Postprint of Multi-Agent Collaborative Algorithm Based on Influence Degree and State Prediction

Authors: Zheng Yanbin, Ma Guangfu, Wang Linlin, Xi Pengxue

Date: 2018-06-19T00:00:00+00:00

Abstract

This study investigates the problem of multi-agent collaboration in disaster environments and proposes a multi-agent collaboration algorithm based on influence degree and state prediction. First, the algorithm processes information perceived by agents using an influence function according to the information requirements of the collaborative task. Second, it employs a prediction algorithm to forecast the subsequent state of the task and agent behaviors, and formulates collaboration strategies based on the prediction results. Finally, agents executing the collaborative task dynamically adjust their collaboration strategies according to action effects and trigger conditions. To validate the effectiveness of the algorithm, a simulation platform was constructed in Unity3D to compare the convergence rate, number of rescued individuals, and overall score across different collaboration algorithms. The results demonstrate that the proposed algorithm achieves faster convergence, a greater number of rescues, and optimal overall scores, can effectively guide inter-agent collaboration, and provides theoretical support for the formulation of practical rescue collaboration strategies.

Full Text

Preamble

Multi-Agent Cooperation Algorithm Based on Influence Degree and State Prediction

Zheng Yanbin, Ma Guangfu, Wang Linlin, Xi Pengxue

(College of Computer & Information Engineering, Henan Normal University, Xinxiang, Henan 453007, China)

Abstract: This paper addresses multi-agent cooperation problems in disaster environments and proposes a multi-agent cooperation algorithm based on

influence degree and state prediction. First, the algorithm processes information perceived by agents using an influence degree function according to the information requirements of collaborative tasks. Second, a prediction algorithm forecasts the subsequent states of tasks and agent behaviors, and formulates collaboration strategies based on these predictions. Finally, agents executing collaborative tasks dynamically adjust their cooperation strategies according to action effects and trigger conditions. To verify the algorithm' s effectiveness, a simulation platform was built in Unity3D to compare the convergence rate, number of rescued individuals, and overall scores across different cooperation algorithms. The results demonstrate that the proposed algorithm achieves faster convergence, rescues more individuals, and obtains optimal overall scores. It can effectively guide inter-agent cooperation and provide theoretical support for developing practical rescue coordination strategies.

Keywords: multi-agent; information filtering; state prediction; collaborative strategy; action effect

0 Introduction

Multi-agent systems (MAS) represent a hot topic in distributed artificial intelligence research, with multi-agent cooperation in dynamic environments being one of the core challenges [**?**, **?**, **?**]. For instance, [**?**] proposed a multi-agent decentralized deployment strategy for harbor protection that enables agent scheduling, deployment, and missile interception despite noisy information and intermittent communication, but this algorithm exhibits low efficiency in processing rapidly changing information. [**?**] employed Monte Carlo tree search for UAV cooperation, constructing relationship graphs based on neighbor node values and formulating joint actions through reward functions; however, when node relationships change abruptly, algorithm performance degrades and may prevent task completion. [**?**] introduced a prediction system for multi-agent cooperation that combines genetic algorithms with evolutionary theory to predict agent evolution based on environmental changes, though it fails when environmental mutations occur. [**?**] identified and predicted individual intentions and behaviors to derive group joint intentions and actions, providing support for strategy formulation but neglecting environmental impacts on individual behaviors. Domestic researchers such as [**?**] improved MA-PDDL by adding continuous planning elements, proposing a continuous planning-based cooperation algorithm that prevents task failure due to environmental changes, but requires updating the system environment before each action, incurring excessive time costs. [**?**, **?**] processed and synthesized environmental information to obtain comprehensive target states for strategy formulation, but this fusion process involves massive computational complexity and difficulty finding global optimal solutions in dynamic environments. [**?**] used particle filtering to process noisy information and obtain target states, but high algorithmic complexity directly affects strategy formulation. [**?**] proposed a state prediction-based dynamic multi-agent cooperation algorithm that adds urgency to information for processing and prediction,

reducing information volume and improving filtering efficiency; however, individuals may discard important low-urgency information while receiving high-urgency information, affecting strategy correctness.

Disaster environments are complex and volatile, characterized by strong suddenness, massive destructive power, susceptibility to secondary disasters, far-reaching impacts, and difficult rescue operations. These features render the aforementioned algorithms inadequate for post-disaster multi-agent cooperative rescue. Therefore, this paper proposes a multi-agent cooperation algorithm combining influence degree and state prediction. The algorithm first uses an influence degree function to process perceived information according to task requirements, obtaining task-relevant information while ignoring irrelevant data. Second, it introduces a state prediction algorithm to forecast task development trends and individual behaviors, formulating cooperation strategies based on predictions. Finally, cooperating agents select appropriate actions according to strategies and task requirements, dynamically adjusting cooperation strategies based on action effects to enable dynamic adaptation, ensuring system cooperation efficiency and accuracy. Simulation experiments demonstrate that the algorithm can integrate environmental, predictive, and planning information to complete post-disaster cooperative rescue tasks, providing theoretical and technical support for practical cooperation decision-making.

1 Information Filtering

Disaster environments typically contain numerous dynamic elements including collapsed buildings, fires, casualties, and blocked roads, characterized by diverse information types, high noise, high dispersion, and inconsistent processing methods. Moreover, the impact of information varies depending on its relationship with cooperation tasks. Therefore, it is essential to filter information closely related to rescue operations from complex data. Additionally, cooperating agents have limited perception and reception capabilities, necessitating extraction of real-time, efficient information while ignoring irrelevant data to reduce inter-agent communication overhead and improve cooperation efficiency. This paper employs an influence degree function to filter cooperation information.

1.1 Influence Degree Function

The influence degree function consists of two layers, as shown in [Figure 1: see original paper]. Equations (1)-(3) constitute the first layer, which determines rescue task type information such as disaster category, affected area, and number of casualties. The second layer is a supplementary information layer composed of many disaster classes, each containing numerous content-specific information filtering functions. Its role is to call the corresponding disaster class based on `dis` from Equation (1) to supplement task-relevant information.

dis : Disaster type including earthquakes, mudslides, floods, fires, and other natural disasters or public emerge

α : Disaster level constant; larger values indicate greater damage

S : Disaster-affected area

r : Distance from disaster source (x_0, y_0) to farthest affected point (x_i, y_i)

N : Damage value to individuals or buildings

P : Initial damage value

F : Coefficient indicating impact from disaster source; values closer to 1 indicate greater hazard

f : Adjustment factor

$$D_{dis} = \alpha \times 2 \tag{1}$$

$$S = \pi r^2 \tag{2}$$

$$N = P \times F \times \alpha \tag{3}$$

Since different disaster types have distinct characteristics and hazards, the required cooperation rescue information varies. The second layer supplements detailed task information. Taking fire, casualty, and road blockage categories as examples:

Fire Information: When agents execute firefighting tasks, they obtain specific information using Equations (4)-(6).

$$w_f = \text{area} \times R \times \text{Mat} \times M \times K \times K_s \times \varphi \tag{4}$$

where w_f represents fire spread speed, R is fuel ignitability, Mat is fuel initial mass, M includes temperature, humidity, building structure, and casualty distribution at fire points, and φ is ventilation factor.

$$C(x, y, z, t) = \frac{Q}{(2\pi)^{3/2}\sigma_x\sigma_y\sigma_z} \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} + \frac{(z-H)^2}{\sigma_z^2}\right)\right] \tag{5}$$

This equation yields toxic gas information: air toxic gas concentration C, gas height H, disaster source gas generation quantity Q, propagation time t, and diffusion in x, y, z directions.

$$\text{TL} = \int_0^t C^n \, dt \tag{6}$$

Equation (6) obtains casualty toxic gas hazard information: toxic load TL, surrounding gas concentration C_i, exposure time t, where n is a power exponent with different values for different gases.

Casualty Information: When discovering casualties, Equations (7)-(8) retrieve relevant information.

$$H(t) = H_0 - \int_0^t d(r)\, dt \tag{7}$$

where $H(t)$ obtains specific casualty health information: initial health value H_0, initial disaster damage value $d(r)$, and health decline rate $r(t)$.

$$r(t) = r_0 + k \times \ln(1+t) + \sum_{i=1}^{n} b_i \times t \times \mathrm{dis}_i \tag{8}$$

Equation (8) obtains $r(t)$ specifics: casualty type b_t, k_{dis} and l_{dis} as proportion factors in different disaster types, n as Gaussian distribution parameter.

Road Blockage Information: Agents receiving cooperation tasks call the road blockage class using Equations (9)-(11) to obtain obstacle information and expected arrival time.

$$m_i = N_{b_i} \times T_{b_i} \tag{9}$$

Equation (9) obtains obstacle type b_i, quantity N_{b_i}, and clearance time T_{b_i}.

$$T_i = \frac{S_i}{V} \tag{10}$$

where S_i is the optimal path length for clearance vehicles to reach obstacles, and V is average clearance vehicle speed.

Assuming rescue vehicles move at constant speed V, comparing agent arrival time t_i and clearance vehicle arrival time T_i yields obstacle passage time:

$$t_i - T_i = \begin{cases} t_i - T_i & \text{if } t_i > T_i \\ 0 & \text{if } t_i = T_i \\ T_i - t_i + \frac{N_{b_i}}{V} & \text{if } t_i < T_i \end{cases} \tag{11}$$

When $t_i > T_i$, passage time is $t_i - T_i$; when $t_i = T_i$, passage time is 0; when $t_i < T_i$, passage time is $T_i - t_i + \frac{N_{b_i}}{V}$.

$$T_c = \sum_{j=1}^{m} T_j \tag{12}$$

Equation (12) calculates rescue vehicle arrival time at target points, where j is the number of obstacles traversed.

1.2 Information Processing

While the influence degree function can filter task-relevant information, multiple similar tasks may exist simultaneously in the scene, generating similar information. Without processing, this confuses receiving agents; receiving incorrect information leads to cooperation failure, while receiving all information increases communication pressure and time consumption, and conflicting prediction results prevent correct payoff values and dynamic strategy modification. This paper proposes processing information using task IDs combined with cooperation information to create unique task-corresponding information. Agents only need to select information matching their task ID.

Task IDs consist of three parts: time, location, and random number. Time refers to the system time when the search agent first discovers the task; location is the task' s position in the scene; the random number distinguishes different tasks at the same time and location. Senders transmit cooperation information with its task ID, and receivers only accept information matching their task ID.

1.3 Filtering Algorithm

The influence degree function filters information based on its impact on current tasks, extracting scene information relevant to rescue tasks while ignoring invalid data. The main filtering algorithm proceeds as follows:

1. Search the scene, discover tasks, obtain information via sensors;
2. Call first-layer formulas (e.g., Equation (1)) to obtain disaster type `dis` and level α, call Equation (2) for affected area S, call Equation (3) for casualty/damage value N; if $\alpha > \lambda$ or ($\alpha < \lambda$ and $N > 0$), proceed to step 3, otherwise return to step 1;
3. Call second-layer disaster class corresponding to `dis` for task details (e.g., Equations (4)-(6) for fire information, Equations (7)-(8) for casualty information), proceed to step 4;
4. Call Equations (9)-(12) for road blockage details, proceed to step 5;
5. Process and save filtered information.

Using the influence degree function for cooperation information filtering improves information accuracy while reducing inter-agent communication volume and system communication pressure.

2 State Prediction

To ensure cooperation strategy and individual action correctness, the algorithm must accurately predict task development processes. While individual behavior prediction techniques are mature, effective methods for predicting task and environmental development trends are lacking. This paper proposes a new prediction algorithm that first uses cached data for preliminary predictions of task trends, individual behaviors, and environmental changes, then formulates cooperation strategies based on predictions. Second, it calculates action effects to determine impacts on tasks, environment, and other agents, adjusting cooperation strategies accordingly.

Cached information X_n and current system state U_m serve as inputs to the state prediction function $f(X_n, U_m, \delta)$, predicting subsequent states T_{c+1} at time $t+1$. The main prediction function formulas are:

$$T_{c+1} = f(X_n, U_m) \pm \delta \times f(X_n, U_m) \tag{13}$$

$$p_{t-1} = f(x_{t-1}, u_{t-1}) \tag{14}$$

$$p'_{t-1} = f(x_{t-1}, u_{t-1}) \tag{15}$$

$$E_k = \frac{1}{k} \sum_{i=1}^{k} |p'_{t-i} - p_{t-i}| \tag{16}$$

$$\delta = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (p'_{t-i} - p_{t-i})^2} \tag{17}$$

where p'_{t+1} is the system prediction value, p_{t+1} is the measured value at time $t+1$, E_k represents the average error of the previous k moments, and δ is the standard error after removing $N-k$ large error values. Prediction functions f have different definitions for different events or individuals.

The state prediction algorithm proceeds as:

1. Input cached information X_n and U_m;
2. Use Equations (14) and (15) to obtain previous $t-k$ moment predictions and measurements, calculating error values $|p' - p|$;
3. Calculate average error E_k of previous k moments;
4. Remove predictions with large deviations from average error;
5. Calculate standard deviation δ of remaining N error values;
6. Call Equation (13) to obtain state T_{c+1} (an approximate state within a certain error range).

Removing large-error values prevents abnormal moment predictions from affecting results. The state prediction algorithm obtains subsequent states of individuals, tasks, and environment. Since environment, individual actions, and behaviors all affect cooperation, correct and efficient cooperation strategies are essential for agents to make reasonable actions and ensure smooth cooperation.

3 Cooperation Strategy

3.1 Strategy Formulation

Search agents discovering tasks first use the information filtering algorithm to obtain task-relevant information and save it, then use the prediction algorithm to obtain task subsequent states. Search agents transmit filtered information, current states, and predicted states to the commander agent, which determines the cooperation team, formulates preliminary cooperation strategies, and sends them to team commanders and cooperating agents. Agents receiving cooperation requests select appropriate actions based on current information, evaluate action effects after execution, and determine whether trigger conditions are met. The cooperation flowchart is shown in [Figure 2: see original paper].

Trigger classes set corresponding trigger conditions for different problems. Agents adjust cooperation strategies, actions, and teams when conditions are met during task execution. Trigger conditions fall into three categories: (1) planning for problems en route to task points such as extended clearance time, accidental damage, or vehicle breakdown; (2) addressing cooperation process issues such as task volume changes, environmental changes, and inter-agent conflicts; (3) resolving resource, time, and space conflicts between teams.

3.2 Cost Function

Idle individuals calculate expected costs for task completion and decide whether to accept cooperation tasks based on Cost values. The cost function is defined in Equation (13):

$$\text{Cost} = H_i(t) - \sum_{j=1}^{n} B_{ij}(t) \tag{18}$$

where $H_i(t)$ is the current health value of agent i determining task completion capability, and $B_{ij}(t)$ is the expected cost value for completing cooperation task j. When Cost < 20, the agent suffers serious damage and abandons the cooperation task to replenish capability; when Cost > 20, the agent accepts the task and joins the cooperation team.

$$B_{ij}(t) = s(t) + \sum_{i=1}^{n} x_{ij} \tag{19}$$

where $s(t)$ represents sudden damage during action execution, and x_{ij} represents actual cost value after completing specified actions.

$$R = \begin{cases} -2 & \text{if agent deliberately abandons task} \\ 0 & \text{if agent exits task due to injury} \\ 1 & \text{if agent completes task} \end{cases} \tag{20}$$

R serves as the reward value returned to agents based on task completion. The commander agent selects the idle individual with maximum R as team commander.

3.3 Action Effects

After completing actions, individuals must evaluate effects and modify cooperation strategies accordingly. Actions are described using triples.

Definition 1: Action O is a triple $\langle A, Q, E \rangle$, where A is the acting agent, Q is the precondition set, and E is the action benefit.

Definition 2: Precondition Q is a quintuple $\langle I, O, S, G, U \rangle$, where I is the initial state T_c, O is the agent' s action set, S is action constraints, G is the new state after executing selected actions, and U is the predicted state T_{c+1}.

Action effect evaluation has two parts: using Equation (20) to calculate actual action cost, primarily comparing $|G - U|$ with standard error δ to determine impacts on tasks, other agents, and environment, then modifying strategies accordingly. The main action evaluation algorithm:

1. Select action O based on I, S, and strategy;
2. If $|G - U| > \delta$, abandon current task and feedback information to modify strategy; otherwise obtain current state G;
3. When $|G - U| > \delta$, obtain specific impact information;
4. If intra-team adjustment is needed, perform adjustment and plan new actions; if action set meets requirements, proceed; otherwise modify strategy and update action set;
5. If task incomplete, select new action; otherwise finish.

This algorithm evaluates action effects while ensuring action set correctness, enabling dynamic cooperation strategy adjustment to avoid slow updates affecting tasks or agents, improving task execution efficiency.

4 Cooperation Rescue Algorithm

Due to the complex and volatile nature of disaster rescue tasks and limited individual agent capabilities, multiple agents must cooperate to minimize disaster losses. An efficient rescue algorithm supporting inter-agent cooperation is crucial. The rescue cooperation algorithm in this system proceeds as:

1. Search agents move through the scene searching for tasks;
2. Upon discovery, obtain task and environment information via sensors;
3. Use influence degree function first layer to get disaster type `dis` and level α; if no cooperation needed, ignore and return to step 1; otherwise proceed;
4. Determine affected area S, casualty numbers, and damaged buildings;
5. Use second-layer disaster class corresponding to `dis` for task details;
6. Process and save filtered information to cache;
7. State prediction algorithm uses cached information to predict task, agent, and environment states, sending results to commander agent;
8. Commander agent formulates cooperation strategy, determines team members and commanders, and sends to cooperating agents; updates strategy when receiving modification information;
9. Send cooperation information to team individuals and commanders;
10. Individuals receive cooperation information and calculate expected costs using Equation (19); if Cost < 20, abandon task and return to commander; otherwise join team;
11. Team commander receives strategy and sends to task-accepting individuals;
12. Agents select actions from action set and execute; after completion, evaluate action effects; if impact is small, check trigger conditions; otherwise perform intra-team adjustment;
13. If trigger conditions met, adjust strategy; otherwise continue;
14. Task-completing agents obtain reward R and send current state to commander; if end conditions met, finish; otherwise accept new tasks.

This cooperation algorithm first uses influence degree for information processing, reducing system communication volume while improving information accuracy. Second, the state prediction function uses cached information to obtain subsequent system states and formulate cooperation strategies, ensuring correctness. Finally, cooperating individuals dynamically modify strategies based on benefit values, guaranteeing algorithm convergence and robustness while improving agent decision-making.

5 Experimental Simulation and Analysis

5.2 Agent Modeling

To improve simulation efficiency, agent functions must be clearly defined, modeled, and behavior-constrained. The commander agent' s main functions: (a) allocate tasks using relevant states and allocation algorithms to determine required agent types and quantities; (b) formulate cooperation strategies, determine team commanders, and send cooperation information; (c) update strategies promptly when receiving agent feedback; (d) store status information of agents, environment, and rescue vehicles.

Team commanders primarily communicate with the commander, receive and transmit cooperation strategies, and handle intra-team planning.

To improve data collection speed and quality, relevant sensors are deployed on search agents. Upon task discovery, they promptly perceive information, reducing acquisition time. Search agent functions: (a) carry sensors to perceive environment, other agents, and task information; (b) use influence degree function to filter and process cooperation information; (c) use functions to predict task and individual subsequent states; (d) receive and transmit information.

Fire agents handle firefighting and rescue, police agents command clearance vehicles and maintain order, doctor agents treat casualties, and communication agents handle information transmission. Agents have three behavioral states: idle, busy, and task-executing, with 8 movement directions at 0.5 m/s. Health decreases by 20 upon collision and varies based on fire, burial, gas, and other hazards. Agents can modify action sets and transform functions according to task requirements during cooperation.

5.3 Simulation Results

In dynamic environments, agents lack global dynamic and static environment information before cooperation and cannot obtain dynamic obstacle movement information during cooperation. Under identical experimental backgrounds and algorithm parameters, the first experiment compares: agents cooperating using algorithm from [**?**], agents using algorithm from [**?**], and agents using the proposed algorithm. Agent final objectives are consistent, but algorithm convergence speeds differ, as shown in [Figure 4: see original paper].

The simulation scenario is 500m $\times$ 500m ([Figure 3: see original paper]). Model information and functions are listed in . Scene details: yellow area D is a 50m $\times$ 50m shelter for casualties and supplies; black lines represent 4m-wide roads; red circles indicate static obstacles with color intensity representing quantity and type; black blocks represent mobile obstacles that can move one grid randomly in four directions per time step without collision; disaster type `dis` and level α jointly determine building damage; blue areas represent casualties with different colors indicating injury types and severity; rescue vehicles move at 5 m/s; each ambulance carries one doctor agent (adjustable); fire trucks carry 6 firefighting agents with 5000L water and 1000L foam (range 65m and 60m), extinguishing 2 m^2/s over 100 m^2; aerial ladder trucks reach 100m height with 20m working diameter, rescuing 4 adults per platform operation; clearance vehicles remove road obstacles at 3 m^3/s with driver and commander agents.

Due to single-run randomness, 100-run averages are used. Results show the proposed algorithm converges faster than the other two, attributable to information processing and state prediction during cooperation. The algorithm from [**?**] calculates joint actions but lacks effective state prediction, affecting convergence when tasks mutate. The algorithm from [**?**] includes state prediction for obstacles, individuals, and tasks but suffers low information processing efficiency, reducing convergence and completion rates. The proposed algorithm combines improved information processing with state prediction, significantly improving

convergence speed by processing only task-relevant information and using it for state prediction and strategy formulation, enabling rapid convergence path identification after interference.

A second experiment under identical information backgrounds compares: agents using algorithm from [**?**], agents using algorithm from [**?**], agents using algorithm from [**?**], and agents using the proposed algorithm. Rescue numbers are shown in [Figure 5: see original paper] and overall scores in [Figure 6: see original paper].

The proposed algorithm rescues more individuals than [**?**] because it obtains more comprehensive cooperation information, improving prediction accuracy, and uses action evaluation with trigger classes for dynamic strategy adjustment, ensuring task completion. Algorithm [**?**] suffers from high information processing complexity, reducing cooperation efficiency. Algorithm [**?**] struggles to obtain optimal cooperation strategies, resulting in low efficiency. [Figure 6: see original paper] further demonstrates the proposed algorithm' s superior performance.

6 Conclusion

This paper proposes a multi-agent cooperative rescue algorithm combining influence degree function and state prediction. First, the information filtering algorithm reduces inter-agent communication while improving information accuracy. Second, filtered information predicts subsequent states to formulate cooperation strategies, ensuring correctness. Finally, cooperating individuals dynamically modify strategies based on benefit values, guaranteeing convergence, robustness, and improved decision-making. Unity-based simulations validate the algorithm and compare it with four other algorithms. Results demonstrate superior performance, showing the algorithm can efficiently complete cooperative rescue tasks and support rescue decision-making.

References

[1] Gerhard W. Multi-agent systems: a modern approach to distributed artificial intelligence [M]. Boston: MIT Press, 1999.

[2] Jiao W P. Multi-agent cooperation via reasoning about the behavior of other [J]. Computational Intelligence, 2010, 26 (1): 57-83.

[3] Semsa-Kazerooni E, Khorasani K. A game theory approach to multi-Agent team cooperation [C]// Proc. of the American Control Conference on Hyatt Regency Riverfront. 2009: 10-12.

[4] Suruz M, Bao N, Davide S. Nonuniform deployment of autonomous agents in harbor-like environments [J]. Unmanned Systems, 2014, 2 (4): 377-389.

[5] Baker C, Ramchurn G, Teacy L, et al. Factored Monte-Carlo tree search for coordinating UAVs in disaster response [C]// Distributed and Multi-Agent

Planning. 2016.

[6] Provenzi M D O, Götz M. Activity prediction in a smart environment using Bayesian network and multi-agent system [C]// Computing Systems Engineering. 2017: 140-146.

[7] Saldana D, Assunçao R, Hsieh M A, et al. Cooperative prediction of time-varying boundaries with a team of robots [C]// Proc of International Symposium on Multi-Robot and Multi-Agent Systems. 2018.

[8] Wu Kun, Liu Wei, Li Shuang, et al. A multi-agent cooperation method in dynamic environment [J]. Journal of Wuhan Institute of Technology, 2017, 39 (2): 186-192.

[9] Zhou F, Xian M, Xiao S P. Research on co-operative information fusion system based on multi-agent technology [J]. Command Control & Simulation, 2006, 28 (4): 13-16.

[10] Wu Zhixin. A multi-agent based information fusion system [J]. Computer and Digital Engineering, 2013, 41 (7): 1074-1077.

[11] Shen Jie, Liang Zhiwei, Liu Juan, et al. Multi-agent collaborative approaches in RCRSS [C]// Chinese Control & Decision Conference, 2013.

[12] Peng Jun, Liu Ya, Wu Min, et al. Dynamic Cooperation Algorithm in Multi-Agent System Based on State Prediction [J]. Journal of System Simulation, 2008, 20 (20): 5511-5515.

[13] Wang Ying. The injury area of the toxic gas instant leaking based on Gaussian model [J]. Fire Science and Technology, 2010, 29 (11): 1002-1004.

[14] Zheng Yanbin, An Deyu, Li Na, et al. Ant Colony Algorithm for Path Planning in Fire Environment [J]. Journal of Shanxi University: Natural Science, 2017, 40 (4): 690-701.

[15] Wei Zhipeng. Heterogeneous Multi-Agent cooperation strategy in RoboCup Rescue [D]. Nanjing: Nanjing University of Posts and Telecommunications, 2016.

Note: Figure translations are in progress. See original paper for figures.