Fig. 2(b)) will send a request message to the group in the front regarded as a respond group (e.g., group $j+1$ in Fig. 2(b)). For instance, as shown in Fig. 2(b), the group j is a request group. Specifically, this group takes extended actions that can be decomposed into two stages. The basic joint actions A_j produced by request Q-network and a Q-value Q_j are executed at the first stage. Then a communicative action vector C_j superimposing the basic joint actions constitutes the request message (i.e., $A_j + C_j$). The communicative action C_j encoded by the binary bits indicates the current state of the agents of the request group j (i.e., 0 represents the agent is on the target lane, 1 represents not on the target lane, and intend to take action $a[1]$). E.g., the code $C_j = (0,0,1,1,0)$ means the 3th and 4th agents do not arrive at the target lane yet and intend to forward again to look for an opportunity to change lane. The respond group $j+1$ must confirm the final execution of the communicative action C_j, i.e., checking whether an available gap exists at corresponding traffic lanes in the respond group. The training process of the request group is the same as a normal DQN method, so the loss function and reward function of the request group can be referred to (11) and (12), respectively.
- **Respond group:** A respond group will take the react action generated from the respond Q-network to maximize the Q-values when the request group receives the request message. As shown in Fig. 2(b), the group $j+1$ in front of the request group is a respond group. The respond Q-network is different from the request Q-network, i.e., both of the local observations from states of the group $j+1$ and the extended actions of the group j are part of the input of the respond Q-network. Suppose the request of the corresponding agent is accepted (i.e., an available gap exists at corresponding traffic lanes in the respond group). In that case, the corresponding lane of the respond group is available, and then an extra reward is added to the reward function of the respond Q-network. The new reward function can be converted from (13) as follows:

$$R'_{j+1,t} = \begin{cases} -\sum_{i=1}^{n}(|T'(i,j+1)-i| + r_c \cdot P(i,j+1)\cdot c_{i,j}), & No\ collision \quad (a) \\ u_{min}, \quad Collision & (b) \end{cases} \tag{14}$$

where r_c is a constant reward, and $c_{i,j}$ as a binary code is the communicative action of agent i in group j. The reward r_c affects the function only if the agents accept the request from the request group. The respond Q-network is also updated in this respond group, defined as $Q(S_{j+1,t}, A_{j,t}+C_{j,t}, A_{j+1,t};\theta_{j+1,t})$. The respond Q-network aims to minimize the loss function, which is derived from (10):

$$\begin{aligned} L'_{j+1,t}(\theta'_{j+1,t}) = E_{d'_{j+1,t}\sim D'}[R'_{j+1,t} \\ + \gamma \max_{A_{j+1,t+1}} \hat{Q}(S_{j+1,t+1}, A_{j,t+1}+C_{j,t+1}, A_{j+1,t+1};\hat{\theta}'_{j+1,t}) \\ - Q(S_{j+1,t}, A_{j,t}+C_{j,t}, A_{j+1,t};\theta'_{j+1,t})]^2 \end{aligned} \tag{15}$$

where $d'_{j+1,t} = (S_{j+1,t}, A_{j,t}+C_{j,t}, A_{j+1,t}, R'_{j+1,t}, S_{j+1,t+1})$ is a tuple which stores the experience in the replay buffer of respond Q-network D', $\hat{\theta}'_{j+1,t}$ and $\theta'_{j+1,t}$ donate the parameters of the target Q-network and Q-network of the respond group, respectively. Above all, The overall training algorithm of the respond group can be presented in the proposed Algorithm 1.

As shown in Fig. 2(b), the request Q-network and respond Q-network are trained independently for the request and respond groups, respectively. The request group and the respond group are disjoint at each time step t and may convert to each other at the next time step $t+1$. The request group is trained using a normal DQN discussed above, and the respond group is trained separately, shown in Algorithm 1. For each episode, the joint actions A_j from the request group will be selected, and a communicative action C_j is superimposed to construct an extended action: the request message. At each time step t ranged from 1 to $\mathbb{T}$, the agents in respond group select the actions based on the ϵ-greedy policy. The joint actions corresponding to the maximum Q-values will probably be selected according to the current states and the request messages as the Q-function converges. After executing the action for each agent, the group's reward is calculated, and the new state of the environment is updated. The experience is then stored in a replay buffer for the next training. Finally, the parameters of the current Q-network are updated by minimizing the loss function, and the parameters of the target Q-network are updated for every $\mathbb{X}$ step to improve the training efficiency.

Since the groups are trained under a decentralized framework, the execution process is treated as decentralized as the training process. The situation can be divided into two cases during the execution process shown in the proposed Algorithm 2: (1) If the following group in the lane-changing system does not have a request, then the current group output the optimal joint actions by using the trained request Q-network according to the current local observation (i.e., states). Moreover, a communicative vector $C_{j,t}$ is superimposed based on the joint actions of the front group. (2) Otherwise, the request group selects the joint actions by using request Q-network, and the joint actions are extended to $A_{j,t}+C_{j,t}$ which is a request message by superimposing an action vector $C_{j,t}$. Using the respond Q-network, the respond group will select a joint action $A_{j+1,t}$ based on the current states and the request message. After finalizing the joint actions of the respond group, the respond group confirms the request for the lanes with available gaps and responds to the request group. Finally, the request group reselects $C_{j,t}$ with the confirmation from the respond group and superimposes $C_{j,t}$ into $A_{j,t}$ to form the final joint actions of the request group. Due to the structure of the combination of the independent deep Q-network and central agent in Fig. 2(b), the total Q-values for one iteration in this lane-changing system can be additively decomposed into the value function across groups:

$$Q((S_1, S_2 ..., S_j), (A_1, A_2, \dots, A_j)) \approx \sum_{j=1}^{J} Q_j(S_j, A_j) \tag{16}$$

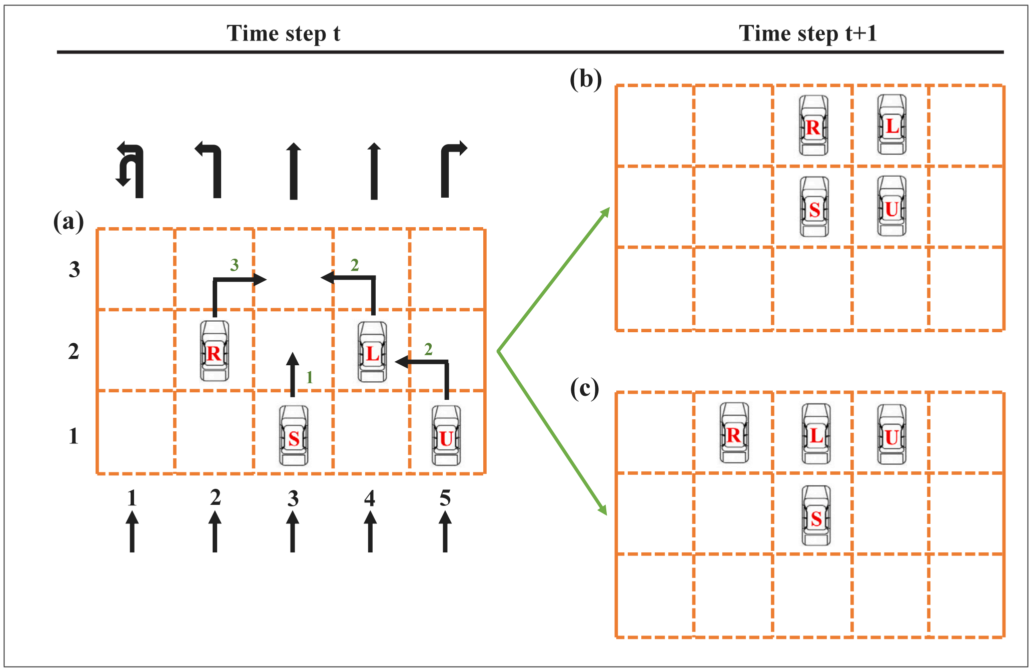

Fig. 3. A example of deep Q-network with a request–respond mechanism.

After a training process referred to Algorithm 1, then Algorithm 2 is executed to optimize the actions of agents. As shown in Fig. 3, one example of DQNRR in the lane-changing system is illustrated. Fig. 3(a) is the initial state of the agents at the time step t, where four agents are in the area. Group g_2 has two agents (i.e., an agent at position $P(2,2)$ intends to have a right turn and the other at position $P(4,2)$ intends to have a left turn) who both intend to change to the position $P(3,3)$. Group g_3 also has two agents, i.e., an agent at position $P(3,1)$ intends to go straight, and the other at position $P(5,1)$ intends to have a left turn. It cannot happen that both the agents in the group g_2 go to the position $P(3,3)$ since it causes a collision. Thus, either one can go to the position $P(3,3)$, and the reward is the same for the group g_2 since it is symmetry. However, the situation can be different with the algorithm DQN and DQNRR, shown in Fig. 3(b) and (c) respectively. Fig. 3(b) shows that the group g_2 close to the intersection has higher priority and probably selects a right-turn action for an agent at the position $P(2,2)$ and a go-straight action for the agent at the position $P(4,2)$ to avoid a collision. Then the same work mechanism is implemented for group g_1. The situation changes in the case of DQNRR in Fig. 3(c). Firstly, group g_1 selects a joint action A_1 based on the request Q-network in the same way in Fig. 3(b), which means the agent at the position $P(3,1)$ would arrive at the position $P(3,2)$ and the agent at the position $P(5,1)$ would arrive at the position $P(4,2)$. Then, the communicative action C_1 is superimposed into A_1 to form a request message. In this case, $C_1 = (0,0,0,1,0)$ since only one agent that would be at the future position $P(4,2)$ is not on the current lane, this agent with the letter "U" is going to lane 1 eventually. Finally, the respond group g_2 will select the joint action A_2 based on the respond Q-network and confirm the request message after the request message is received. Specifically, the agent with the letter "L" will arrive at the position $P(3,3)$ to create an empty position and confirm the request from the group g_1, so the agent with the letter "R" has to go straight to avoid a collision. Additionally, with the response group's confirmation, the agent with the letter "U" will take the communicative action and arrive at position $P(4,3)$. As can be seen from the example, the case of DQRNN has more potential to speed up the lane-changing process and provides more opportunities for a single agent to change lanes in different groups.

4. Simulation

The simulation of a lane-changing system with the control methods is implemented in the environment of a 2.80 GHz Intel core central process unit, 16 GB random-access memory and Windows 10 operating system. The software environment of the lane-changing system is set and run in the platform Matlab R2021b, and the training process of deep Q-learning is run in the platform Python 3.6. The performance of the proposed method (i.e., DQNRR) and the baselines are evaluated and compared with the same environment in this section.

4.1. Experimental setup

4.1.1. Parameters setting

All the experiments were carried out with 500 iterations defined as $\mathbb{K}$, and all the input data are randomly generated in this lane-changing system. For each iteration, the number of incoming vehicles will show randomly at different traffic lanes. Meanwhile, each incoming vehicle carries a target lane's value generated randomly. To compare the performance with the baselines from (Guo & Harmati, 2022), all the input data stream is from (Guo & Harmati, 2022). Specifically, six different categories of experiments with different expected traffic density (i.e., $\rho = 0.36, 0.42, 0.48, 0.54, 0.60, 0.66$ were implemented, and each category contains five randomized data sets.

A standard parameter evaluating the level of congestion is defined as traffic density $\rho = \mathbb{V}/\mathbb{C}$, where $\mathbb{V}$ is traffic volume and $\mathbb{C}$ is traffic capacity. Traffic volume and capacity are the number of passing vehicles and the maximum number of accommodating vehicles per section per period, respectively. The expected rate E_r represents the vehicles that enter the lane-changing system per iteration, regarded as traffic volume. The maximum incoming vehicles per iteration n can be considered traffic capacity. Thus, the expected traffic density can be expressed as $\rho = E_r/n$. The lane-changing rate defined as L_r is significantly affected by a large ρ according to Guo and Harmati (2022) and Huang (2002) since the system cannot provide the vehicles with many available gaps, which is detrimental to perform and compare the lane-changing efficacy of the presented approaches. Thus, $\rho = 0.66$ and 0.36 are quite suitable to examine the lane-changing control approaches for the upper and lower boundary of traffic density, respectively.

Algorithm 1: Training algorithm of deep Q-learning with request-respond mechanism

```
Initialization: initialize the number of lanes $n$, the number of
  groups $m$, the replay memory $D'$, the action-value function $Q$
  with random weights $\theta'$, the target action-value function $\hat{Q}$ with
  weights $\hat{\theta}'$.
for episode e from 1 to $\mathbb{E}$ do
    Reset a random state $S_{j+1,t}$;
    Get the joint actions of the group with request $A_j$ and
      superimpose the action $C_j$;
    Construct the request message $A_j + C_j$ ;
    for time step t from 1 to $\mathbb{T}$ do
        $\varepsilon = e^{-\beta c}$;
        Choose a random number $r \in [0, 1]$;
        if $r \in [\varepsilon, 1]$ then
            Choose joint action $A_{j+1,t} = (a_{1,j+1}, a_{2,j+1}, ...a_{n,j+1}) =$
              $\arg\max_{A_{j+1,t}} Q(S_{j+1,t}, A_{j,t} + C_{j,t}, A_{j+1,t}; \theta'_t)$ ;
        else
            Choose a random joint action
              $A_{j+1,t} = (a_{1,j+1}, a_{2,j+1}, ...a_{n,j+1})$;
        end
        for each agent $i \in N = \{1, 2, \dots, n\}$ do
            Execute the action $a_{i,j+1}$;
        end
        Observe reward $R'_{j+1,t}$ based on (14) and update the next
          state $S_{j+1,t+1}$;
        Store transition $(S_{j+1,t}, A_{j,t} + C_{j,t}, A_{j+1,t}, R'_{j+1,t}, S_{j+1,t+1})$ in a
          replay buffer $D'$;
        Sample a minibatch of K experience from $D'$;
        if episode m terminates then
            $y_{j+1,t} = R'_{j+1,t}$;
        else
            $y_{j+1,t} =$
              $R'_{j+1,t} + \gamma \max_{A_{j+1,t+1}} \hat{Q}(S_{j+1,t+1}, A_{j,t+1} + C_{j,t+1}, A_{j+1,t+1}; \hat{\theta}'_t)$;
        end
        Update Q-network by minimizing loss function
          $[y_{j+1,t} - Q(S_{j+1,t}, A_{j,t} + C_{j,t}, A_{j+1,t}; \theta'_t)]^2$ based on (15);
          Update the target Q-network $\hat{Q} = Q$ for every $\mathbb{X}$ steps
    end
end
```

Algorithm 2: Execution algorithm of lane-changing system

```
Initialization: initialize the number of lanes $n$, the number of
  groups $m$ and the iterations $\mathbb{K}$;
for Iteration k from 1 to $\mathbb{K}$ do
    for Group j from 1 to m do
        if Request == true then
            Select the joint action
              $A_j = (a_{1,j}, a_{2,j}, ...a_{n,j}) = \arg\max_{A_{j,t}} Q(S_{j,t}, A_{j,t}; \theta_t)$ for the
              request group by using request Q-network;
            Extend the joint action $A_j$ by superimposing a
              communicative action $C_j$ which is based on the
              requirement of current agent;
            Select the joint action $A_{j+1} = (a_{1,j+1}, a_{2,j+1}, ...a_{n,j+1}) =$
              $\arg\max_{A_{j+1,t}} Q(S_{j+1,t}, A_{j,t} + C_{j,t}, A_{j+1,t}; \theta'_t)$ for the respond
              group by using respond Q-network;
            Select a communicative action
              $C_{j+1} = (c_{1,j+1}, c_{2,j+1}, ...c_{n,j+1})$ based on the joint actions
              $A_{j+2}$ of the front group $j + 2$;
            for each agent $i \in N = \{1, 2, \dots, n\}$ do
                Execute the action $a_{i,j+1} + c_{i,j+1}$ for the respond
                  group;
            end
            Respond to the group with a request and confirm the
              request;
            Reselect $C_j = (c_{1,j}, c_{2,j}, ...c_{n,j})$ with the confirmation and
              superimpose into $A_j$;
            for each agent $i \in N = \{1, 2, \dots, n\}$ do
                Execute the action $a_{i,j} + c_{i,j}$ for the request group;
            end
            $j = j + 2$;
        else
            Select the joint action
              $A_j = (a_{1,j}, a_{2,j}, ...a_{n,j}) = \arg\max_{A_{j,t}} Q(S_{j,t}, A_{j,t}; \theta_t)$ by using
              request Q-network;
            Extend the joint action $A_j$ by superimposing a
              communicative action $C_j = (c_{1,j}, c_{2,j}, ...c_{n,j})$ which is
              based on the joint actions $A_{j+1}$ of the front group $j + 1$;
            for each agent $i \in N = \{1, 2, \dots, n\}$ do
                Execute the action $a_{i,j} + c_{i,j}$;
            end
            $j = j + 1$;
        end
    end
end
```

Various states (indicators) are defined to evaluate the performance of the lane-changing approaches. The number of the total incoming vehicles is I_v, and the incoming rate indicating the incoming vehicles per iteration is I_r. The passing vehicles defined as P_v pass through the lane-changing area, and the throughput P_r is defined as the number of passing vehicles per iteration. The lane-changing rate L_r represents how much percentage of vehicles with the lane-changing demand change to the target lane from the original lane. The lane-changing maneuvers must be completed before passing through the lane-changing area. Moreover, the computational time T_c evaluates the computational complexity.

Table 4 lists the hyperparameters and values of the algorithm DQN and DQNRR. Both algorithms share the same hyperparameters except the maximum step size $\mathbb{T}$ and maximum episode size $\mathbb{E}$, i.e., DQNRR have larger values for training due to the larger state space. The network consists of three fully connected hidden layers with 512 connective nodes.

Table 4
List of hyperparameters and values of the training process in both learning algorithms.

Hyperparameters	Value
Learning rate α	0.005
Discount factor γ	0.80
Initial exploration ϵ	1
Final exploration ϵ	0.001
Number of exploration steps	2500000(DQN),3000000(DQNRR)
Maximum step size $\mathbb{T}$	3000000(DQN), 3200000(DQNRR)
Maximum episode size $\mathbb{E}$	400000
Replay buffer size D	50000
Minibatch size	512
Update frequency of target network	2000
Hidden layers	3
Hidden notes	512

4.1.2. Baselines

The baseline methods, including a Rule-based Method (RULE), a Game Theory-based Decomposition Algorithm (GTDA) and a Deep Q-network (DQN) method, are introduced and implemented in this lane-changing system. The performance of these methods, such as the

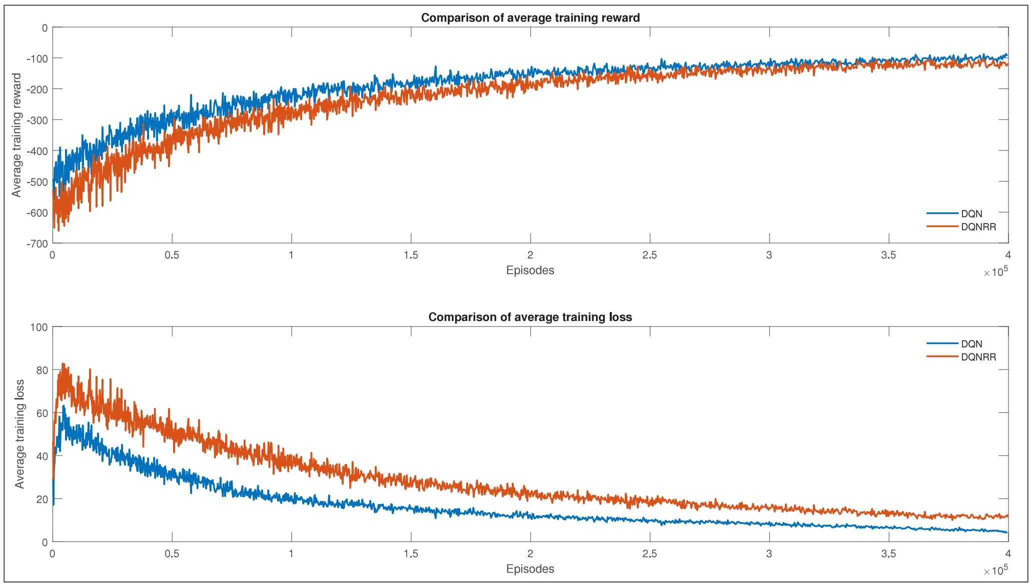

Fig. 4. Comparison of training performance between the methods.

throughput, lane-changing rate and computational time, are compared with the proposed algorithm (DQNRR). The details of the baselines are described as follows:

- **Rule-based method:** RULE generates the lane-changing actions for agents based on the driver's knowledge which the crisp mathematical equations and conventional logic rules can represent. The agent will accelerate or decelerate to create an available gap to satisfy the lane-changing demand of the adjacent agent. The framework of lane-changing was proposed in the early stage (Gipps, 1986). In the previous work (Guo & Harmati, 2022), RULE was adjusted to suit this lane-changing system. Thus, the performance of the same environmental setting can be directly compared with the proposed method. However, there is little difference in the collision and action setting (i.e., the path collision and decelerate action are removed in this paper).
- **Game theory-based decomposition algorithm:** GTDA is proposed by Guo and Harmati (2022) to improve the computational efficiency with a framework of game theory. This method can decompose a non-zero-sum non-cooperative game (i.e., groups) containing multiple players (i.e., agents) into several smaller subgames. The optimal joint actions for the same group can be selected by finding the completely consistent or partially consistent equilibrium. Compared with the classic Nash equilibrium strategy in Guo and Harmati (2022), the efficiency of lane-changing of the decomposition algorithm is almost the same but with much higher computational efficiency. Therefore, this algorithm is selected as the baseline, and a slight difference exists in collision and action settings like RULE.
- **Deep Q-network:** As mentioned in Section 3, DQN method is also considered as a baseline. Each group in the lane-changing system share the same Q-network, and the Q-network is trained based on the local observation (i.e., states) without the connections from other groups. Thus, during the execution process, every group produces joint actions for the agents in the same group by using the trained request Q-network which is also a part of the proposed method.

4.2. *Performance of training process*

There are two different Q-network training for the request and respond groups in this lane-changing system, respectively. As shown in Fig. 4, the average reward and loss of every 200 episodes for both algorithms are illustrated and compared during the training process. The reward referred to (13) and (14) as well as the loss referred to (10) and (15) are calculated in the training process for these two algorithms, respectively. The average training reward and loss of two algorithms tend to converge as the number of episodes increases. The training process terminates if the average training loss remains relatively small for some episodes after a long-term training period (i.e., 4×10^5), considered the state of convergence. The average training reward is the average value of the sum of episode rewards in 200 episodes, and the episode reward is the sum of the reward value of every time step in an episode. The reward of the training group is calculated by summing the rewards of all agents in the group. The curve of the average training reward of DQNRR is quite similar to the reward curve of DQN. Both curves grow as the training process begins and tend to converge. The value of the average training reward of DQNRR is smaller than the reward value of DQN initially since more training steps are required in an episode due to the input of the request message causing more dimensions. However, their difference gets smaller, and the curves are close when the value converges. Undoubtedly, the curves of average training loss reflect an opposite change compared to the curves of the average training reward. The properties and differences of the loss change between these two algorithms are similar to the reward change. Due to the extra reward from the request confirmation, the loss value of DQNRR is a litter higher than the loss value of DQN at the same training episode.

4.3. *Evaluation of control methods*

This lane-changing system implemented six different categories of experiments with different expected traffic density (i.e., $\rho = 0.36, 0.42, 0.48, 0.54, 0.60, 0.66$). The sensitivity analysis and statistical analysis evaluate the performance of baselines and the proposed method, illustrated from Figs. 5 to 8.

4.3.1. *Sensitivity analysis*

In Fig. 5 analyses the indicators' sensitivity as the expected traffic density ρ increases. The change of indicators such as incoming rate I_r, throughput P_r, lane-changing rate L_r and computational time T_c per iteration for all the control methods can be observed. The mean value of the indicators for each experimental group is utilized to analyze the sensitivity. The incoming rate I_r reflects the precision rate of

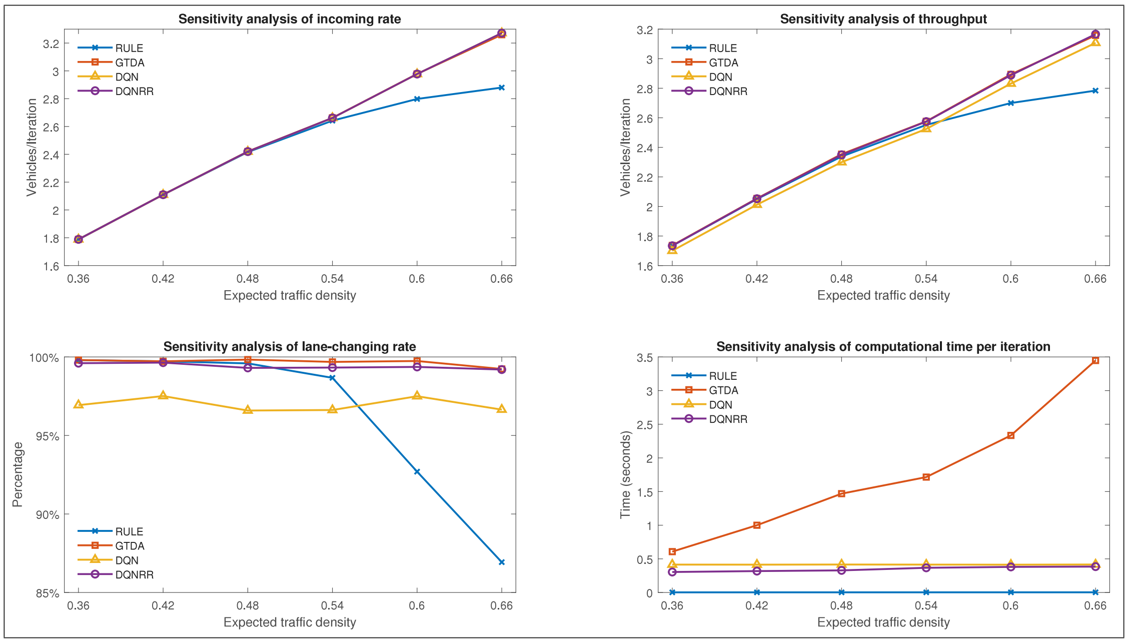

Fig. 5. Sensitivity analysis of incoming rate I_r, throughput P_r, lane-changing rate L_r and computational time per iteration T_c between the methods.

controlling the traffic density ρ (i.e., the incoming rate $I_r \approx E_r = \rho \cdot n$) during the simulation. All the control methods other than RULE comply with the relation between I_r and ρ. When ρ is small that is below 0.54, all methods have the same performance. As ρ grows, the traffic congestion gets heavier in the lane-changing sector, which causes accumulating incoming queues outside the lane-changing sector and fewer incoming vehicles enter the system for RULE.

The throughput P_r indicating the outgoing vehicles per iteration is also an important indicator of the lane-changing system. As mentioned above, the tendency of P_r in Fig. 5 is quite similar to the curves of I_r for all the methods since there exists a positive correlation between P_r and I_r (i.e., $P_v = I_v + \sum_{i=1}^{n} q_i$, where $\sum_{i=1}^{n} q_i$ is the total queue calculated from (5) and is slightly different among the methods). The only difference in the curve of DQN is that P_r is always litter below the curve of GTDA and DQNRR no matter how ρ changes in the specific range. However, the performance is much better than RULE as ρ increases.

The curves of the lane-changing rate L_r of the approaches directly reflect the lane-changing efficiency, representing the rate of vehicles that have the lane-changing demand change to the target lane successfully. As can be seen in Fig. 5, both GTDA and DQNRR have almost the same and best performance in L_r, i.e., nearly 100%. As the expected traffic density ρ grows, they only slightly fluctuate and do not have a noticeable change. When ρ is below 0.54, L_r in RULE approximates the value of that two methods. However, the value of RULE drops dramatically as ρ increases from 0.54 to 0.66. Although L_r in DQN keeps in a specific small range as ρ grows, the value is always smaller than the value of GTDA and DQNRR for the same ρ.

Compared with other methods in Fig. 5, the curve of computational time per iteration T_c in RULE is always below the others since the actions are generated based on a series of logic rules, which indicates that it has the least T_c and barely changes. The T_c of DQN and DQNRR remains a small value smaller than 0.5 s. Specifically, the value in DQN keeps almost a constant, and the value in DQNRR slightly increases until a similar value to the number in DQN as the expected traffic density ρ grows. The method GTDA changes steadily and obviously from 0.61 s to 3.45 s in this range. Moreover, the change rate of T_c increases when ρ increases from 0.54 to 0.66, i.e., more incoming traffic flow enters the lane-changing system, which shows a much more significant difference than the value in other methods.

4.3.2. *Statistical analysis*

The regression analysis as a statistical tool is implemented to evaluate the relationship between the indicators and the traffic density ρ, shown from Figs. 6 to 8. The core parameters estimate the linear or nonlinear models in the statistical analysis. For instance, R-squared indicates how well the observed outcomes are replicated by the model (Steel, 1960), RMSE (i.e., root mean square error) measures the difference between predicted values and observed values (Hyndman & Koehler, 2006) and residuals histograms (i.e., the area of each bar is the relative number of observations, and the sum of the bar areas is equal to 1) show whether the residuals are normally distributed.

The changing trend of incoming rate I_r for all the methods is as same as the tendency of throughput P_r in Fig. 5, so the regression analysis of I_r can refer to the regression analysis of P_r in Fig. 6. As can be seen in Fig. 6, the data model in RULE is quite fitted in a nonlinear function model with the form of logarithmic function (i.e., $y = b_1 ln(x)+ b_2$, where b_1 and b_2 are the coefficients) with R-squared = 0.971 and RMSE = 0.0648, and the residuals is nearly in a normal distribution. For the other three methods, the regression models are quite similar, as well as the residuals analysis. They all reflect the positive correlation between the expected traffic density ρ and the throughput P_r in a linear function model. The value of R-squared is close to 1, and the value of RMSE is relatively small compared with the scale of P_r. The residuals are distributed uniformly for these three methods.

The regression models of lane-changing rate L_r analyze the characteristics further in Fig. 7. Indeed, the control mechanism of these methods is the critical element of this lane-changing system and determines the value of L_r. The regression model of RULE is a logarithmic function model, and the expression of the fit line can be seen from the enlarged sub-figure, i.e., $y = b_1 ln(b_2 x + b_3) + b_4$. The lane-changing rate decreases exponentially as the expected traffic density ρ rises, which complies with the property in Fig. 5. The difference between the raw data enlarges with an increasing ρ. Unlike RULE, the other three methods remain almost the same value with a slight fluctuation as ρ increases, which fits in a constant model (i.e., $y = b1$). The value R-squared = 0 or close to 0 for these approaches also indicate L_r (i.e., y-axis) does not correlate with ρ (i.e., x-axis). However, the value of DQN is smaller than the other two, and the residuals are distributed worse than the other two.

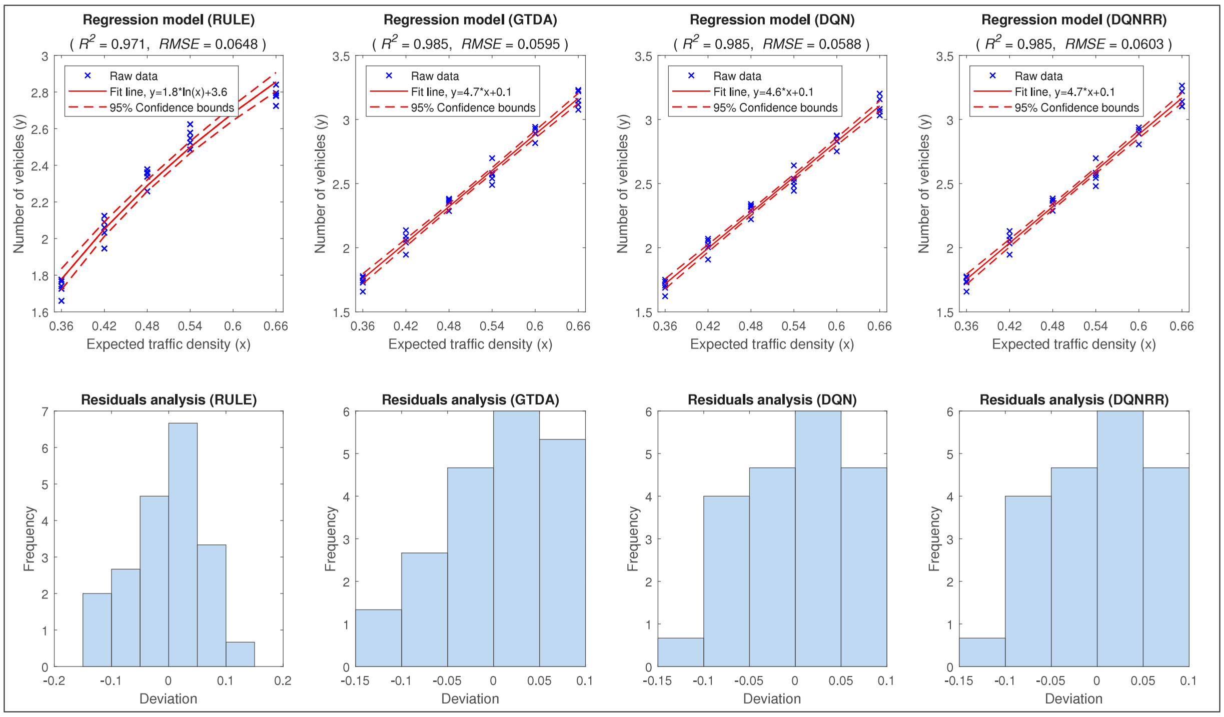

Fig. 6. Regression analysis of throughput P_r for all the methods.

According to the statistical analysis of the computational time per iteration T_c in Fig. 8, RULE fits approximately in a linear function model with R-squared = 0.773, although the computational time slightly changes and remains a small value as the expected traffic density ρ increases. The changing trend of T_c satisfies the exponential function model well (i.e., $y = b_1 x_2^b$, where b_1 and b_2 are the coefficients) in GTDA. T_c exponentially increases as ρ increases since the complexity of the game theory-based system expands exponentially with more incoming vehicles. For the method DQN, T_c is nearly a constant below 0.5 s no matter how ρ changes. In this regression model, R-squared = 0 represents the y-axis value (i.e., computational time) and does not correlate with the x-axis value (i.e., the expected traffic density). The last regression model in DQNRR is similar to the model in RULE, i.e., a linear function model. Although the growth range for T_c is relatively small, an upward tendency still exists.

5. Discussion

In this section, some reasons behind the results above are discussed. Fig. 4 compares the difference of averaged training rewards and loss between DQN and DQNRR in the training process. As shown in Fig. 2, the respond group in DQNRR receives request message from the request group to form a new state, which results in different reward functions for the approach DQN and DQNRR, i.e., (12) and (14) respectively. That explains why the average training reward of DQNRR is smaller than the reward value of DQN initially and close to each other after a long-term training process. The request message expands more dimensions of state space in DQNRR, which means more training steps are required in an episode and lower training reward value at the initial training stage. Increasing training episodes minimizes the difference in average training reward gradually. Similar to the changing trend of average training reward, average training loss shows a slight difference between DQN and DQNRR due to (11) and (15). The training loss value of DQNRR is a litter higher than the one of DQN at the same training episode since DQNRR receives an extra reward from the request confirmation, and the loss value is related to the scale of the reward function.

Fig. 5 suggests that the incoming rate I_r of all the control approaches are approximately the same when the traffic situation is not so congested (i.e., below 0.54 in this case), and RULE increases slower than the others when ρ is above 0.54. The changing trend of incoming rate I_r of DQN and the proposed DQNRR approach comply with the relation with the increasing expected traffic density ρ (i.e., from 0.36 to 0.66). Having such a standard curve for them is reasonable since the decelerate action is removed in this lane-changing system, which results in no accumulated incoming queues waiting outside the lane-changing system. Thus, if the traffic density ρ increases larger than 0.66, they would have no significant effects. GTGA can still regulate the traffic flow under congested traffic, i.e., $\rho \leq 0.66$, so it has few incoming queues, and almost the incoming vehicles enter the system. However, if the traffic density ρ increases, it cannot be guaranteed that GTGA can still regulate the traffic flow efficiently. Due to the composition of a series of sequential rules, RULE cannot deal with the traffic flow in a complicated situation, especially when the traffic congestion gets severe (i.e., $\rho \geq 0.54$). That results in the accumulation of waiting vehicles outside the lane-changing system so that fewer vehicles can enter the lane-changing system. This situation gets much worse as ρ gets more significant. Therefore, I_r of the former three methods is fitted in a linear function model referring to the models in Fig. 6, and a logarithmic function reflects the phenomenon of RULE referring to the models in Fig. 6.

As mentioned in Section 4.3.1, the throughput P_r mainly depends on the incoming rate I_r since more incoming vehicles provide the lane-changing system opportunities to regulate more passing vehicles. Thus, the characteristics of the curves of P_r in Fig. 6 is quite similar to the curves of I_r, also in regression analysis. The only difference between them is that the curve of DQN is always below the curves of DQNRR. That does not mean DQN is less effective than DQNRR in terms of the indicator-throughput P_r. The main reason is that the request group in DQNRR superimposes the communicative action to send a request message and accelerates if the request is confirmed by the respond group, which raises the speed of vehicles in the lane-changing system to a certain extent. Therefore, there are more passing vehicles for DQNRR in the same period, no matter how the traffic density ρ changes.

The lane-changing rate L_r value is critical in evaluating the lane-changing efficiency. The value of L_r for the methods except DQN keeps

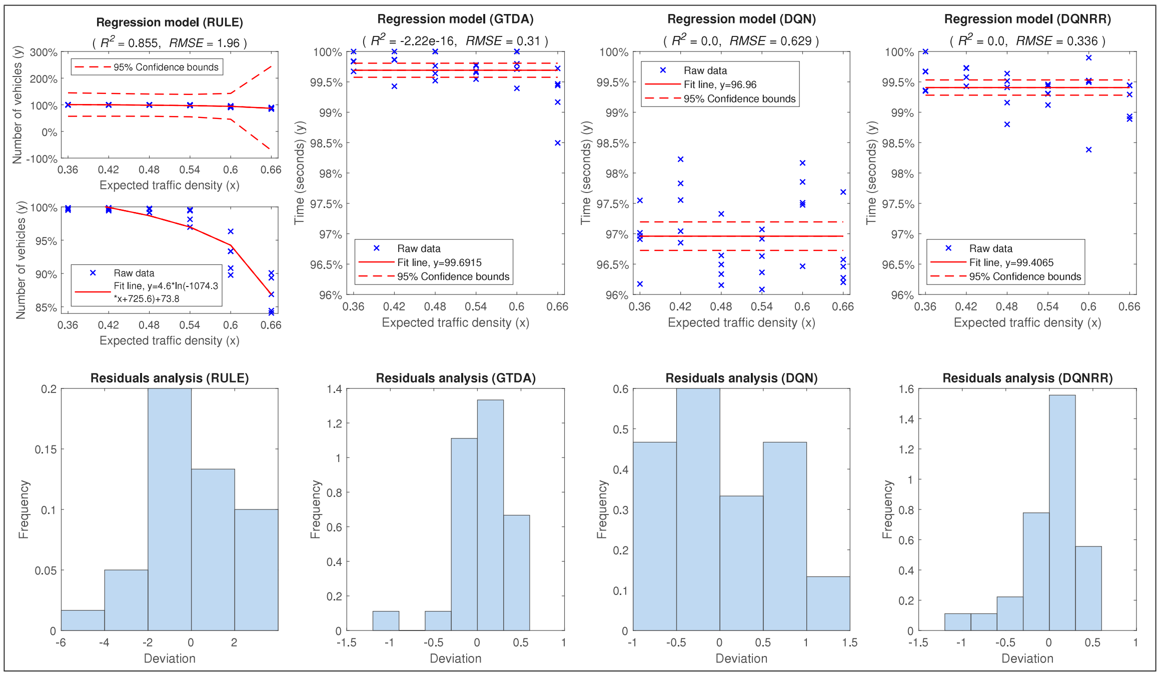

Fig. 7. Regression analysis of lane-changing rate L_r for all the methods.

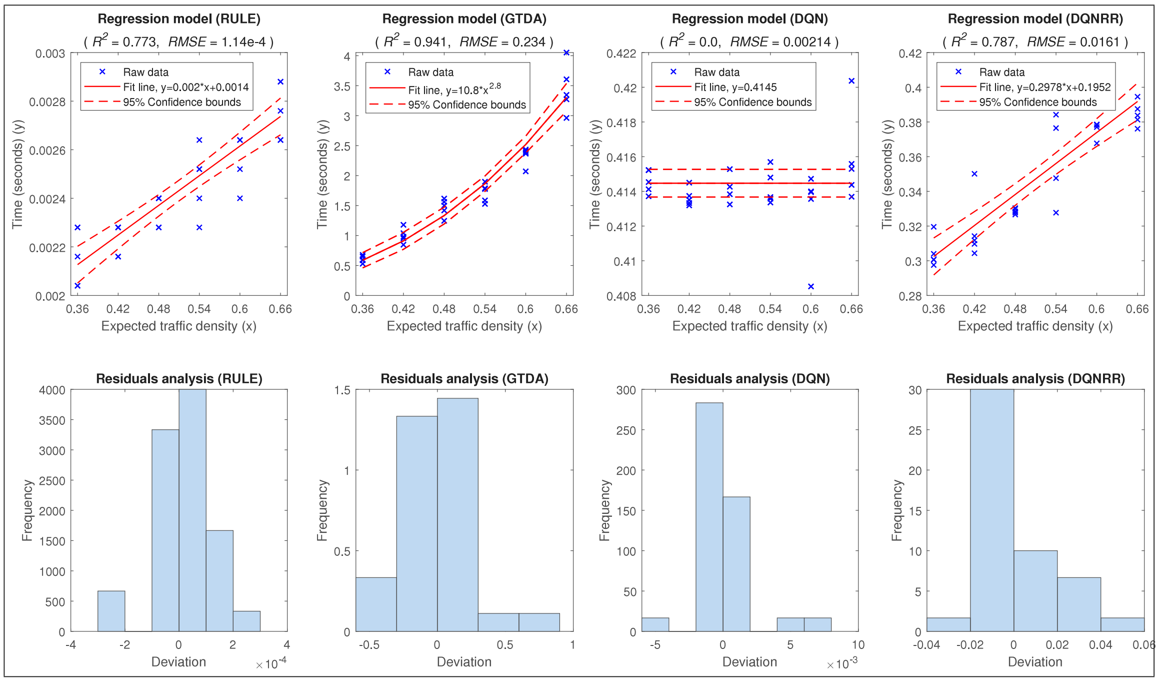

Fig. 8. Regression analysis of computational time per iteration T_c for all the methods.

consistency when ρ is small (i.e., $\rho \leq 0.54$), shown in Fig. 5. Due to a lack of communication mechanism between other groups in DQN, the value of L_r in DQN is always smaller than the value in DQNRR. As mentioned previously, RULE method is capable of regulating the traffic flow efficiently when the ρ is less than 0.54, so the value of L_r in RULE keeps a high value when the traffic is not congested. However, L_r in RULE reduced dramatically as ρ increases, especially when ρ is above 0.54. The main reason is that RULE has the worst capability to control the traffic flow in congested traffic among these control approaches, resulting in a serious situation where many vehicles with lane-changing demand cannot arrive at the target lane before passing the intersection. Thus, L_r decreases exponentially when ρ is above 0.54, shown in Fig. 7. Unlike RULE, the other three control approaches seem to have larger control limits for regulating traffic flow, so the value of L_r keeps quite stable when ρ changes, shown in Figs. 5 and 7. As ρ increases from 0.36 to 0.66, both DQNRR and GTDA have almost the same and best performance, followed by DQN with a smaller number due to the reason explained previously. DQNRR and GTDA almost satisfy all the vehicles with the lane-changing demand (i.e., L_r approximates 100%), which indicates high lane-changing efficiency. Some incoming vehicles just entered the lane-changing system after the experiment finished and did not have the chance to change lanes in the lane-changing area.

That slightly reduces the value L_r and results in a value close to 100% instead of 100%.

Computational time T_c is also an essential indicator of evaluating the computational complexity of the control methods during the execution process. Undoubtedly, the method GTDA spends much more time generating the optimal actions for agents than other methods since calculating Nash equilibrium is complicated with many game players. It is known that the complexity of calculating Nash equilibrium increases exponentially as the number of game players (i.e., incoming vehicles) increases. Therefore, the difference of computational time T_c between GTDA and other methods enlarges as ρ increases, i.e., more incoming vehicles exist in the lane-changing system. The regression model for GTDA in terms of T_c in Fig. 8 is exactly an exponential function model with R-squared = 0.941. As for RULE, T_c barely increases even when ρ grows to a large number since the controller comprises a series of sequential logic rules, which means RULE can produce the optimal actions for agents with a small computational complexity. According to the algorithm in Guo and Harmati (2022), the complexity is also determined by the number of vehicles. As ρ increases, the number of vehicles also increases, which means more rules must be evaluated in the system. Hence, T_c increases slightly and linearly with a linear function model in Fig. 8. Unlike the regression models of GTDA and RULE, DQN shows T_c is nearly a constant and does not correlate with ρ. After investigating the execution process, the method DQN generates the joint actions for each group only by the request Q-network. Each group is executed equally without any judgement rules. The value of T_c is quite small for selecting the actions corresponding to the maximum Q-value according to the Q-network. T_c in DQNRR is close to that in DQN by using Q-network instead of calculating Nash equilibrium. However, the regression model differs from the model in DQN, i.e., a linear function model. Based on Algorithm 1, some additional rules and operations exist for handling the request–respond mechanism. Thus, like RULE, it is reasonable that T_c linearly increases with more incoming vehicles entering the system as ρ increases.

6. Conclusions

Overall, these results above suggest that the approaches GTDA, DQN and DQNRR reveal more advantages in lane-changing efficiency when the expected traffic density increases (i.e., traffic congestion gets severe), compared with RULE. However, if the traffic density is small, RULE is still suitable for the lane-changing system. That reflects it can regulate the traffic flow efficiently in uncongested traffic and has the least computational time due to the composition of a series of sequential rules. The proposed DQNRR proves more lane-changing efficacy than DQN in this lane-changing model. The improvement is revealed in several aspects, such as throughput and the lane-changing rate, which means the request–respond mechanism contributes to a practical framework based on a deep Q-network. Moreover, DQNRR and GTDA have almost the same performance, which is the best among methods concerning lane-changing rate, while DQNRR spends much less time than GTDA in the execution process. The difference in computational time between DQNRR and GTDA would be more significant as the traffic gets congested. Therefore, the proposed method DQNRR has the most potential to ease traffic congestion with high efficacy in lane-changing and short computational time under heavy traffic conditions.

Not only can the proposed method DQNRR be implemented efficiently in a lane-changing system, but also it can improve the efficiency in other multi-agent systems where the agents are under the hierarchical framework. Specifically, the agents can be divided into two groups, i.e., request group and respond group, and the communicative message between groups can be passed to improve efficiency. For instance, it can be used in the industrial field, i.e., a smart grid where the household can efficiently distribute extra electricity to adjacent households that require more energy. The households closer to the power station can be donated with higher priority, and the request–respond mechanism can be introduced between this household and other adjacent households to improve efficiency. In mobile robotics, the robots can cooperate to coordinate or finish tasks to improve working efficiency. Generally, the levels of robots are defined by the ranks of the tasks handled by the robots, the robots that can handle complicated tasks are at a high level, and the robots are at a low level with simple tasks. Thus, the proposed approach can also be used in such a robotic system. Similarly, in some teamwork games such as basketball and football, the game players can be regarded as agents and cooperate efficiently with the optimal strategies generated by the approach.

Despite the significant benefits of the strategies, some limitations mentioned above still exist in this proposed model. For instance, some assumptions are still ideal, such as the discrete space of traffic lanes, the discrete velocity of vehicles, and the smooth crossing maneuvers. For the proposed approach, it cannot take much effect on some scenarios if the level of agents is disorganized since the request–respond mechanism is created based on the hierarchical framework.

In the future, the continuous velocity instead of the discrete velocity of vehicles should be evaluated to make the lane-changing system more realistic. Furthermore, it deserves to be studied to extend the request–respond mechanism in a more complicated environment where agents or groups simultaneously have more connections with other agents or groups.

CRediT authorship contribution statement

Jian Guo: Conceptualization, Methodology, Software, Validation, Writing – original draft. **Istvan Harmati:** Conceptualization, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The research reported in this paper is part of project no. BME-NVA-02, implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021 funding scheme. This paper was also supported by Stipendium Hungaricum Scholarship of Tempus Public Foundation and China Scholarship Council (201802890009) (Guo Jian).

References

An, H., & Jung, J.-i. (2019). Decision-making system for lane change using deep reinforcement learning in connected and automated driving. *Electronics*, *8*(5), 543.
Arai, K., & Sentinuwo, S. R. (2012). Spontaneous-braking and lane-changing effect on traffic congestion using cellular automata model applied to the two-lane traffic. *IJACSA International Journal of Advanced Computer Science and Applications*, *3*(8).
Chen, C., Qian, J., Yao, H., Luo, J., Zhang, H., & Liu, W. (2018). Towards comprehensive maneuver decisions for lane change using reinforcement learning.
Das, J. N., Tiwari, M. K., Sinha, A. K., & Khanzode, V. (2023). Integrated warehouse assignment and carton configuration optimization using deep clustering-based evolutionary algorithms. *Expert Systems with Applications*, *212*, Article 118680.
Daskalakis, C., Goldberg, P. W., & Papadimitriou, C. H. (2009). The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, *39*(1), 195–259.
Dong, J., Chen, S., Li, Y., Du, R., Steinfeld, A., & Labi, S. (2021). Space-weighted information fusion using deep reinforcement learning: The context of tactical control of lane-changing autonomous vehicles and connectivity range assessment. *Transportation Research Part C (Emerging Technologies)*, *128*, Article 103192.

Gipps, P. G. (1986). A model for the structure of lane-changing decisions. *Transportation Research, Part B (Methodological)*, *20*(5), 403–414.

Guo, J., & Harmati, I. (2020). Evaluating semi-cooperative Nash/stackelberg Q-learning for traffic routes plan in a single intersection. *Control Engineering Practice*, *102*, Article 104525.

Guo, J., & Harmati, I. (2022). Lane-changing decision modelling in congested traffic with a game theory-based decomposition algorithm. *Engineering Applications of Artificial Intelligence*, *107*, Article 104530.

Halati, A., Lieu, H., & Walker, S. (1997). CORSIM-corridor traffic simulation model. In *Traffic congestion and traffic safety in the 21st century: Challenges, innovations, and OpportunitiesUrban transportation division, ASCE; highway division, ASCE; Federal Highway Administration, USDOT; and National Highway Traffic Safety Administration, USDOT*.

Hidas, P. (2005). Modelling vehicle interactions in microscopic simulation of merging and weaving. *Transportation Research Part C (Emerging Technologies)*, *13*(1), 37–62.

Hoel, C.-J., Driggs-Campbell, K., Wolff, K., Laine, L., & Kochenderfer, M. J. (2019). Combining planning and deep reinforcement learning in tactical decision making for autonomous driving. *IEEE transactions on intelligent vehicles*, *5*(2), 294–305.

Hou, Y., & Graf, P. (2021). Decentralized cooperative lane changing at Freeway Weaving Areas using multi-agent deep reinforcement learning. arXiv preprint arXiv:2110.08124.

Huang, D.-w. (2002). Lane-changing behavior on highways. *Physical Review E*, *66*(2), Article 026124.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679–688.

Jiang, S., Chen, J., & Shen, M. (2019). An interactive lane change decision making model with deep reinforcement learning. In *2019 7th international conference on control, mechatronics and automation* (pp. 370–376). IEEE.

Kesting, A., Treiber, M., & Helbing, D. (2007). General lane-changing model MOBIL for car-following models. *Transportation Research Record*, *1999*(1), 86–94.

Kita, H. (1999). A merging–giveway interaction model of cars in a merging section: a game theoretic analysis. *Transportation Research Part A: Policy and Practice*, *33*(3–4), 305–312.

Li, H., Luo, Y., & Wu, J. (2019). Collision-free path planning for intelligent vehicles based on Bézier curve. *IEEE Access*, *7*, 123334–123340.

Liu, C., Lee, S., Varnhagen, S., & Tseng, H. E. (2017). Path planning for autonomous vehicles using model predictive control. In *2017 IEEE intelligent vehicles symposium* (pp. 174–179). IEEE.

Liu, H. X., Xin, W., Adam, Z., & Ban, J. (2007). A game theoretical approach for modelling merging and yielding behaviour at freeway on-ramp sections. *Transportation and Traffic Theory*, *3*, 197–211.

Makantasis, K., Kontorinaki, M., & Nikolos, I. (2019). A deep reinforcement learning driving policy for autonomous road vehicles. arXiv preprint arXiv:1905.09046.

Mukadam, M., Cosgun, A., Nakhaei, A., & Fujimura, K. (2017). Tactical decision making for lane changing with deep reinforcement learning.

Pan, F., Zhang, L., Wang, J., Ma, C., Yang, J., & Qi, J. (2020). Lane-changing risk analysis in undersea tunnels based on fuzzy inference. *IEEE Access*, *8*, 19512–19520.

Preitl, Z., Precup, R.-E., Tar, J. K., & Takács, M. (2006). Use of multi-parametric quadratic programming in fuzzy control systems. *Acta Polytechnica Hungarica*, *3*(3), 29–43.

Rigatos, G., Siano, P., Selisteanu, D., & Precup, R. (2017). Nonlinear optimal control of oxygen and carbon dioxide levels in blood. *Intelligent Industrial Systems*, *3*, 61–75.

Ross, T. J. (2005). *Fuzzy logic with engineering applications*. John Wiley & Sons.

Sen, B., Smith, J. D., Najm, W. G., et al. (2003). *Analysis of lane change crashes*: *Technical report*, United States. National Highway Traffic Safety Administration.

Shalev-Shwartz, S., Shammah, S., & Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295.

Sharifzadeh, S., Chiotellis, I., Triebel, R., & Cremers, D. (2016). Learning to drive using inverse reinforcement learning and deep q-networks. arXiv preprint arXiv:1612.03653.

Steel, R. G. (1960). *Principles and procedures of statistics: with special reference to the biological sciences*: *Technical report*.

Steger-Vonmetz, D. (2005). Improving modal choice and transport efficiency with the virtual ridesharing agency. In *Proceedings. 2005 IEEE intelligent transportation systems* (pp. 994–999). Ieee.

Subramani, P., Sattar, K. N. A., de Prado, R. P., Girirajan, B., & Wozniak, M. (2021). Multi-classifier feature fusion-based road detection for connected autonomous vehicles. *Applied Sciences*, *11*(17), 7984.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning* (pp. 330–337).

Vechione, M., Balal, E., & Cheu, R. L. (2018). Comparisons of mandatory and discretionary lane changing behavior on freeways. *International Journal of Transportation Science and Technology*, *7*(2), 124–136.

Wall, G., & Hounsell, N. (2005). Microscopic modelling of motorway diverges. *European Journal of Transport and Infrastructure Research*, *5*(3).

Wang, H., Huang, Y., Khajepour, A., Zhang, Y., Rasekhipour, Y., & Cao, D. (2019). Crash mitigation in motion planning for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *20*(9), 3313–3323.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3–4), 279–292.

Woźniak, M., Zielonka, A., & Sikora, A. (2022). Driving support by type-2 fuzzy logic control model. *Expert Systems with Applications*, *207*, Article 117798.

Ye, F., Cheng, X., Wang, P., Chan, C.-Y., & Zhang, J. (2020). Automated lane change strategy using proximal policy optimization-based deep reinforcement learning. In *2020 IEEE intelligent vehicles symposium* (pp. 1746–1752). IEEE.

Yu, H., Tseng, H. E., & Langari, R. (2018). A human-like game theory-based controller for automatic lane changing. *Transportation Research Part C (Emerging Technologies)*, *88*, 140–158.

Zheng, Z. (2014). Recent developments and research needs in modeling lane changing. *Transportation Research Part B: Methodological*, *60*, 16–32.