IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 6, 2019, accepted October 3, 2019, date of publication October 11, 2019, date of current version October 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946659

Improved Multi-Agent Deep Deterministic Policy Gradient for Path Planning-Based Crowd Simulation

SHANGFEI ZHENG AND HONG LIU
School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Jinan 250014, China

Corresponding author: Hong Liu (lhsdcn@126.com)

This work was supported by the National Natural Science Foundation of China under Grant 61876102, Grant 61472232, and Grant 61272094.

ABSTRACT Deep reinforcement learning (DRL) has been proved to be more suitable than reinforcement learning for path planning in large-scale scenarios. In order to more effectively complete the DRL-based collaborative path planning in crowd evacuation, it is necessary to consider the space expansion problem brought by the increase of the number of agents. In addition, it is often faced with complicated circumstances, such as exit selection and congestion in crowd evacuation. However, few existing works have integrated these two aspects jointly. To solve this problem, we propose a planning approach for crowd evacuation based on the improved DRL algorithm, which will improve evacuation efficiency for large-scale crowd path planning. First, we propose a framework of congestion detection-based multi-agent reinforcement learning, the framework divides the crowd into leaders and followers and simulates leaders with a multi-agent system, it considers the congestion detection area is set up to evaluate the degree of congestion at each exit. Next, under the specification of this framework, we propose the improved Multi-Agent Deep Deterministic Policy Gradient (IMADDPG) algorithm, which adds the mean field network to maximize the returns of other agents, enables all agents to maximize the performance of a collaborative planning task in our training period. Then, we implement the hierarchical path planning method, which upper layer is based on the IMADDPG algorithm to solve the global path, and lower layer uses the reciprocal velocity obstacles method to avoid collisions in crowds. Finally, we simulate the proposed method with the crowd simulation system. The experimental results show the effectiveness of our method.

INDEX TERMS Deep reinforcement learning, multi-agent reinforcement learning, path planning, crowd simulation for evacuation, improved multi-agent deep deterministic policy gradient algorithm.

I. INTRODUCTION

Reasonably planning an evacuation path to reduce the evacuation time in densely populated areas, especially complex environments with obstacles and multiple exits, is one of the important issues of evacuation simulations caused by emergency disasters. In such a situation, most pedestrians will run to the nearest exit or follow the crowd to the exit [1], which will cause congestion delay, trampling fatalities and other security incidents [2]. To further obtain the experimental data in the evacuation process, it is necessary for a large number of people to conduct real-world experiments on crowd evacuation. However, these experiments have many limitations that prevent further application, such as the complexity of organization and the dangerousness of uncontrollable factors [3]. Computer simulation technology not only guarantees safety and convenience in reality, but also provides the meaningful information for crowd evacuation [4]. Therefore, it is indispensable to apply a computer simulation method to study crowd evacuation, for example, large public places in emergencies.

Considering the problem of the large-scale planning of an evacuation path, multi-agent reinforcement learning (MARL) has widespread attention [5], [6], wherein the agents interact with the environment and obtain the reward and punishment information transmitted from the environment to maximize the reward through trial and error, which is closely related to the path knowledge accumulation of pedestrians in reality.

The associate editor coordinating the review of this manuscript and approving it for publication was Chaitanya U. Kshirsagar.

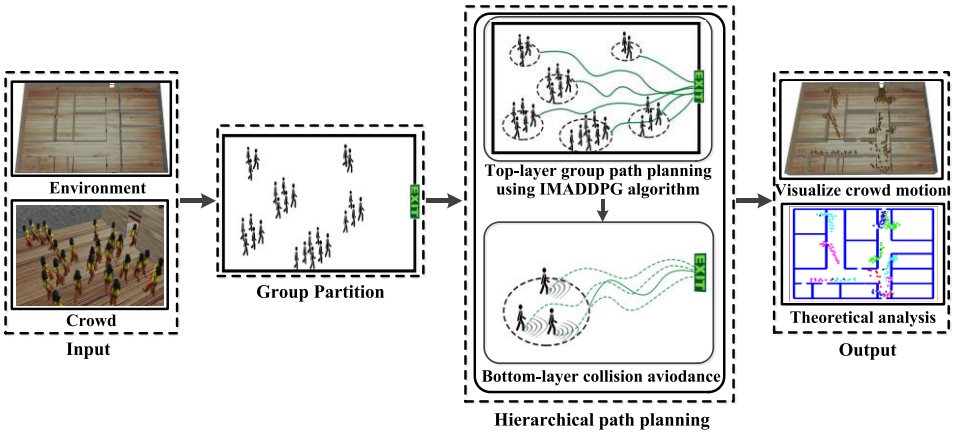


FIGURE 1. The framework of our method.

These characteristics naturally relate to the path planning problem [7]–[10]. However, when MARL is applied to crowd evacuation simulation, a relatively large-scale state space is formed, and the size of the state space can increase exponentially according to the number of agents involved. These problems of MARL have promoted the development of DRL. Multi-agent DRL inherits the advantages of reinforcement learning (RL) and can better address the curse of dimensionality of RL with the characteristics of a strong nonlinear approximation ability and a generalization ability of neural networks [11], [12].

Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm is a new population DRL algorithm, which is proposed by Lowe et al. [13]. It can find the global optimization solution and can easily defeat various DRL methods, including DQN (Deep Q-Network), TRPO (Trust region policy optimization) and DDPG (Deep Deterministic Policy Gradient). However, few researchers have applied the MADDPG algorithm to crowd evacuation simulation because the input space of Q grows linearly as the number of people increases. In addition, complex circumstances (e.g., exit selection and congestion) are another reason why the MADDPG is not suitable for crowd evacuation simulation. All above the factors should be considered, which may lead to the critical decline of the evacuation efficiency and increase of the evacuation time.

To address the problems mentioned above, this paper proposes a planning approach for crowd evacuation based on DRL to search for the evacuation path. Our method is mainly implemented in four steps, as shown in Fig. 1. In the first step, the crowd and the environment are modelled to prepare for the next phase of work. The second step demonstrates the grouping and leadership selection of the initialized groups, and the followers in the same group evacuate according to the path of the leader. The third step shows a hierarchical path planning method, whose process is divided into two layers: the upper layer is multi-agent collaborative path planning based on the IMADDPG algorithm, which enables the leader agent to maximize the performance of a collaborative path planning, and the lower layer is the movement of collision avoidance based on the reciprocal velocity obstacles (RVO) method [14]. The fourth step is to output and analyse the evacuation results on our simulation platform. The main contributions of our paper are listed below.

(1) The framework of congestion detection-based MARL is proposed by considering the congestion detection area of the reward function, which is beneficial to realize reasonable congestion avoidance and the selection of an evacuation exit.
(2) A new deep reinforcement learning algorithm called IMADDPG is proposed, IMADDPG adds the mean field network to maximize the returns of other agents, and uses the mean field idea to reduce the complexity of training additional samples, it can enable all agents to work better and maximize performance of a collaborative planning task.
(3) A hierarchical path planning method applied in the field of crowd evacuation is proposed to reduce evacuation time, which couples the IMADDPG and the RVO algorithms under the framework of congestion detection-based MARL.

The rest of this paper is organized as follows. Section II describes the related work on MARL, DRL and the path planning approach. Section III describes the theoretical background about our algorithm. Section IV describes the framework of congestion detection-based MARL. Section V proposes the hierarchical path planning methods. Section VI performs simulation experiments to show the efficiency of the approach proposed in this paper. Section VII describes the conclusion of this study and suggests future research.

II. RELATED WORK

A. MULTI-AGENT REINFORCEMENT LEARNING

A MAS comprises independent, autonomous and interactive agents. Because of a limited perception of the entire environment of each agent in the system, agents need to cooperate with one another, follow specific social rules, and make joint

decisions that aim to simulate the cognitive process of human beings. MAS captures the dynamics of natural and human activities in a cooperative way, which makes it a suitable solution to solve the problem of crowd evacuation [15]–[17].

The clarity of human behaviour makes the idea of multi-agent beliefs, desires and intentions (BDI) increasingly more accepted. An agent is driven by desire (goal) and guided by own beliefs through the intention to achieve the goal [2]. For example, when an emergency evacuation is required, the desire of the crowd is to escape from the dangerous area and advance to the safe area. The belief is the knowledge of an optimal path to the current stage of the safe area, and the intention is the necessary action taken by the crowd. In a multi-agent system, although the basic behaviour of a multi-agent can be presupposed, it is difficult to presuppose good behaviour when the environment is too complex. Many tasks require agents to learn new behaviours online for the improved performance of a multi-agent system [18].

RL is a type of algorithm in the field of machine learning that aims to allow agents to learn how to maximize future rewards in the environment. Specifically, an agent changes his state through actions and obtains a reward signal in the form of a scala [19]. RL can be regularized by the Markov decision process framework $< S, A, T, R >$, including state S, action A, the transformation function T, the reward equation R, etc. For different spatial problems, the Markov decision-making process is divided into the continuous domain and the discrete domain Markov decision-making process [20]. RL produces better behaviour by reinforcing higher rewards, and it has attracted close attention in the fields of game making, robot planning, computer simulations, etc.

MARL combines the advantages of multi-agent technology and RL, overcomes the mechanical mode of a multi-agent system that needs pre-set behaviour, and enhances the efficiency and scalability of a multi-agent system by strengthening the learning mechanisms. All agents maximize the common return. However, Busoniu *et al.* clearly pointed out that MARL still faces some challenges, specifically the problem of curse of dimensionality and the instability of the environment becoming more serious [18].

B. DEEP REINFORCEMENT LEARNING

Mnih *et al*. [21] indicated that traditional RL is limited to the scale of space, and tasks close to the actual situation often have a large state space and continuous action space. However, deep learning (DL) can better handle high dimensional problems. DRL combines the high dimensional perception of DL and the decision-making ability of RL by using neural networks to more effectively solve the dimensional problems that perplex RL. DL provides a new perspective to solve the problem of cognitive decisions in the complex environments.

DRL is mainly divided into two categories. The first category is value-based DRL, which learns the value model by monitoring the sequence of information and updating the policy with the value model. Mnih *et al.* proposed the DQN in [22]. To some extent, it is the pioneering work of DRL. Sample pools and a fixed target network are used in DQN. Hasselt *et al.* [23] introduced Double Deep Q-Network (DDQN), compared with DQN, DDQN can obtain not only a more accurate value estimation but also a higher score in Atari 2600. Wang *et al.* introduced Dueling DQN, which divides the Q network into two parts: the value function related only to the state and the superiority function related to the action of the state, which results in a more accurate Q value [24].

The second type of DRL based on a policy gradient is more concerned, since it updates the parameters by constantly calculating the gradient of the policy expectation total reward for the policy parameters, and then finally, it converges to the optimal policy. In [25], the DDPG algorithm proposed by Lillicrap is a typical deep policy gradient algorithm based on Actor-Critic [26] architecture. It draws on the idea of empirical playback and the separation of the target network from DQN, and the discrete behaviour space is successfully extended into the continuous space. However, almost all of the current reinforcement learning methods mentioned above are not applicable to MAS. MADDPG proposed by Lowe *et al.* extends DDPG into a multi-agent system to make all agents cooperate globally [13]. In MADDPG, agents adopt the principle of centralized training and decentralized execution. That is, during training, a centralized critic module provides agents with information about the observation and potential behaviour of all agents, and this interaction among agents converts an unpredictable environment into a predictable environment. When testing, the agent can not see all the information, and he does not know the policy of the other agents; he uses only the information suggested by his own critic module to take action alone. Therefore, compared with other DRL methods, MADDPG is more suitable for collaborative crowd path planning. The schematic figure of the MADDPG algorithm is shown in Fig. 2. MADDPG includes the network module of both actor and critic. Each actor corresponds to an agent in MAS, and the input observation of the critic network contains local observations and operations from all agents, which eliminates the non-static nature of the environment and enables the agents to collaboratively learn the path knowledge.

C. PATH PLANNING APPROACH BASED ON RL

Path planning is one of the critical issues of evacuation simulations, which aim to find the optimal path from the starting position to the target position. At present, the more commonly used path planning methods in the crowd simulation field include swarm intelligence algorithms, the methods based on RL, etc. The swarm intelligence algorithm is popular in crowd path planning, but these algorithms often face premature and stagnant phenomena. For example, the Ant Colony Optimization algorithm (ACO) [27], Particle Swarm Optimization algorithm (PSO) [28], and Artificial Bee Colony algorithm (ABC) [29] do not fundamentally solve the problem of premature algorithms to some extent, which limits their further application [30].

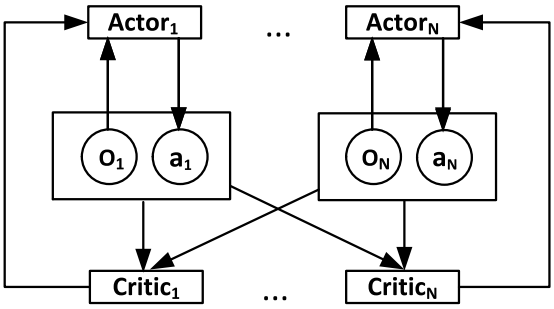


FIGURE 2. Schematic of the MADDPG algorithm.

RL based on single agent has been widely used in most previous cases of path planning, such as Q-learning [31] and Sarsa [32]. The rewards are usually set to find exits and avoid obstacles. However, MARL has been developed more in the field of planning an evacuation path. In [7]–[9], a small case of MARL and tile coding [33] applied to crowd path optimization is provided. The method of tile coding discretization is used to simplify the calculation to some extent, which inevitably leads to lower precision, reduces the maximum possible reward in path planning, and leads to a convergence to the local optimum when the input changes are small. Lee [10] proposed a method based on MARL for predicting crowd congestion to achieve crowd path planning in the event of a disaster. The temporary target on the path is determined by the grid as each step unit, and the human actions in the simulation scene are simply separated into five actions. Then, according to the extracted direction, the current state and adjacent feature information are driven to predict the degree of congestion, and the path direction with a small crowd density is selected for evacuation.

The above methods provide a new idea for the research on crowd path planning based on RL to some extent. However, because of the curse of dimensionality, the continuous behaviour and observation space of the crowd are discretized, and the expansion of the observation and action space and the increase of the number of agents are still the focus of the research work of the above methods in the future. In addition, with the development of the field of DRL, many studies have been devoted to applying DRL to the path planning of complex scenes as we are doing here.

In his thesis [34], Panov *et al.* applied the method of DRL for path planning on square grids. The experiments show that heuristic algorithms such as A^* in the same scene is not suitable for the path planning of complex scenes with multi-target points, but robust learning can be realized in similar scenarios by using the DRL method. Although the DRL method can not be regarded as the optimal method for path finding, it provides a perspective for applying it to path planning.

Cruz and Yu [35] proposed a method that combines kernel smoothing and the DRL of the WoLF- Policy Hill Climbing algorithm to solve the difficulty of traditional reinforcement learning in path planning in an unfamiliar environment. Without prior knowledge, the discrete action space was used to approximate the state of MARL by kernel smoothing, thereby reducing the state space in the Q table. However, when the number of agents increases, the efficiency of the path planning method will be affected.

Referring to [36], Lei *et al.* applied Double DQN, to the path planning of robots in a high-dimensional space, designed a new reward function and expanded the state space of the sample pool by inputting lattice information and local target point coordinates to explore how to solve the problem of path planning.

In [37], Sui *et al.* proposed a method similar to ours in which they set the reward function and used the DRL algorithm to implement the path planning for pedestrians. Their approach differs from our approach in the following main aspects. (1) The scenarios that we address are different from their dynamic random obstacles and are modelled around real buildings, and more agents in complex scenarios are considered. (2) The input between our agents includes state-action information about other agents, which will promote better collaborative planning among multi-agents. (3) We designed a suitable reward function to achieve crowd avoidance and exit choice during the evacuation process, and closer to the actual evacuation situation, the crowd chooses the exit with lower congestion rather than heavily congested exits.

III. THEORETICAL BACKGROUND

In this section, we will introduce the concepts and terms in our algorithms concerning MARL, RVO, MADDPG, mean field theory, RL such as Q-Learning.

A. MARKOV DECISION PROCESS

is a theoretical model of sequential decision-making, which is used to simulate the realizable policies and returns of agents in an environment with Markov property of system state [20]. It is constructed based on a set of interactive objects, namely agents and environments. Its elements include state, action, policy and reward. Specifically, the agent perceives the state of the environment and acts on the environment according to the policy, thus changing the state of the environment and getting rewards. The accumulation of rewards over time is called return.

B. POLICIES

are classified into deterministic policy and stochastic policy, and deterministic policy are relative to stochastic policy. The stochastic policy is in the same state, and the action adopted is based on a probability distribution, that is, uncertain. The deterministic policy decides the simple point. Although the action probability is different at the same state, the maximum probability is only one. If we only take the action with the maximum probability, it will be much simpler. That is to say, as a deterministic policy, action is uniquely deterministic in the same state.

C. REINFORCEMENT LEARNING

regards learning as a process of exploratory evaluation [19]. The agent chooses an action a to be used in the environment,

after the environment accepts the action a, we will find that the state S_t of the environment has changed to S_{t+1}. At the same time, a reinforcement signal called reward r is generated and fed back to the agent. The agent chooses the next action a_{t+1} according to the reinforcement reward r and the current state S_t of the environment. Then the individual can continue to choose the next appropriate action, then the state of the environment will change, and there are new rewards. However, as normally time step t is dynamic or even infinite, for example, as one of the classical reinforcement learning algorithms, Q-Learning algorithm, its summation of the rewards is defined as

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (1)$$

where k is the counter, λ is the reward decay factor, between [0,1].

The policy π represents the basis for individual to take action, that is, the individual will choose action according to the policy π. At each time step t, the path planning is to find a policy π such that the return R is maximized for the state s,

$$R = E\{\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, \pi\}, \quad (2)$$

where r is a one-step reward, s_0 is the initial state, γ represents a discount factor, between [0,1].

As a representative RL algorithm, the goal of Q-Learning algorithm is to find an optimal policy π, so that the expected return of agent in any state is the largest. Expected return involves the return value of a series of subsequent actions, and the agent can only get the immediate return of the current step at a time, which is not easy to solve directly. It can be converted into a computed Q-function, which gives the expected return of each state action pair under policy π:

$$Q^{\pi}(s, a) = E\{\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a, \pi\}, \quad (3)$$

where a_0 is initial action, s_0 is the initial state.

D. MULTI AGENT REINFORCEMENT LEARNING

extends RL to the field of multi-agent [18], so the Markov framework appears in the form of a tuple (n, S, $A_{1\ldots n}$, $R_{1\ldots n}$, T). A is the set of actions that an agent can take, n is the number of agents in the system, the transformation function T and the reward function R depend on the joint action $\boldsymbol{a} = (a_1,\ldots, a_n)$, $a_i \in A_i$. When the agents execute the joint policy $\pi = (\pi_1,\ldots \pi_n)$, the reward of each agent depends on

$$R_i^{\pi} = E\{\sum_{t=0}^{\infty} \gamma^t r_{i,t+1} | s_0 = s, \boldsymbol{\pi}\}. \quad (4)$$

When the agents execute the joint policy and joint action, the Q-function of each agent depends on

$$Q_i^{\pi}(s, \mathbf{a}) = E\{\sum_{t=0}^{\infty} \gamma^t r_{i,t+1} | s_0 = s, a_0 = \mathbf{a}, \boldsymbol{\pi}\}. \quad (5)$$

All agents strive to maximize common goals.

E. MADDPG

allows some additional information to be used for learning, as long as local information is used for decision-making when applied [13]. We use $\theta = [\theta_1,\ldots,\theta_n]$ to represent the parameters of n agents and $\pi = [\pi_1,\ldots, \pi_n]$ to represent the policies of n agents. For agent i of the cumulative expected reward

$$J(\theta_i) = E_{s,a_i\sim\pi_{\theta_i}}[\sum_{t=0}^{\infty} \gamma^t r_{i,t}], \quad (6)$$

using deterministic policy μ_θ, gradient formula as

$$\nabla_{\theta_i} J(\mu_i) = E_{x,a\sim D}[\nabla_{\theta_i}\mu_i(a_i|o_i)\nabla_{a_i} Q_i^{\mu}(x, a_1 \ldots a_n)|a_i = \mu_i(o_i)], \quad (7)$$

where x is the state, o represents the observation. MADDPG establishes value function for each agent, which greatly solves the shortcomings of MARL. D is an experience replay buffer with the element of (x, x', a_1,…, a_n, r_1,…, r_n), which keep a memory of past action-rewards and train the neural network with random samples from it instead of using the real time data, therefore eliminating the temporal autocorrelation problem. The centralized critic network is optimized as minimizing the loss:

$$L(\theta_i) = E_{x,a,r,x'}[(Q_i^{\mu}(x, a_1, \ldots, a_n) - y)^2], \quad (8)$$

where y is expected return computed by target network, it can be expressed as

$$y = r_i + \gamma Q_i^{\mu'}(x, a_1', \ldots, a_n')|a_j' = \mu_j'(o_j), \quad (9)$$

Q μ' i denotes the Q-value of the target network, and μ' is the target policy with parameter $\theta' j$ of delayed update. Policies of other agents can be obtained by fitting approximation.

F. MEAN FIELD THEORY

is a method to deal with the effect of environment on objects collectively and replace the sum of individual effects with average effects [47]. This method can simplify the study of complex problems and transform a multi-dimensional and difficult problem into a low-dimensional problem. In MARL, the dimension of joint action increase in proportion to the number of agents. It becomes infeasible to learn the standard Q-function Q(s, $\boldsymbol{a}$). To solve this problem, we incorporate the mean field theory into MARL, that is, when we study an agent, we approximate the actions of other agents to one action, and the joint actions of Q functions can be expressed as paired actions:

$$Q^i(s, \mathbf{a}) = \frac{1}{N^i} \sum_{k\in N(i)} Q^i(s, a^i, a^k), \quad (10)$$

where $N(i)$ is the index set of the neighboring agents of agent i, $\boldsymbol{a}$ is joint action, a^i represents the action of agent i and a^k represents the average action of other neighboring agents.

G. RVO

is a low-level obstacle avoidance algorithm that can avoid collision between agents without global coordination [14]. The velocity of an agent is chosen from valid velocities closest to a desired velocity. Valid velocities are velocities outside an average of the agent's current velocity and velocities outside the velocity obstacle of other agents. The invalid velocity will result in a collision course to the other agents.

H. OPENAI GYM

is a toolkit for developing and comparing RL algorithms. OpenAI Gym provides a unified environment interface and a platform for rebuilding a new environment, and it also encapsulates commonly used functions.

IV. FRAMEWORK OF MARL BASED ON CONGESTION DETECTION

In this section, we propose a framework of congestion detection-based MARL to prepare for our path planning method. This framework is divided into three parts. First, we divide the crowd into groups and select the leader. Second, we model the leader agent. Finally, we design the reinforcement learning element, and add congestion detection to the reward function.

A. CROWD PARTITION AND SELECTION OF LEADERS

When an evacuation occurs, due to the crowd's mentality and limited vision caused by obstacles position and crowd interference in complex environments, most people will decide to move according to the direction of other pedestrians, which will result in the forming of different groups [38], [39]. Once this group is generated, it will reach a relatively stable state internally.

Moreover, it is worth noting that people who are relatively familiar with the environment will become the leaders of the crowd according to the actual evacuation situation. In emergency situations, the path choice of the leader will affect the path choice of other people. Zhang *et al.* studied collective behaviour, showed a process of group gathering and proposed a suitable evacuation framework in which the crowd was designed as a leader-follower framework [40]. Pelechano *et al.* mainly studied the existence of leaders to obtain the best evacuation rate in groups divided by groups [41]. Many other studies divide the role of crowds into two parts, namely, leader and follower, and the evacuation process is based on the leadership framework [42]–[45].

In this paper, to crowd grouping for each room, we propose an algorithm of crowd grouping, which combines the idea of k-means algorithm [46]. According to the distance between people, the crowd is divided into k clusters. There are n people in the crowd. Each cluster has a cluster centre c_k, and (x_i, y_i) represent the coordinates of person i. After several updates, the cluster center of each room gradually stabilizes. Finally, we can obtain the coordinate of the cluster centre $c_k(x_k, y_k)$ after updating. J_{min} is the shortest distance from individual position to cluster center. $\bar{J}$ is the average from individual to cluster center. The details are outlined in Algorithm 1.

Algorithm 1 Group Generation Algorithm

Input: $n, k, i, p_i = (x_i, y_i), i \in n, J = \emptyset$;
Output: c_k, $group_k$
begin
  for each room m **do**
    Specify k cluster and randomly select cluster centre c_k
    repeat
      for $i = 1, 2, 3 \ldots n$ **do**
        for $j = 1, 2, 3 \ldots k$ **do**
          compute $J = ||p_i\text{-}c_j||^2$
          if $J_{min} = J$ **then**
            Cluster i into group $group_j$
          end if
        end for
      end for
    until each data are partitioned to the nearest centre;
    repeat
      for $j = 1, 2, 3 \ldots k$ **do**
        for $i = 1, 2, 3 \ldots n$ **do**
          compute $\bar{J} = \overline{||p_i - c_j||^2}$
          Update the cluster centre based on $\bar{J}$
        end for
      end for
    until the current cluster centre c_j is not updated, select each c_j as leader
  end for
end

After grouping the people, we selected the centre of mass as the leader within the group. The leaders and followers here have the following three constraints.

(1) Leaders need not be near door before evacuation but know the location information of at least one door.
(2) Leaders do not have to be at the front of the group when evacuating.
(3) Followers' path information is obtained from the leader, and they maintain the same evacuation path as the leader until they reach the door.

Based on the above analysis and constraints, we study the process of grouping selection intuitively through Fig. 3. The left side of the figure is the initial position of the crowd in the two-dimensional map scene before grouping. The right side of the figure is the positional distribution of the group after grouping and selecting the leadership within the group, in which the same group in a single room is represented by the same colour, and black represents the leadership position.

B. DESIGN OF LEADER AGENTS' MAPPING

Definition 1 (Input Function): Overall, the input function is responsible for determining how much information is provided from the leader to update the information source of

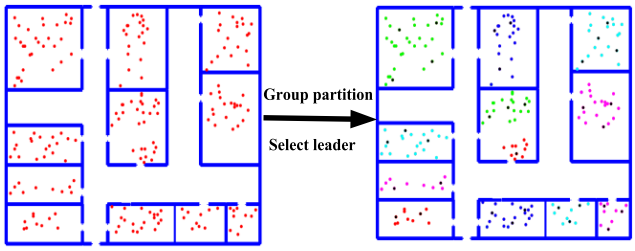


FIGURE 3. Crowd partition and selection of leader.

global path planning. Here, we pass the information (action, observation) used to train the global path from the leader to the leader agent.

Definition 2 (Output Function): Overall, the output function determines that the leader in the crowd will evacuate according to the planned path. Here, we use the function to pass the leader's next path information from the leader agent to the leader.

Definition 3 (Leader Agent): A leader agent can be defined by a six tuple <*Aid*, *Input*, *Communication*, *Output*, *Goal*, *Trigger*>. Aid is the identifier of a leader agent, such as $LeaderAgent_1 \ldots LeaderAgent_N$. The input component is obtained from the input function and is then passed on to the module of global path planning by the communication component. The communication component receives messages and passes the information from and to the module of global path planning. The output component returns the next path information to the leader by the output function. The goal refers to the selected optimal evacuation path to the evacuation exit. The trigger component consists of the event-condition-action rules and determines whether crowd evacuation is necessary, and then, it activates the leader agent to perform the appropriate action.

The design structure of leader agent mapping is shown in Fig. 4, here we use MADDPG as an example of the path planning algorithm. A leader agent corresponds to an actor network in the module of global path planning, and there is also a one-on-one relationship between the leader agent and the leader. The leader passes the current information on to the leader agent. Then, the leader agent interacts with the module of global path planning, executes the global path planning algorithm, and computes the optimal path from the current position to the exit. Subsequently, the next path information of the leader agent is selected, and the information is passed to the leader. The leader agent possesses the following capabilities:

(1) It can interact with the leader and global path planning module and guide the leader to evacuate under the optimal path during the evacuation process.
(2) It can participate in the collaboration among agents in MAS because each agent logically corresponds to an actor network, all agents learn to cooperate with one another and choose the action with the greatest joint reward in MAS for decentralized execution.

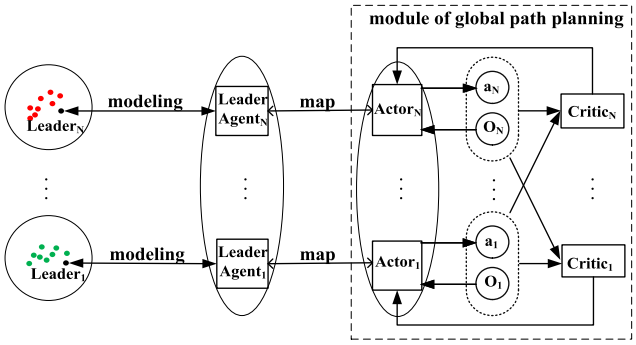


FIGURE 4. Design structure of leader agents' mapping.

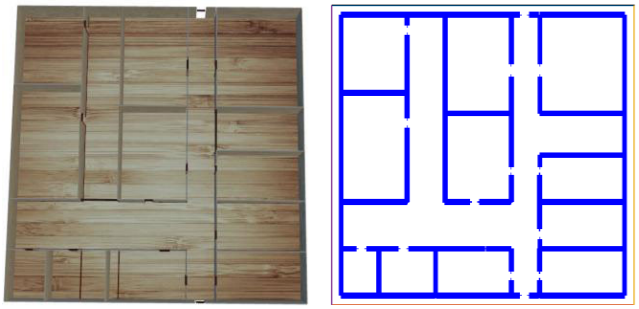


FIGURE 5. Environment of the 3D office building and 2D plane map.

C. CONGESTION DETECTION-BASED ELEMENTS OF REINFORCEMENT LEARNING

We define the RL elements based on congestion detection, which standards for MARL. The specific definitions are as follows.

Definition 4 (Environment): We model the real office hall scene, which is a complex scene with size of 300 m×300 m and two exits (as shown in Fig. 5(a)). Fig. 5(b) is a two-dimensional projection model of this three-dimensional environment, which consists of areas surrounded by boundary lines, each of which is a separate room. To verify the effectiveness of our path planning algorithm, we tested several algorithms in this environment.

We describe a limited number of leader agents in the form of coordinates and believe that tasks that can be performed well with a fully connected network and do not have to be conducted with pure images. Additionally, we build the same continuous bounded two-dimensional particle environment in the OpenAI Gym to train the optimal path of the leader agents. What needs to be clear is that

(1) The exit position and the wall obstacles are fixed.
(2) Except for the position of the leader agent is absolute coordinates, the rest are represented by relative coordinates.

Definition 5 (Observation Space): We use continuous observation space of the OpenAI Gym to restore the cognitive activities of pedestrians in the evacuation scenes to the greatest extent. The continuous observation space includes

the relative coordinates of each entity in the scene in each time period. Specifically, it includes the coordinates of the leader agent, the relative distance from the leader to the obstacles, the relative distance from the leader to the target, and the relative distance between the leader and other people.

Definition 6 (Action Space): We use discrete action space in the movement. Specifically, the action a_i of agent i is a discrete categorical variable represented as the one-hot encoding, and the total action space consists of east, west, south, north, southeast, northeast, southwest, northwest, which is conducive to planning a feasible path and avoiding collision before reaching the destination to better simulate the movement of pedestrians in the real world.

Definition 7 (Reward Function): The design of the reward function is especially important in reinforcement learning. The appropriate reward function ensures that the crowd can plan an effective route to the exit, and reward high route information to reduce evacuation time, and the inappropriate reward function makes the training meaningless. In the path planning of multi-agents, it is mainly necessary to complete three tasks, including reaching the destination, avoiding collision with obstacles, and avoiding collision between agents. Here, we set up the crowd density detection area at the door to judge the congestion at the door and to further realize the choice of exit and the avoidance of congestion. The obstacle is named *obs*, and the set of agents position (x, y) is expressed as

$$\{(x_1, y_1) \ldots (x_i, y_i), (x_j, y_j), \ldots (x_n, y_n)\} \in (x, y). \quad (11)$$

The following assumptions provide the basis for our work.

Assumption 1: The position of the target centre is (x_{goal}, y_{goal}), the position of the agent i centre is (x_i, y_i), and the Euclidean distance *dist* between them is according to Eq. (12), it can also be expressed as $dist((x_i, y_i), (x_{goal}, y_{goal}))$,

$$\sqrt{(x_i - x_{goal})^2 + (y_i - y_{goal})^2}. \quad (12)$$

Assumption 2: The radius of the agent is R, and $dist((x_i, y_i),(x_j, y_j))$ represents the Euclidean distance between agent i and j, the condition of collision can be further expressed as

$$dist\left((x_i, y_i), (x_j, y_j)\right) \leq 2R. \quad (13)$$

Assumption 3: The obstacle will not move and has no perception, $dist((x_i, y_i), (x_{obs}, y_{obs}))$ represents the Euclidean distance between agent i and obstacle. The condition for collision between agent i and the obstacle is Eq. (14),

$$dist\left((x_i, y_i), (x_{obs}, y_{obs})\right) \leq R. \quad (14)$$

Assumption 4: For the density detection sector for the N gates in the scene, the area is S, according to coordinates, the average speed of leader agent i in the crowd density detection area is avg_speed_i, when agent enters this area, avg_speed_i is the same as half the sum of the speed of the start time and the end time of this period, the maximum speed is max_speed_i, and the speed score of agent i is divided into value; then, according to Eq. (15),

$$value\_speed_i = min(\frac{avg\_speed_i}{max\_speed_i}, 1). \quad (15)$$

Each leader agent adjusts the optimal speed according to the distance during training to avoid collisions between agents. When $value\_speed_i$ is higher reach 1, the leader agent is faced with no congestion, otherwise, it means congestion. If in the congested area, $value\_speed_i \leq N$ is considered to be congested when the followers are considered.

Based on the above analysis, we obtain our comprehensive reward function as follows.

(1) In a complex environment, not every agent knows exactly where the exit is, which requires the agent to interact with the environment continuously. According to assumption 1, we set the distance from the agent to the target point as $dist((x_i, y_i), (x_{goal}, y_{goal}))$, and this paper designs multiple exits; then, $r_{goal} = -min(dist((x_i, y_i), (x_{goal}, y_{goal})))$.
(2) There is no collision between the agents when $r_{coll\_a} = 0$, but there is a collision when $r_{coll\_a} = -3$.
(3) There is no collision between agents and obstacles when $r_{coll\_Obs} = 0$, but there is a collision when $r_{coll\_Obs} = -1$.
(4) According to assumption 4, there is no congestion in the scene when $r_{cong} = 0$, but there is congestion when

$$r_{cong} = -\frac{1}{value\_speed}. \quad (16)$$

The total reward for r_{total} is based on Eq. (17):

$$r_{total} = r_{goal} + r_{coll\_a} + r_{coll\_Obs} + r_{cong}. \quad (17)$$

V. A PATH PLANNING APPROACH BASED ON DRL

In this section, we elaborate on our proposed hierarchical path planning. The design process of hierarchical path planning is divided into two parts. Specifically, the upper layer uses the IMADDPG algorithm under a framework of congestion detection-based MARL to achieve global path planning. At the lower layer, we use the RVO algorithm to achieve collision avoidance.

A. THE UPPER-LAYER GLOBAL PATH PLANNING BASED ON IMADDPG

We propose the IMADDPG algorithm, in addition to the critic network and actor network, we use the network which introduces the mean field theory [47], and we call it the mean field network(MFN). Embedding MFN into the decentralized policy of each agent, which can increase other agent's return when training. In addition, the mean field theory is used to update the parameters in our algorithm and reduce the complexity of training additional samples. This reduces the interaction between an agent and its neighboring agents to the interaction between two agents. The expansion of model space and the training complexity caused by the number

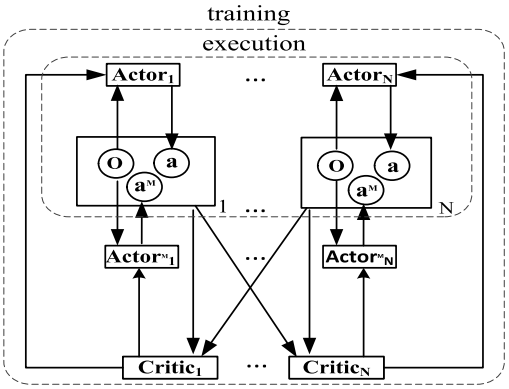


FIGURE 6. IMADDPG schematic diagram.

of agents are greatly simplified. The IMADDPG schematic diagram is shown in Fig. 6.

In IMADDPG, the experience replay buffer D^M contains $(x, a^M, r, \bar{a^M}, x')$, where x' is the state at the next time, r is reward, joint action a^M is generated not from original actor but from the MFN, $\bar{a^M}$ is mean field action. The Q function of agent j expressed as $Q_j(s, a) = Q(s, a_j^M, \bar{a_j^M})$, where a_j^M is action of agent j from the MFN, $\bar{a_j^M}$ is mean field action of agent j, it can be expressed as

$$\bar{a}_j^M = \frac{1}{N^j} \sum_k a^{M^k} \tag{18}$$

where mean action $\bar{a_j^M}$ in mean field network based on neighborhood $N(j)$ of agent j, and represent action a^{M^k} of each neighbor k in terms of the sum of $\bar{a_j^M}$. The mean field Q-function Q_j $(s, a_j, \bar{a}_j)$ updates are as follows:

$$Q_j(x, a_j^M, \bar{a}_j^M) \leftarrow (1-\alpha) Q_j(x, a_j^M, \bar{a}_j^M) + \alpha[r^j + \gamma v_j(x')] \tag{19}$$

where α denotes the learning rate, π represents random policy, $v_j(x')$ is the mean field value function, it can be expressed as

$$v_j(x') = \sum_{a_j^M} \pi_j\left(a_j^M | x', \bar{a}_j^M\right) E_{\bar{a}_j^M\left(a_j^M\right) - \pi_{-j}^M}\left[Q_j\left(x', a_j^M, \bar{a}_j^M\right)\right] \tag{20}$$

More concretely, for the key part of the training, reducing the error between estimated expected return and online expected return on each i individual Q-network, the parameter φ_i^M is optimized as minimizing the loss:

$$L(\varphi_i^M) = E_{x, a^M, r, x' \sim D^M}[(Q_i(x, a_i^M, \bar{a}_i^M) - y_i)^2], \tag{21}$$

where y_i is expected return computed by target network, it can be expressed as

$$y_i = r_i\left(x, a_i^M, \bar{a}_i^M\right) + \gamma v_i\left(x'\right) \tag{22}$$

The above is the step in which each centralized critic is trained. Then, the actor network corresponding to the each agent i under deterministic policy μ is optimized using gradient as follows:

$$\nabla_{\theta_i} J = E_{x, a^M \sim D^M}[\nabla_{\theta_i} \mu_i(o_i) \nabla_{a_i} Q_i (x, a_1^M, \ldots, a_i, \ldots, a_N^M) | a_i = \mu_i(o_i)]. \tag{23}$$

The introduced networks MFN is to use the idea of average field theory to generate samples. The parameter θ_i^M of the MFN of each agent i is optimized using the specially computed gradient as follows:

$$\nabla_{\theta_i^M} J = E_{x, a^M \sim D^M}[\nabla_{\theta_i^M} \mu_i^M(o_i) \nabla_{a_i^M} Q_{-i} (x, a_i^M, \bar{a}_i^M) | a_i^M = \mu_i^M(o_i)]. \tag{24}$$

where $-i$ is all agents except i. The policy gradient of the MFN is computed using the policy gradient of the Q-network of $-i$ and the gradient of MFN with its parameters. During the execution period after training, the Q-networks and MFNs of all agents are removed, only the actor network of each agent controls the action based on local observation. This still conforms to the principle of centralized training and decentralized execution. The implementation details are outlined in Algorithm 2.

For agent i, θ represents the parameters of the online actor for updating parameters instantly, θ' represents the parameters of the target actor for delayed updating parameters, θ^M and θ'^M represent the parameters of the online actor and the target actor for the MFN, φ^M and φ'^M represent the parameters of the online cirtic and the target critic for the MFN, τ is a balance factor.

We propose that the top-level global path planning method mainly includes two parts, namely, training the model and getting the path. IMADDPG algorithm is adopted to train the model, it has robustness when solving large-scale crowd path planning problem. We complete the training within the prescribed total number of training steps *num_episodes*, and action exploration is performed in each *max-episode-length* steps. The size of mini-batch we set is 1024. The numerical size of experience replay buffer is the product of the numerical size of mini-batch and the numerical size of *max-episode-length* we set. After selecting the action a, the agent i reaches the next state x' from the current state x and simultaneously obtains the observation total return r given by the environment. Each step $(x, a^M, r, \bar{a^M}, x')$ is stored in the experience pool for the parameter learning of the module, and then, we update $x \leftarrow x'$ while replacing newly generated batch $(x, a^M, r, \bar{a^M}, x')$ with the previously stored $(x, a^M, r, \bar{a^M}, x')$ tuple until the maximum iteration step is reached. We can steadily update the parameters through the network that both contain real-time updates to the parameters of the online network, and we can delay the updates to the parameters of the target network. The main improvement of IMADDPG is that we add a new network structure on the basis of MADDPG, in this network structure, we introduce the mean field theory to average the actions of other agents. Specifically, in algorithm 2, our actions are not only from the environment, but also from the mean field network. In addition, in the latter

Algorithm 2 The improved Multi-Agent Deep Deterministic Policy Gradient Algorithm (IMADDPG) for N Agents

for episode $= 1$ to *num_episodes* **do**
  Initialize a random process P for action exploration, and
  receive initial state information x.
  for $t = 1$ to *max-episode-length* **do**
    For each agent i, select action $a_i = \mu_{\theta i}(o_i)$;
    compute new mean action $\bar{a} = (\bar{a_1}, \ldots, \bar{a_N})$.
    Execute actions $a = (a_1, \ldots, a_N)$, observe reward $r = (r_1, \ldots, r_N)$, next state x', Store(x, a, r, x', $\bar{a}$) in replay buffer D^M, set $x \leftarrow x'$

    for agent $i = 1$ to N **do**
      Sample a random mini-batch of S samples $(x, a, r, x', \bar{a})$ from D
      Set $y_i = r_i(x, a_i^M, \bar{a}_i^M) + \gamma v_i(x')$ by Eq. (20)
      Update critic by minimizing loss:
      $L(\varphi_i^M) = E_{x,a^M,r,x' \sim D^M}[(Q_i(x, a_i^M, \bar{a}_i^M) - y_i)^2]$
      Update actor using sampled policy gradient:
      $\nabla_{\theta_i} J = E_{x,a^M \sim D^M}[\nabla_{\theta_i} \mu_i(o_i) \nabla_{a_i} Q_i(x, a_1^M, \ldots, a_i, \ldots, a_N^M) | a_i = \mu_i(o_i)]$
      Update mean field network actor using mean field theory sampled policy gradient:
      $\nabla_{\theta_i^M} J = E_{x,a^M \sim D^M}[\nabla_{\theta_i^M} \mu_i^M(o_i) \nabla_{a_i^M} Q_{-i}(x, a_i^M, \bar{a}_i^M) | a_i^M = \mu_i^M(o_i)]$
    end for
    Update target network parameters for each agent i:
    $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$;
    $\theta_i'^M \leftarrow \tau\theta_i^M + (1-\tau)\theta_i'^M$;
    $\varphi_i'^M \leftarrow \tau\varphi_i^M + (1-\tau)\varphi_i'^M$;
  end for
end for

half of the algorithm 2, IMADDPG updates a MFN more than MADDPG by Eq. (24), the policy gradient of the MFN is computed using the policy gradient of the Q-network of $-i$ and the gradient of MFN with its parameters, and when we last update the parameters, we update an additional parameter φ^M, which will be used in Eq. (21). And the generated action samples from MFN can train original actor and critic networks, which add the return of other agents and enable all agents to perform together better in the collaborative planning task.

After the above training within the prescribed number of training steps, the trained model is obtained by IMADDPG, which allows the agent to move around according to the framework of MARL we set. Next, we elaborate path planning based on the IMADDPG algorithm. When an initial location $S_{leader-i}$ of the leader agent i is known and the action is performed, the next location $S'_{leader-i}$ is reached; then, the location is taken as the current location to continue the path-finding, and the process is repeated until it reaches any gate position S_{goal-j}. The sequence of positions is regarded as an evacuation path of the leader agent i, and these k paths are temporarily stored in path buffer $buffer^k_{leader-i}$. However, because of the influence of other agents simultaneously, the total reward r_{total} of the path defined in the MARL framework will be very different. Then, we complete the evaluation of the sequence of the agent's position for k paths in $buffer^k_{leader-i}$. We can obtain the reward $R\text{-}buffer^m_{leader-i}$ of path m according to reward and punishment rules of the MARL framework discussed in section 3. When k candidate paths in the path buffer are traversed, we select the optimal evacuation path $path_{leader-i}$ of agent i with the greatest reward $R\text{-}buffer^{max}_{leader-i}$ in the path library *buffer*. The implementation details are outlined in Algorithm 3.

Algorithm 3 Path Planning Based on the IMADDPG Algorithm

Input: $s_{lead-i}, s_{goal-j}, buffer_{lead-i} = \emptyset, path_{lead-i} = \emptyset$
Output: $Path_{lead-i}$
begin
  $k \leftarrow 0, m \leftarrow 0, R - buffer^{max}_{leader-i} \leftarrow 0, R - buffer^t \leftarrow 0$
  for leader agent i **do**
    for each goal j **do**
      repeat
        $s_{lead-i} \leftarrow s'_{leader-i}$;
        $buffer^k_{leader-i} \leftarrow buffer^k_{leader-i} \cup s_{leader-i}$;
      until $s_{leader-i} \leftarrow s_{goal-j}$;
      $k++$;
    end for
    repeat
      if $then R - buffer^m_{leader-i} > R - buffer^{max}_{leader-i}$ **then**
        $R - buffer^{max}_{leader-i} \leftarrow R - buffer^m_{leader-i}$;
      end if
      $m++$;
    until $m \leftarrow k$
  end for
  $path_{leader-i} = buffer^{max}_{leader-i}$;
End

B. THE LOWER-LAYER COLLISION AVOIDANCE BASED ON RVO

At the bottom, as we constrained leaders in section 3, with the addition of followers, the position of leader does not have to be in the front of the queue, the followers use the same path as the leader in each group. Therefore, we focus on collision avoidance in this section.

We set the preferred velocity v^{pref} and collision-free velocity v^{coll} of each individual. The preferred velocity is mainly used to lead the individual to the door, and the collision-free velocity is mainly for collision avoidance. p_i represents the global path of individual i obtained by the upper-layer, and p_i represents the position of individual i at time t. Preferred velocity v^{pref} includes velocity magnitude v and direction d_i^t, $v_{min} = 1.5$ m/s, desired velocity $v_{max} = 3.5$ m/s, the preferred

velocity v is initialized in the fixed interval v_{max} $[v_{min}, v_{max}]$, and p_i^{t+1} is the current subgoal visible to p_i^t. Everyone is evacuated according to the sub-target node, the subgoal enters the view range, and the next sub-goal becomes the guiding direction of global target, $d_i^t = p_i^{t+1}\text{-}p_i^t$. Therefore, $v_i^{pref} = v \times d_i^t$.

We use RVO [14] to realize the process of obstacle avoidance. Collision avoidance can be implemented in two steps. First, we calculate all the velocities of individual i that will collide with individual j in a neighbourhood. Second, the penalty measure can determine that individual i will choose the optimal collision-free velocity v^{coll}. Once the individual's collision-free speed v_i^{coll} is obtained, its position is updated to $x_i^{n+1} = x_i^n + \Delta t v_i^{coll}$. Because of the application of collision avoidance, pedestrians can keep distance and move steadily without deviating from the evacuation path. This makes all pedestrians' movement in the evacuation process based on the optimal path trained by upper-level path plan.

VI. EXPERIMENT AND ANALYSIS

To illustrate the performance efficiency and authenticity of the proposed path planning method, we perform the following tests in the next sections: (1) a comparison of the rewards between our method and other DRL methods; (2) a comparison between the algorithm training and evacuation time between our method and other DRL methods; (3) a comparison of the evacuation time between our method and other swarm intelligence algorithms for evacuation planning; (4) a comparison of the evacuation effect of the IMADDPG in the congestion detection area; (5) An evacuation behavior analysis of real video scenario; and (6) a crowd evacuation simulation realized on our rendering platform. All reported results were obtained on a machine with a 2.4 GHz Intel(R) Xeon CPU with 32GB of RAM.

https://github.com/Sfree973/path-planning-for-crowd-simulation

A. COMPARISON OF THE REWARD RETURNS IN DIFFERENT DRL ALGORITHMS

To verify the performance of our proposed hierarchical path planning method, we model the office hall with an equal scale and set up the training scene in the OpenAI Gym environment. The population size is 200, the individual radius is $r = 0.25$ m, the quality of the individual is $m = 80$ kg, and we perform the path planning method from a 300 m $\times$ 300 m plane area as shown in Fig. 5(b). There are four fully connected layers, each with 64 neurons, the activation function uses ReLu. In addition, γ, α, τ and size of mini-batch we set is 0.9, 0.02, 0.8 and 1024. We conduct unified training with the double Dueling DQN method in [37], the DDPG algorithm [25] and MADDPG algorithm [13]. Since it is difficult to limit the total number of steps in the training, we chose 80,000 iterations as the end and recorded the average total reward for every 1,000 training episodes. The results are shown in Fig. 7. The red line represents the average reward of our method based on the IMADDPG

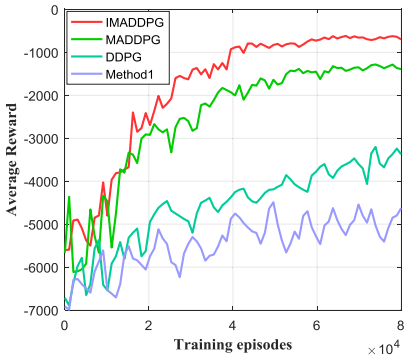


FIGURE 7. Comparison of the average reward.

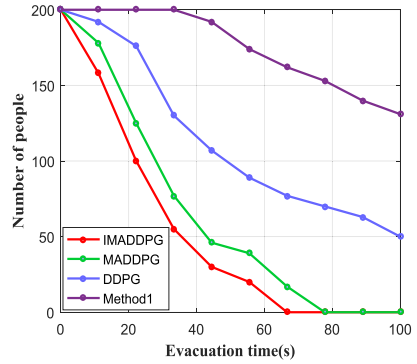
algorithm, the green line represents the average reward of the MADDPG algorithm [13], the blue line represents the average reward of the DDPG algorithm [25] and the purple line named Method1, it represents the average reward of [37].

As shown in Fig. 7, the average rate of return of the IMADDPG-based method is significantly different from the average rate of return of the other methods in the same action space. Because of the setting of the reward function, multiple exits and agents, negative reward is inevitable. Moreover, the reward for DRL may not converge to optimal learning, which differs from supervised learning, but it is clear that our method has converged after 60,000 steps of training and that the reward returns have always been higher than the other algorithms. The overall trend of IMADDPG and MADDPG is remarkable, which increases by about 5000 and 4000, respectively, while the remaining DDPG and method1 methods changed slowly and slightly, increasing about 3600 and 2000 respectively. Before the end, our algorithm completes convergence and its reward values are all at their highest. Although MADDPG is also close to convergence, the reward value is about 800 lower than our algorithm, the remaining two algorithms do not reach convergence, and the reward value of method1 still oscillates around -5000.

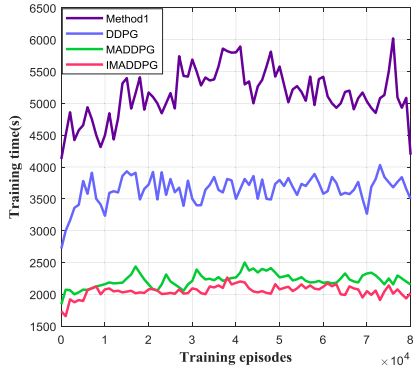
This result also shows that in the same environment, our approach can better adapt to learning how to map out better paths. For Method1 and DDPG, since this DRL algorithm unadaptable have learned in large-scale agent environment, as can be seen from the above figure, the number of agents still greatly affect the convergence of the algorithms. And for MADDPG, it can learn policy in a multi-agent environment, but we take into account the returns of other agents more than MADDPG, our method makes cooperative planning more efficient and makes the average reward higher. In general, under the same number of training steps, our method can obtain high reward and fast convergence, under the same reward function, our method based on the IMADDPG algorithm can conduct the path planning in a complex environment.

B. COMPARISON OF THE TIMES OF DIFFERENT DEEP REINFORCEMENT LEARNING ALGORITHMS

In this section, we use the evacuation time and training time to evaluate the effectiveness of our method. We compare the



(a) Comparison of the average evacuation time.

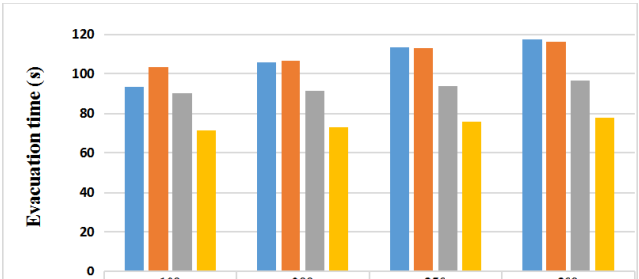


(b) Comparison of the average training time.

FIGURE 8. Comparison of the average time.

time of the four methods in the scene. Fig. 8 (a) shows a comparison of the time of evacuation simulation of 200 people in the same scene. The overall change trend of IMADDPG and MADDPG was significant, with the increase of evacuation time, the number of evacuation was reduced from 200 to 0. Moreover, at 50 seconds, the number of people in the scenarios using these two methods is less than 50, which is less than the number of people in the last moment of other methods. The overall trend of DDPG and Method1 is smaller. At 100 seconds, Method1 has 131 people in the scene, and its decline is the smallest, because Method1 does not train a reasonable evacuation path, which results in people still in the scene, and the evacuation time remains at a high level. In addition, at 10 seconds, the four methods have shown significant differences. The number of scenarios using our method is close to 160, while the other methods are more than 170 pedestrians. In 70 seconds, our method has completely opened the gap with other methods, using our method successfully evacuated all people, while the remaining number of MADDPG, DDPG and Method1 scenarios is 23, 78 and 164, respectively.

Training time is shown in Fig. 8 (b). The top Method1 has the longest training time and obvious fluctuation, average training time is 5136, the shortest training time is 4125 seconds, and the algorithm is inefficient when applied to large-scale crowd path planning. Even if we discretize the action, this algorithm itself does not solve the ever-changing environment faced by each agent in the multi-agent system,



	100	200	250	300
PSO	93.3	106.1	113.7	117.5
ACO	103.3	106.8	113.2	116.1
IABC	90.1	91.3	93.7	96.5
IMADDPG	71.6	73.3	75.7	77.8

FIGURE 9. Comparison of the evacuation times.

and the policy is constantly changing, which is the main reason for the long training time. For the DDPG algorithm in Fig. 8 (b), its training time still has fluctuation, the average training time is 3632 seconds, and the minimum training time is 2719 seconds. Each critic network still only inputs its own state-action information, which greatly affects the training duration of the algorithm. Although the DDPG algorithm can also evacuate people to a certain extent, its training time is approximately twice as long as our method. For MADDPG method, the average training time is 2235 seconds, the change of training time is still unstable, the minimum training time is about 1844, it can complete the evacuation in 80 seconds, but it did not plan the optimal path for the agent in a collaborative planning scenario, which is reflected in the evacuation time. Moreover, MADDPG does not consider the return of other agents, but only updates its own return, which is not conducive to shortening the training time. In our contrast experiment, the average training time and evacuation time of our method based on the IMADDPG algorithm in the crowd path planning are much shorter than the rest of the path planning algorithm, the average training time of IMADDPG is 2052 seconds, and the minimum fluctuation of time is 1656. These two times are in the shortest time among the above methods. Our hierarchical path planning method based on IMADDPG realizes the centralized learning and decentralized execution in the multi-agent environment, and the method adds the mean field network to use the mean field idea to reduce the complexity of training additional samples and maximize performance of a collaborative planning task in our training period. That is why time stays at a low level. Our method can be obtained by combining the analysis in the previous section, and the performance of our method based on the IMADDPG algorithm is better than the rest of the path planning method based on DRL methods.

C. COMPARISON OF THE EVACUATION TIMES OF SWARM INTELLIGENCE ALGORITHMS

In addition to the DRL algorithm, we compare the current traditional and improved swarm intelligence algorithms applied to crowd path planning, including the PSO algorithm,

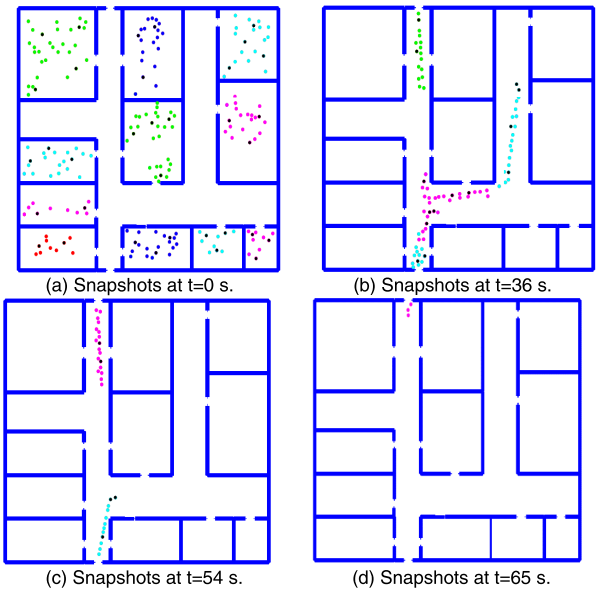


FIGURE 10. Snapshots generated by the use of the congestion detection area.

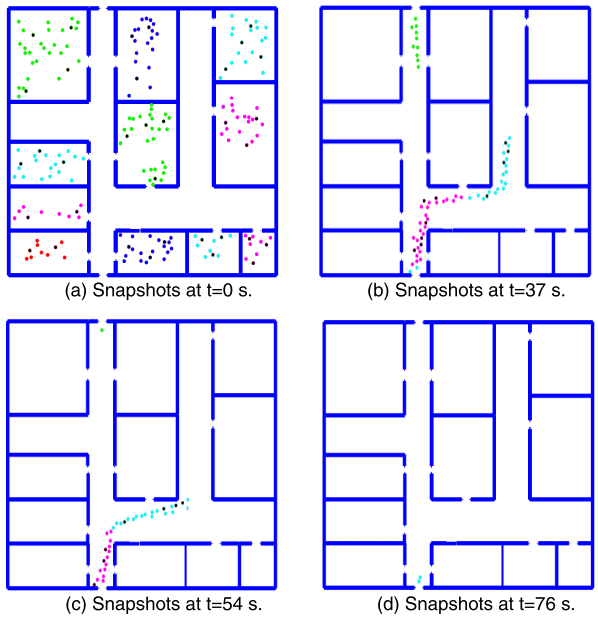


FIGURE 11. Snapshots generated by nonuse of the congestion detection area.

the ACO algorithm and the improved ABC algorithm [48]. In this example, we still evaluate the effectiveness of our method by comparing evacuation times by using Visual Studio 2012 as a platform. The population size is varied from 100 to 300 in the same office hall scene, as shown in Fig. 5(b). We performed 30 experiments to obtain the average evacuation time for each case. We compare the evacuation times as shown in Fig. 9.

As seen from Fig. 9, our column chart shows the comparison of evacuation time of four algorithms under different numbers of people. The table is the numerical value.

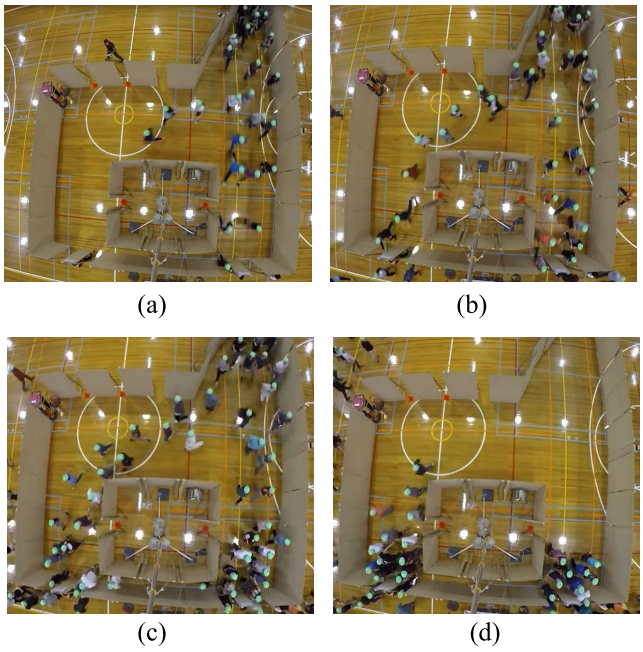


FIGURE 12. The scene of crowd evacuation drill.

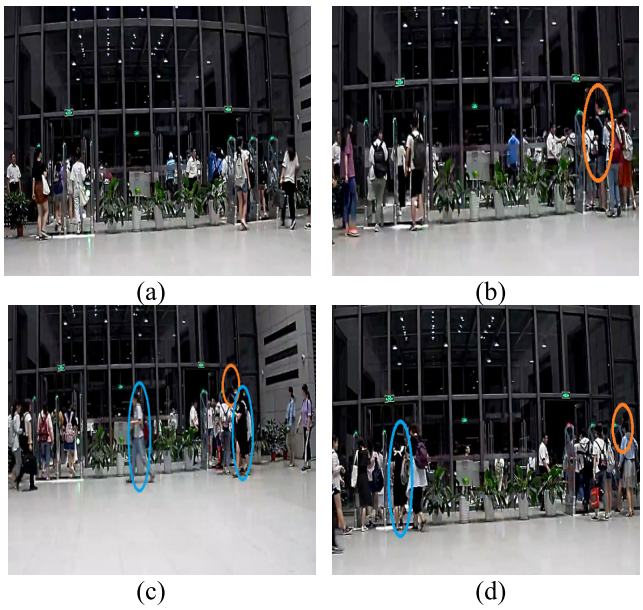


FIGURE 13. The scene of Shandong Normal University Library.

The evacuation time of PSO and ACO is obviously higher than that of IABC and IMADDPG. In the complex office hall, the evacuation time will increase with the increase of the evacuation crowd in this situation. Although the IABC algorithm can reduce the evacuation time compared with the traditional particle swarm optimization algorithm, because the swarm intelligence algorithm is applied in a complex environment, an exit bottleneck problem appears gradually, and the evacuation efficiency is not dramatically improved. Although the swarm intelligence algorithm is a classic representation of simulated crowd, it is unsuitable for large-scale and complex path planning due to its own limitations.

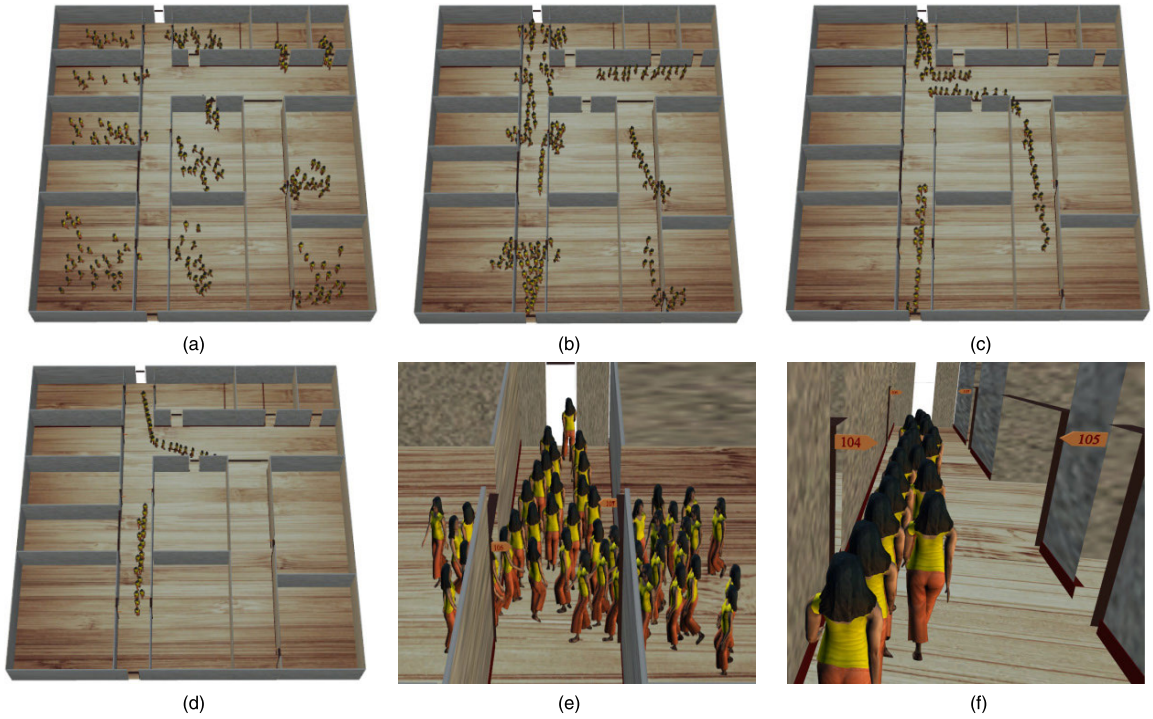


FIGURE 14. A sequence of the crowd simulation results of a 3D scene.

However, our method based on the IMADDPG fully considers the centralized learning path knowledge among the leader agents, and then, decentralized execution under the premise of avoiding collisions. This is more suitable for crowd behaviour than swarm intelligence in some ways, and has a strong adaptability in complex environments. As seen, the yellow column that represents our method always takes the least evacuation time. The average evacuation time of our method improves by 30.7% compared with the PSO, improves by 32.1% compared with the ACO, and improves by 19.7% compared with the IABC. These results show that our hierarchical method can be better applied to crowd evacuation.

D. COMPARISON OF THE EVACUATION RESULTS OF THE CONGESTION DETECTION AREA

We continue to perform simulation experiments in the two-dimensional office hall scenario, and use MATLAB as the simulation platform to obtain the visualization effect that refer to different time points as shown in this section. Pedestrians evacuate orderly under the proposed path planning method. To further compare the impact of the congestion detection area that we designed on exit selection and congestion avoidance in the crowd evacuation, we conducted a set of experiments in this section, and the experimental results are shown in Fig. 10 and 11.

In the case of adding the congestion detection area, Fig. 10(b) and (c) show the state of the 36 second and 54 second crowds in the evacuation process, respectively. Instead of moving to the nearest exit, the crowd in the hall corridor moves towards another less congested exit, which is closer to the crowd's ability to judge congestion and the exit selection in the case of congestion. As shown in Fig. 11 (c), in the case of not adding the congestion detection area, the crowd is still waiting at the congested gate when the time has reached 54 seconds, although the other door has no one. This is the main reason why a small number of pedestrians still failed to reach the exit in 72 seconds in Fig. 11 (d).

E. EVACUATION BEHAVIOR IS VERIFIED THROUGH CAPTURED MONITORING VIDEO

The above experiments have verified the evacuation efficiency of the proposed method. In order to ensure the authenticity of the proposed method, we used video of the evacuation scene captured by the surveillance camera to verify the behavior of evacuation, such as congestion, exit selection, formation of queues and so on. Fig. 12 is part of the scene of emergency evacuation drill. We mainly selected the picture at the exit. In this scenario, we can directly observe the congestion, exit selection, formation of queues and the crowd following phenomenon. In Fig. 12 (a), the crowd began to evacuate to the exit. Because of the different directions of the exits and the large number of people on the right side, three pedestrians chose to evacuate directly from the left exit. In (b), the people far away from the door try to follow the people ahead to the exit, which initially formed evacuation queues in different directions. People within the queue are close to each other and have the same goal orientation. They show a tendency to act together to resist environmental pressures, even if they have no social connections. This is also our grouping principle, location and goals are the key to grouping. There was the congestion at the door in Fig. 12 (c) and (d).

In addition, the people close to the crowded exit on the right gradually choose to move to the less crowded exit on the left. Because there were so many people, people were waiting to leave at the exit.

Fig. 13 is the evacuation process of the fire drill captured by the library of Shandong Normal University. There are two exits for students to evacuate at the library exit. In this scenario, we can directly observe congestion and exit selection behavior. In Fig. 13 (a), students ran toward the nearest door and evacuate smoothly. In Fig. 13 (b), due to unexpected circumstances, the population density at the right exit increased, resulting in congestion. The student in the yellow ellipse chose to wait, but there was no congestion at the left exit, and the people on the left side still left orderly evacuated. In Fig. 13(c), small number of students in the blue ellipse give up the right exit and choose to leave from the left exit. Some students still choose to stay in the nearest right exit in Fig. 13 (d) is still crowded. The students in the blue ellipse are about to evacuate successfully, while the student in the right yellow ellipse is still in the congested area and have not evacuated safely in a short time. This video capture confirms the necessity and importance of exit selection in our path planning process.

F. CROWD SIMULATION RESULTS

We adopt the Microsoft. Net Framework 4.5, Microsoft XNA Game Studio 4.0, Microsoft Visual Studio 2012, OpenSceneGraph to develop a crowd evacuation simulation platform. The proposed method is validated on the existing simulation platform as Fig. 14, which shows the effect sequence of group evacuation after introducing the pedestrian model. (a), (b), (c) and (d) vividly show us that 200 people are evacuating to two exits in the office; compared with the far-away images, (e) and (f) provide close-up evacuation images.

According to the images display, our method and simulation platform can express social and emergency behaviours such as crowd gathering, formation of queues, exit selection and congestion etc. with high visual authenticity. In addition, video in the previous section proves the authenticity of our simulation results, which will ensure that our method can be used to extend the real scenario. It is beneficial to provide such an intuitive way of viewing, which makes our research results not only more visually easier to integrate with architectural design and planning management but also easier to understand.

VII. CONCLUSION

In this paper, the problem of path planning in the field of crowd evacuation is studied. How to improve the efficiency of evacuation while considering complex circumstances (e.g., exit selection and congestion) remains a challenge for the crowd evacuation simulation. However, existing swarm intelligence algorithms and novel DRL algorithms will affect the evacuation efficiency for large-scale crowd path planning, and evacuation behavior is rarely considered. Therefore, we propose a hierarchical path planning method based on improved DRL algorithm to search collaboratively for the optimal evacuation path. First, we propose a framework of congestion detection-based MARL. In this framework, we consider a multi-agent system model for the leaders after grouping, and presuppose a leading role of leaders in path finding for their followers in the group by considering the congestion detection area of the reward function, which is beneficial to realize the reasonable selection of evacuation exits. Next, we propose IMADDPG, which adds an additional mean field network to evaluate the returns of other agents and enables all agents to maximize the performance of a collaborative planning task. Then, we implement the hierarchical path planning method based on DRL under the framework, the upper layer is based on the IMADDPG algorithm to perform global path planning for multi-agent, the lower layer uses RVO to avoid collisions between the pedestrians. Moreover, to verify the effectiveness of the proposed method, our research team compares not only with other path planning algorithms based on DRL in the related work but also with the traditional and improved swarm intelligence algorithm used in evacuation simulation, we also proved that our method can be extended to the real world through real video. Finally, we have performed a more visually realistic analysis and verification of the method on our simulation platform.

The results of the experiments that uses our approach produces several conclusions. In a complex environment with obstacles and multiple exits, our path planning method based on DRL has obvious advantages over the DRL and swarm intelligence algorithm compared in this paper. The congestion detection area that we designed can reduce the crowd congestion in emergencies to a certain extent, which will ensure that prevent and reduce the number of deaths with safer and faster way. This method can be applied to emergency response management for safety accidents. Since the multi-agent system combined with DRL has great expandability, we would like to extend our method to more complex crowd evacuation problems in the future.

REFERENCES

[1] M. Haghani and M. Sarvi, "Pedestrian crowd tactical-level decision making during emergency evacuations," *J. Adv. Transp.*, vol. 50, no. 8, pp. 1870–1895, 2016.

[2] J. E. Almeida, R. J. F. Rosseti, and A. L. Coelho, "Crowd simulation modeling applied to emergency and evacuation simulations using multi-agent systems," in *Proc. 6th Doctoral Symp. Inform. Eng. (DSIE)*, Porto, Portugal, 2011, pp. 1–12.

[3] Y. Li, H. Liu, X. Zheng, Y. Han, and L. Li, "A top–bottom clustering algorithm based on crowd trajectories for small group classification," *IEEE Access*, vol. 7, pp. 29679–29698, 2019.

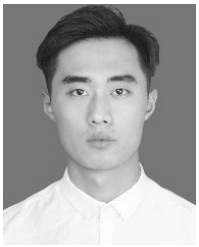
[4] L. Tan, M. Hu, and H. Lin, "Agent-based simulation of building evacuation: Combining human behavior with predictable spatial accessibility in a fire emergency," *Inf. Sci.*, vol. 295, pp. 53–66, Feb. 2015.

[5] V. Sadhu, G. Salles-Loustau, D. Pompili, S. Zonouz, and V. Sritapan, "Argus: Smartphone-enabled human cooperation for disaster situational awareness via MARL," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2017, pp. 79–81.

[6] M. Manley, Y. S. Kim, K. Christensen, and A. Chen, "Airport emergency evacuation planning: An agent-based simulation study of dirty bomb scenarios," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 10, pp. 1390–1403, Oct. 2016.

[7] L. Torrey, "Crowd simulation via multi-agent reinforcement learning," in *Proc. 6th AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, 2010, pp. 89–94.

[8] F. Martinez-Gil, M. Lozano, and F. Fernández, "MARL-Ped: A multi-agent reinforcement learning based framework to simulate pedestrian groups," *Simul. Model. Pract. Theory*, vol. 47, pp. 259–275, Sep. 2014.
[9] F. Martinez-Gil, M. Lozano, and F. Fernández, "Multi-agent reinforcement learning for simulating pedestrian navigation," in *Proc. Int. Workshop Adapt. Learn. Agents*. Berlin, Germany: Springer, 2011, pp. 54–69.
[10] H. Lee, "Human crowd evacuation framework and analysis using look-ahead-based reinforcement learning algorithm," *Int. J. Digit. Hum.*, vol. 1, no. 3, pp. 248–262, 2016.
[11] N. D. Nguyen, T. Nguyen, and S. Nahavandi, "System design perspective for human-level agents using deep reinforcement learning: A survey," *IEEE Access*, vol. 5, pp. 27091–27102, 2017.
[12] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass, "Hierarchical deep reinforcement learning for continuous action control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5174–5184, Nov. 2018.
[13] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.
[14] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 1928–1935.
[15] E. Higo, N. Okada, K. W. Hipel, and L. Fang, "Cooperative survival principles for underground flooding: Vitae system based multi-agent simulation," *Expert Syst. Appl.*, vol. 83, pp. 379–395, Oct. 2017.
[16] H. Liu, B. Liu, H. Zhang, L. Li, X. Qin, and G. Zhang, "Crowd evacuation simulation approach based on navigation knowledge and two-layer control mechanism," *Inf. Sci.*, vols. 436–437, pp. 247–267, Apr. 2018.
[17] L. Crociani, G. Lämmel, and G. Vizzari, "Multi-scale simulation for crowd management: A case study in an urban scenario," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.* Cham, Switzerland: Springer, 2016, pp. 147–162.
[18] 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.
[19] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998.
[20] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. 11th Int. Conf. Mach. Learn.*, 1994, pp. 157–163.
[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," in *Proc. NIPS Workshop Deep Learn.*, 2013, pp. 1–10.
[23] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 529–541.
[24] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. 33th Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 1045–1054.
[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *Comput. Sci.*, vol. 8, no. 6, pp. 187–201, 2015.
[26] V. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.
[27] A. Lissovoi and C. Witt, "Runtime analysis of ant colony optimization on dynamic shortest path problems," *Theor. Comput. Sci.*, vol. 561, pp. 73–85, Jan. 2015.
[28] T. T. Mac, C. Copot, D. T. Tran, and R. D. Keyser, "A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization," *Appl. Soft Comput.*, vol. 59, pp. 68–76, Oct. 2017.
[29] Y. Tusi and H.-Y. Chung, "Using ABC and RRT algorithms to improve mobile robot path planning with danger degree," in *Proc. 5th Int. Conf. Future Gener. Commun. Technol. (FGCT)*, Aug. 2016, pp. 21–26.
[30] S. Wang, H. Liu, K. Gao, and J. Zhang, "A multi-species artificial bee colony algorithm and its application for crowd simulation," *IEEE Access*, vol. 7, pp. 2549–2558, 2018.
[31] A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar, "A deterministic improved Q-learning for path planning of a mobile robot," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1141–1153, Sep. 2013.
[32] T. X. Sang, T. Kiet, and N. ThiUyen, "Path finding algorithms for autonomous robots based on reinforcement learning," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 6, no. 4, pp. 2278–2293, 2017.
[33] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 1038–1044.
[34] A. I. Panov, K. S. Yakovlev, and R. Suvorov, "Grid path planning with deep reinforcement learning: Preliminary results," *Procedia Comput. Sci.*, vol. 123, pp. 347–353, Jan. 2018.
[35] D. L. Cruz and W. Yu, "Path planning of multi-agent systems in unknown environment with neural kernel smoothing and reinforcement learning," *Neurocomputing*, vol. 233, pp. 34–42, Apr. 2017.
[36] X. Lei, Z. Zhang, and P. Dong, "Dynamic path planning of unknown environment based on deep reinforcement learning," *J. Robot.*, vol. 2018, Sep. 2018, Art. no. 5781591.
[37] Z. Sui, Z. Pu, J. Yi, and X. Tan, "Path planning of multiagent constrained formation through deep reinforcement learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.
[38] O. J. Akinwande, H. Bi, and E. Gelenbe, "Managing crowds in hazards with dynamic grouping," *IEEE Access*, vol. 3, pp. 1060–1070, 2015.
[39] L. Lyu and J. Zhang, "Toward modeling emotional crowds," *IEEE Access*, vol. 6, pp. 55893–55906, 2018.
[40] H. Zhang, H. Liu, X. Qin, and B. Liu, "Modified two-layer social force model for emergency earthquake evacuation," *Phys. A, Stat. Mech. Appl.*, vol. 492, pp. 1107–1119, Feb. 2018.
[41] N. Pelechano and N. I. Badler, "Modeling crowd and trained leader behavior during building evacuation," *IEEE Comput. Graph. Appl.*, vol. 26, no. 6, pp. 80–86, Nov./Dec. 2006.
[42] M. Fachri, S. Juniastuti, S. M. S. Nugroho, and M. Hariadi, "Crowd evacuation using multi-agent system with leader-following behaviour," in *Proc. 4th Int. Conf. New Media Stud. (CONMEDIA)*, Nov. 2017, pp. 92–97.
[43] S. Juniastuti, M. Fachri, S. M. S. Nugroho, and M. Hariadi, "Crowd navigation using leader-follower algorithm based reciprocal velocity obstacles," in *Proc. Int. Symp. Electron. Smart Devices (ISESD)*, Nov. 2016, pp. 148–152.
[44] Y. Ma, R. K. K. Yuen, and E. W. M. Lee, "Effective leadership for crowd evacuation," *Phys. A, Stat. Mech. Appl.*, vol. 450, pp. 333–341, May 2016.
[45] H. Dong, X. Gao, T. Gao, X. Sun, and Q. Wang, "Crowd evacuation optimization by leader-follower model," *IFAC Proc. Vol.*, vol. 47, no. 3, pp. 12116–12121, 2014.
[46] J. A. Hartigan and M. A. Wong, "A K-means clustering algorithm," *Appl. Stat.*, vol. 28, no. 1, pp. 100–108, 1979.
[47] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn. (PMLR)*, vol. 80, 2018, pp. 5571–5580.
[48] H. Liu, B. Xu, D. Lu, and G. Zhang, "A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm," *Appl. Soft Comput.*, vol. 68, pp. 360–376, Jul. 2018.



SHANGFEI ZHENG was born in 1997. He is currently pursuing the master's degree with the College of Computer Science and Engineering, Shandong Normal University, China. His main research interests include deep reinforcement learning and evacuation simulation.

HONG LIU received the Ph.D. degree from the Chinese Academy of Sciences, in 1998. She is currently a Professor of computer science with the School of Information Science and Engineering, Shandong Normal University. She has published more than 200 refereed articles. Her main research interests include computational intelligence and cooperative design.