2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

Demonstration of a Distributed Cooperative Strategy Planning Algorithm in a Multi-Agent System with Multiple Objectives

Yuta Takemoto[1]

Abstract— **Robot-based services are increasingly being integrated into everyday life. In the diverse environments in which humans live, systems are expected to respond to dynamically changing surroundings by enabling distributed control and cooperation. However, in spaces where various robots coexist, it is practically challenging to ensure that all robots in the space can communicate all necessary information with one another. This poses challenges related to communication failures, the need for unified communication standards, and legal regulations for robots that do not comply with these standards. Furthermore, from the perspective of coexistence with robots in urban areas, there are many entities, such as humans, birds, and cats, with which standardized communication is not feasible. When introducing robots into environments such as cities, it is necessary to share pathways such as roads, sidewalks, and corridors, as well as public infrastructure such as buses and trains, with non-communicating entities. Therefore, in the social application of such robots, it is desirable to construct a system that can operate on the premise that it cannot obtain complete information about other robots or living organisms, and it is necessary to consider a system that can continue its mission even when information about other robots is ambiguous. In this study, we investigated a distributed cooperative strategy planning algorithm for a multi-agent system with multiple tasks of patrolling three designated points specified by coordinates in an unknown space. The algorithm assigns tasks to each agent and enables the system to continue its mission even when agents cannot accurately obtain information about one another. We also verified the effectiveness of this algorithm using actual robots and report the results.**

I. INTRODUCTION

Robot-based services are increasingly being integrated into everyday life, with service robots for indoor and outdoor transportation and assistance in daily life at the forefront. Such robots are expected to address the declining workforce due to the aging population and low birthrate, as well as to serve as alternatives for dangerous tasks. However, most robots are designed for limited and constrained environments, making it difficult for them to flexibly respond to changes in their surroundings and perform tasks in the diverse environments where humans live. To address the dynamic changes in the environment surrounding robots, systems must be capable of distributed control and dynamic reorganization [1]. Additionally, multi-agent systems can easily scale up or down depending on the mission, making them adaptable to a wide range of applications [2], and enabling them to complete missions more quickly and efficiently [3].Furthermore, in multi-agent systems, even in situations where it is difficult for a single robot to perform a task, such as transporting heavy objects or searching over a wide area, multiple robots can cooperate to accomplish the task. In such situations, rapid and accurate information exchange between robots is desirable, but in real-world environments, communication failures must be considered, and research on communication methods between robots is underway [4]. Moreover, there are many challenges in using robots to explore unknown environments, such as reconnaissance [5] and cleaning [6]. Discussions are also underway regarding robot systems that can exchange information with each other and operate autonomously and as a team in these unknown environments[7].

In spaces where diverse robots coexist, it is difficult for all robots in the space to communicate all necessary information to each other. This is because, in addition to the communication failure issues mentioned earlier, there are challenges in terms of the standardization of communication protocols and legal regulations for robots that do not comply with these standards. Furthermore, from the perspective of coexistence with robots in urban areas, there are many entities, such as humans, birds, and cats, with which standardized communication with robots is not practically possible. Fig. 1 shows a situation where decisions must be made based on incomplete information. In this example, when A attempts to grasp the surrounding information, A and B are the same type of agent, so the communication standards are unified, and A can obtain B's current position and destination information. However, some communication standards are not compatible with C, so while its current location can be obtained, its destination information cannot be obtained. Furthermore, since D is a human, A can detect its presence through observation, but cannot obtain information through communication, resulting in low accuracy of its current location and unknown destination information. A must make decisions based on this diverse information with varying degrees of accuracy and take actions such as yielding the way or cooperating. When introducing robots into environments such as cities, it is necessary to share information about communication-impossible targets, roadways, sidewalks, corridors, and public infrastructure such as buses and trains. Therefore, in the social application of such robots, it is desirable to construct a system based on the assumption that it is not possible to obtain complete information about other machines or living organisms, and it is necessary to consider a system that can continue missions even under conditions where information about other robots is ambiguous.

In this study, we investigated a distributed cooperative strategy planning algorithm for a multi-agent system with multiple tasks of patrolling three designated points specified by coordinates in an unknown space.

[1]Yuta Takemoto is with Information Technology R&D Center, Mitsubishi Electric Corporation, 5-1-1 Ofuna, Kamakura Kanagawa Japan (e-mail: takemoto.yuta@dn.mitsubishielectric.co.jp).

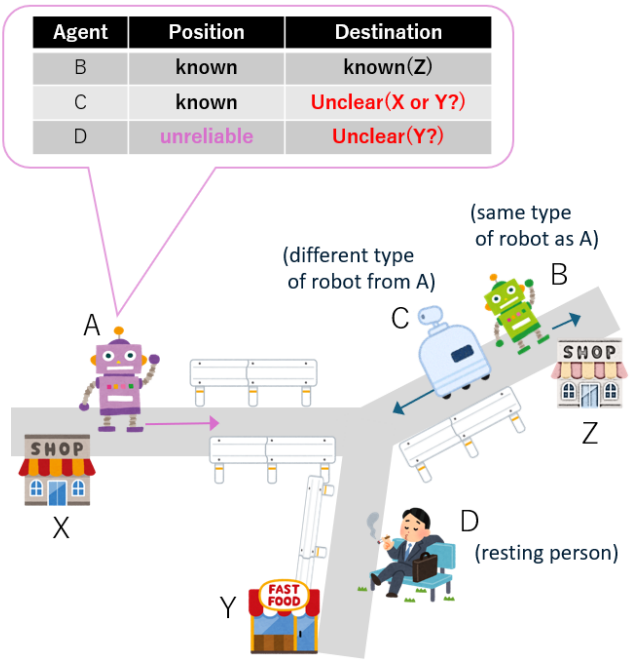

Fig. 1 Situations where decisions must be made based on incomplete information

The algorithm assigns tasks to each agent and enables the system to continue the mission even when agents cannot accurately obtain information about other agents. We also verified the effectiveness of this algorithm using actual robots and report the results.

II. System Architecture and Algorithms

The process flow of the system discussed in this paper is shown in Fig. 2. Actions are selected based on the strategy of the own agent, and then executed. A strategy refers to a policy for achieving a mission, such as whether to take a roundabout route or a direct route when heading to a target location. Actions are specific behaviors performed by agents, such as movement or sensing. The results of actions are used for estimation by the estimator. The estimator estimates information about the agent itself, such as the presence of obstacles and its own position, as well as information about the surrounding environment. It also estimates information about other agents, such as their positions, speeds, and objectives. If high-probability information obtained from communication with other agents or higher-level control systems is available, this information can be utilized. The strategy selection algorithm uses this estimated information to refer to its own strategy list and the assumed strategy list of other agents, and selects the strategy that maximizes the overall value of the system. By repeating this process, each agent participating in distributed coordination can continue to take actions that maximize the overall value of the system.

The calculation of strategic evaluation values is shown in Fig. 3. The self-agent possesses its own information and estimated values of other agents' information. The self-agent's own information includes its position coordinates, velocity, objective, strategy, and environmental information, while estimated values include information about other agents and

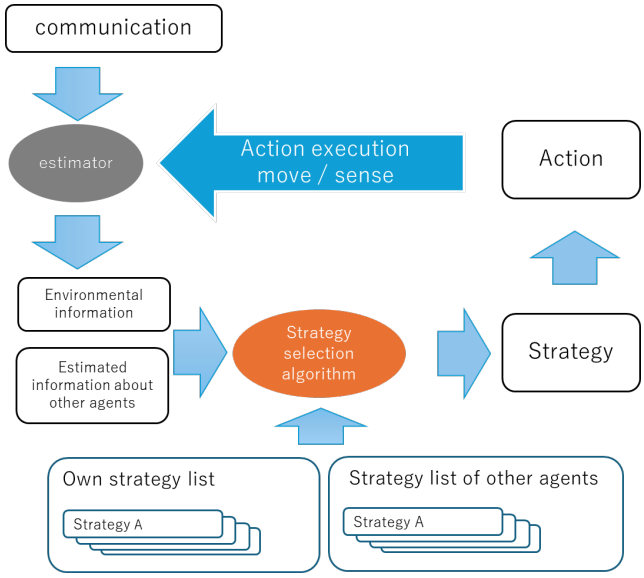

Fig. 2 Process flow of the system

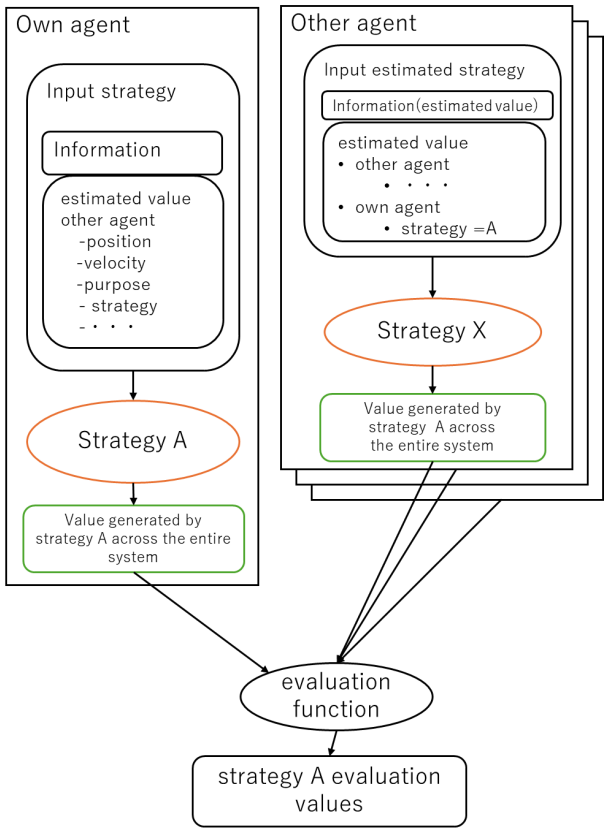

Fig. 3 Calculation of strategic evaluation values

their broader environment. Other agents' information is also estimated, including their positions, velocities, objectives, and strategies. Based on this information, when the self-agent selects a strategy A, it calculates the value that strategy A will generate for the entire system. Similarly, estimates of strategies are made for other agents. The information used for estimation includes the position, velocity, objective, strategy, and environmental information of other agents. If there is

sufficient computational capacity, other agents can also estimate other agents. At this time, the strategies decided by other agents are estimated based on the assumption that the self-agent will adopt the strategy A it has decided. Based on the estimation results, it is determined that other agents have selected strategy X, and the value generated by strategy X for the entire system is calculated. This process is executed for all agents in the system, and the results are combined using an evaluation function to determine the evaluation value of the own agent when it adopts strategy A. Each agent evaluates the strategies it can adopt based on the information it can obtain and the estimated information, and selects the strategy, thereby constituting a distributed cooperative system.

We describe the calculation of value. The expected value of the value generated by agent n when the agent selects a strategy is as follows:

$$v_n = \sum_d \{g_{n,d} + s_{n,d}\} \quad (1)$$

where g is the actual value generated by agent n's task d, and s is the expected value of the value generated by agent n for task d when the agent adopts a strategy.

The evaluation value of whether the strategy taken by agent n was effective is as follows:

$$R_n = v_n - v_{n-1} \quad (2)$$

The overall evaluation value V of the system expected when Agent x performs Strategy A, as seen from agent x, is as follows:

$$V_{1,A} = p \cdot v_{all} - q \cdot std_{agt} - r \cdot std_{task} + s \cdot R_n \quad (3)$$

$$v_{all} = \sum_n v_n \quad (4)$$

$$std_{agt} = \sqrt{\frac{1}{n}\sum_n (v_n - \bar{v})^2} \quad (5)$$

$$std_{task} = \sqrt{\frac{1}{d}\sum_d \left(g_{x,d} + s_{x,d} - \overline{(g+s)}\right)^2} \quad (6)$$

where p, q, r, and s are constants, v_{all} is the expected value of the total value comprising all agents in the system, std_{agt} is the standard deviation of the value generated between agents, and std_{task} is the standard deviation of the value generated when agent x performs different tasks.

This evaluation function is just one example and can be freely configured. For example, in missions where it is not necessary to minimize the standard deviation, these terms can be omitted.

III. Demonstration Environment

The effectiveness of the algorithm discussed in this paper was confirmed through actual operation. The agent specifications and field specifications are described below.

A. Agent Specifications

The appearance of the agent is shown in Fig. 4. The size is 125 × 75 × 130 mm, and the weight is approximately 400 g. A colored ball is mounted on the upper part for the identification of the agent. Fig. 5 shows the side view of the agent. It has a differential two-wheel configuration using DC motors and performs directional changes using spin turns.

Fig. 6 shows the agent configuration. The agent consists of a real-time control system and an application execution system. The real-time control system is implemented using Free RTOS and handles tasks such as sensing wall information, sensing with gyro sensor, and controlling the movement motors. The application execution system runs on Ubuntu using ROS2 and controls the real-time control system via RPC (Remote Procedure Call). This system can communicate with external devices using Wi-Fi on a Raspberry Pi 4B.

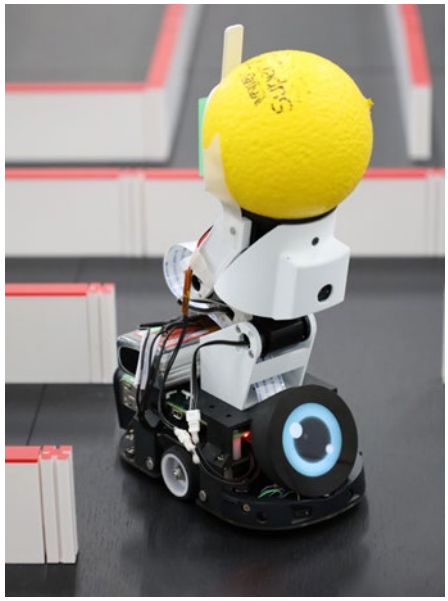

Fig. 4 Appearance of the agent

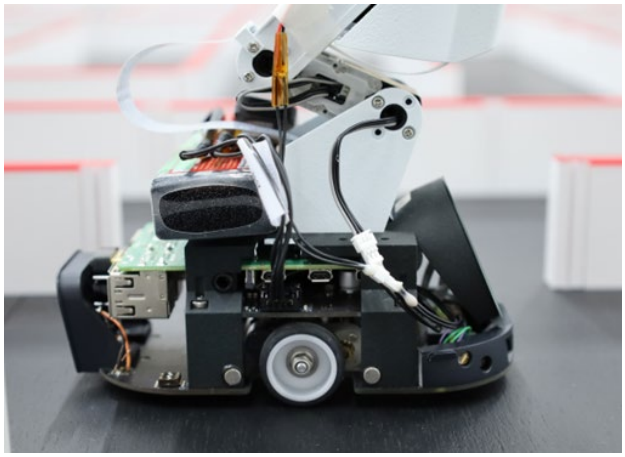

Fig. 5 Side view of the agent

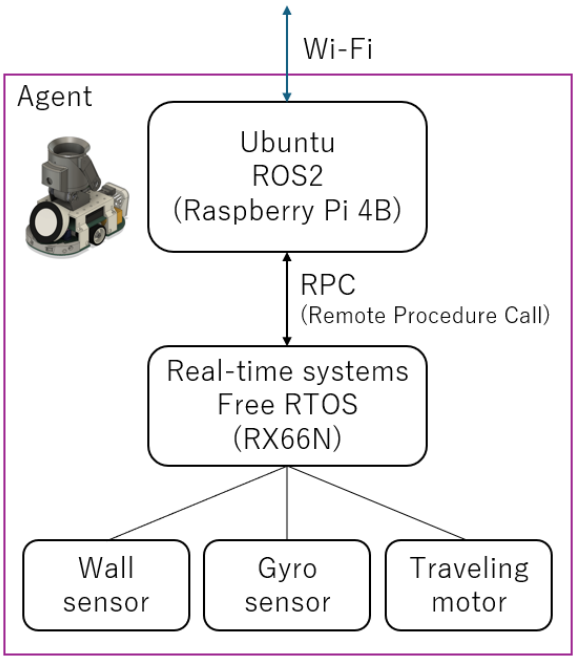

Fig. 6 Agent configuration

B. Test field specifications

The top view of the test field is shown in Fig. 7. There is a black base, with walls and pillars installed on top of it. The walls and pillars are white on the sides and red on the top. The height of the walls is 50 mm, and sensors attached to the front of the agent can detect the presence of walls and the distance to them. The pillars are installed at intervals of 180 mm, and walls can be freely installed between adjacent pillars, allowing the structure of passable areas to be changed. Fig. 8 shows the agents installed on the field. The agents are too small to pass each other in the passageways. Therefore, deadlocks and path collisions are likely to occur. At the center of each section where the distance from nearby columns is the same, agents can perform spin turns regardless of the presence of walls. No information regarding position or orientation is sent from the field to the agents, so the agents must estimate their own positions based on information about walls and other obstacles using their own functions.

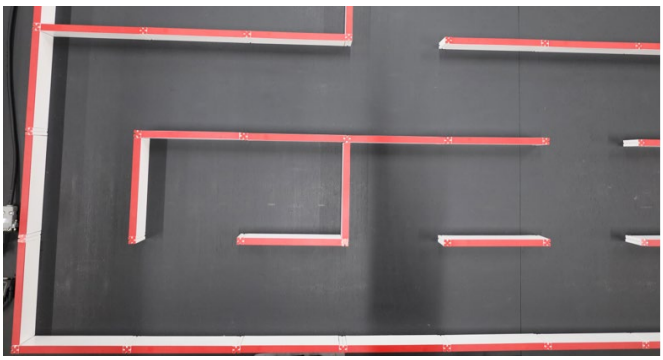

Fig. 7 The top view of the test field

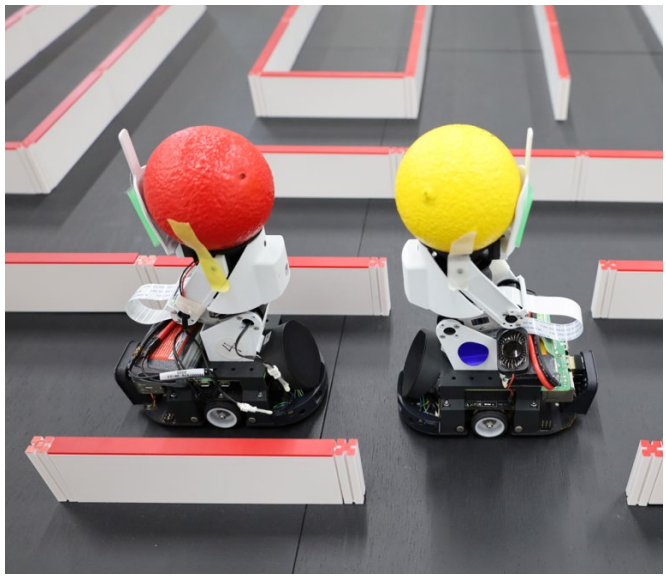

Fig. 8 Agents on the field

IV. Distributed Coordination Algorithm Demonstration Configuration

To verify the validity of the algorithm, its operation is confirmed using actual equipment. Fig. 9 shows the demonstration configuration. The processing PC is connected to a Wi-Fi router, which is connected to three agents. The algorithm is executed on the processing PC, but since it is executed in parallel processes, the algorithm is executed independently by agents Blue, Red, and Yellow. The operation is also performed assuming a situation where all information cannot be exchanged via communication. The information recognized by each agent will be described later.

Fig. 10 shows the test field. An 8×8 field is used, and the paths within the field are not known. The agents are not aware of the size of the field and move assuming an area larger than 8×8. There are three agents, each carrying a blue, red, or yellow ball for identification. In addition, target locations are set at the four corners of the field, with three destinations for each color. The value for each agent is determined by the number of times it visits destinations of the same color as itself, and the overall value of the system is maximized by maximizing the total number of destination visits by all agents. However, the order of visits is not fixed, and each agent can choose its next destination from the two destinations it has not yet visited. At this point, the agent evaluates the value of its actions based on its own situation and the path information it can obtain from other agents and known wall information, and then decides on its actions. For path calculation, the A* algorithm[8] is used for both the agent's own path and the predicted paths of other agents. In this test, no wall information is exchanged between agents. Therefore, each agent predicts the actions of other agents based only on the wall information it knows. Wall information is updated as agents move, and the actions of other agents are predicted based on the updated wall information. The positions of other agents are distributed with probabilities of existence in the four squares to the front, back, left, and right based on their

exact positions and treated as likelihoods. In this test, the correct position is set at 90%, with 2.5% is allocated to each of the four surrounding squares. This is because in the real world, SLAM accuracy may decrease due to changes in the environment, and errors in self-position estimation may occur due to GPS errors. Furthermore, when the position of other agents cannot be obtained through communication, measurement errors may occur when sensing the positions of other agents. The direction of other agents can be obtained. Each agent shares accurate information about its current destination with all other agents, but the route to the destination is not shared. In addition, since each agent executes the algorithm asynchronously, the destination is updated as appropriate for each agent's execution unit. Agents have nine strategies. For destinations 1, 2, and 3, there are agents that use the shortest path algorithm to reach the destination, agents that use a different path, agents that wait for 2 seconds or 5 seconds, and agents that move randomly. The random movement strategy is executed when the only solution is for the agent to retreat to a location other than the goal. Fig. 11 shows the information integration screen displayed on the processing PC. The positions and directions of each agent are displayed as colored triangles, and destinations are displayed as colored circles. Sections where walls are unknown are displayed in gray, and sections where the presence or absence of walls has been determined are displayed in black where there are no walls and red where there are walls. However, as mentioned before, wall information is not shared between agents, so while the screen displays integrated information, each agent only recognizes the presence or absence of walls in the sections it has moved through. Fig. 12 shows the change in strategy. During algorithm execution, the agent evaluates nine strategies based on current information and executes the one with the highest evaluation value.

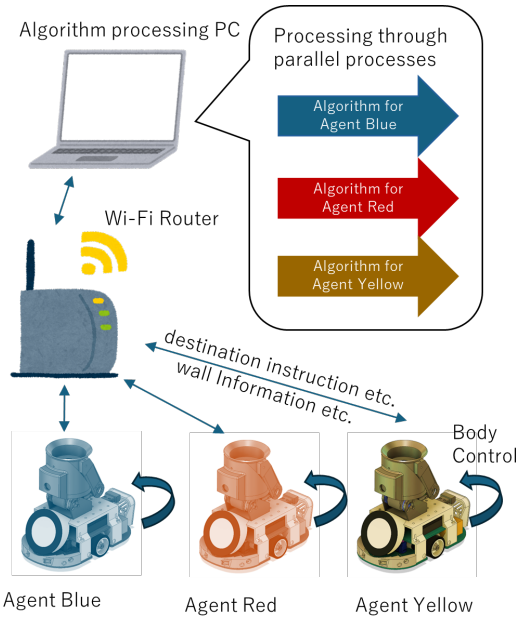

Fig. 9 Demonstration setup

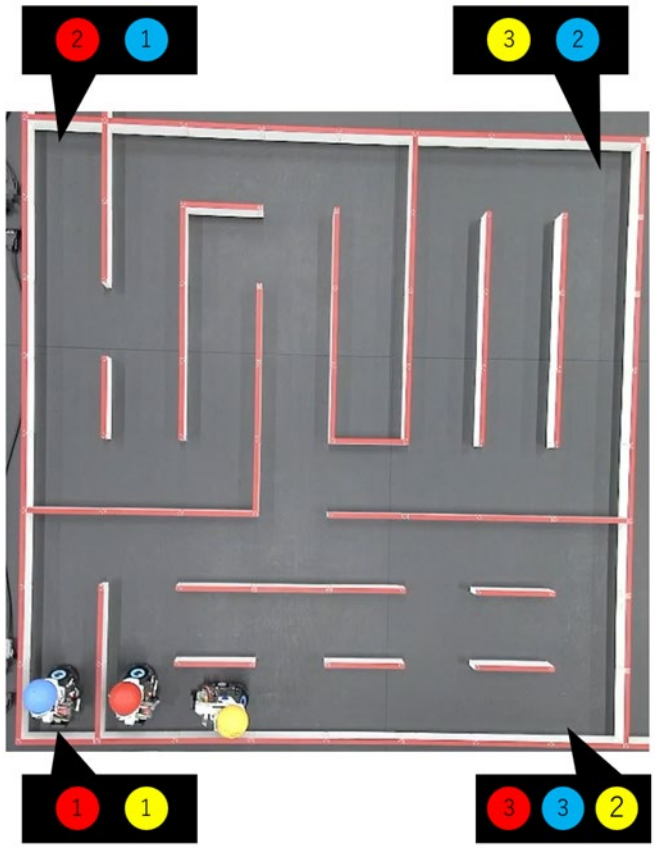

Fig. 10 Test Field and starting point for each agent

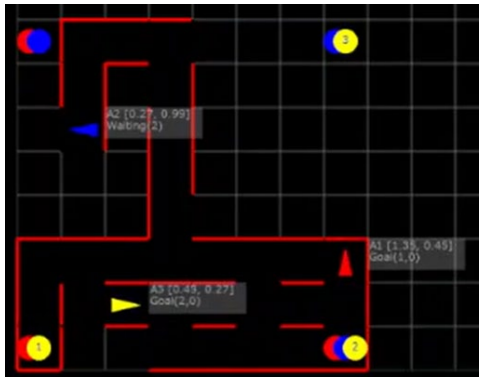

Fig. 11 Information integration screen

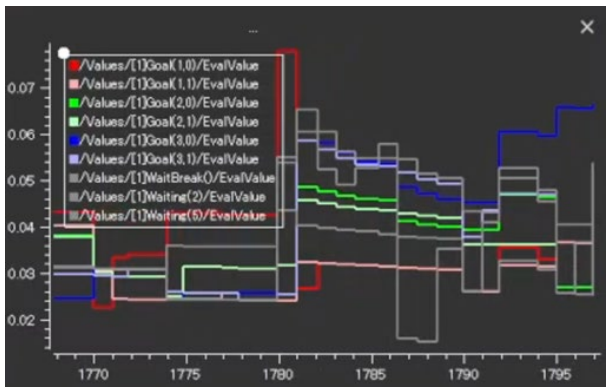

Fig. 12 Strategic change

V. RESULT

The test was conducted in the test field shown in Fig. 10. The test was conducted for 15 minutes at a time, and the value was assumed to be the maximum number of times the destination was reached. Two types of tests were conducted: one without a term to suppress the dispersion value between agents and one with such a term. In this test, the dispersion of the total number of times the destination was reached was suppressed. There are no terms to suppress the number of times each destination was reached. The difference in the total number of goals scored by the three agents after 15 minutes has become smaller.

The results of the test demonstrated that the agents were able to operate for 15 minutes without collision. Table 1 shows the test results when no term to suppress dispersion between agents was included. The table shows the number of times each of the three agents (Red, Blue, and Yellow) reached destinations 1, 2, and 3, with the total values shown in the SUM row. Additionally, the number of times each agent reached the destination at 5 minutes, 10 minutes, and 15 minutes after the start is shown. The number of times each destination was reached is the average of the results from six tests. In this test, it can be seen that the blue agent has a lower number of times reaching destinations. This is because, as shown in Fig. 10, Blue agent has the longest distances between destinations.

The results of adding a term to suppress the variance in the number of times each agent reaches its destination are shown in Table 2. The number of times Blue agent reaches its destination increases, and the variance between agents decreases significantly from 4.69 in Table 1 to 2.58 in Table 2. The total number of times all agents reach their destinations increases slightly from 58.33 to 59.83, indicating that minimizing variance does not necessarily reduce overall efficiency.

Table 1 Test without dispersion term

Agent-Goal \ Time		5min	10min	15min
Red	1	3.67	7.17	11.00
	2	0.67	1.17	1.67
	3	3.67	7.00	10.50
	SUM	**8.00**	**15.33**	**23.17**
Blue	1	1.33	2.67	4.67
	2	1.33	3.50	5.67
	3	1.50	2.17	2.50
	SUM	**4.17**	**8.33**	**12.83**
Yellow	1	2.83	5.83	9.33
	2	3.17	5.83	9.83
	3	0.67	2.17	3.17
	SUM	**6.67**	**13.83**	**22.33**

Table 2 Test with dispersion term

Agent-Goal \ Time		5min	10min	15min
Red	1	3.83	7.33	10.67
	2	0.67	1.17	1.50
	3	2.83	6.50	10.00
	SUM	**7.33**	**15.00**	**22.17**
Blue	1	1.33	3.50	5.67
	2	1.67	4.17	7.00
	3	1.83	2.83	3.67
	SUM	**4.83**	**10.50**	**16.33**
Yellow	1	2.50	5.50	8.67
	2	2.83	5.67	8.50
	3	1.17	2.67	4.17
	SUM	**6.50**	**13.83**	**21.33**

VI. CONCLUSION

We proposed a distributed coordination algorithm for multi-agent systems with multiple objectives, which performs distributed coordination while estimating the value generated by the entire system. We also verified the effectiveness of the algorithm by conducting operational verification using three agents and a grid-shaped field. In the future, we will conduct operational verification with an increased number of agents and agents with different characteristics and proceed with discussions toward practical application.

REFERENCES

[1] G. D. M. Serugendo, M. P. Gleizes, et al. : "Self Organization in Multi-Agent Systems", The Knowledge Engineering Review, Vol. 20, No. 2, pp. 165–189, 2005.

[2] T. Halsted and M. Schwager, "Distributed multi-robot localization from acoustic pulses using euclidean distance geometry," in 2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), pp. 104–111, IEEE, 2017.

[3] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed algorithms for environment partitioning in mobile robotic networks," IEEE Trans-actions on Automatic Control, vol. 56, no. 8, pp. 1834–1848, 2011.

[4] P. MAZDIN and B. RINNER, "Distributed and Communication-Aware Coalition Formation and Task Assignment in Multi-Robot Systems," IEEE Access (Volume: 9), pp. 35088 – 35100, 2021.

[5] D. F. Hougen, S. Benjaafar, J. C. Bonney, et al., "A miniature robotic system for reconnaissance and surveillance," in Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), vol. 1, pp. 501–507, IEEE, 2000.

[6] M. Simoncelli, G. Zunino, H. I. Christensen, and K. Lange, "Au-tonomous pool cleaning: Self localization and autonomous navigation for cleaning," Autonomous Robots, vol. 9, no. 3, pp. 261–270, 2000.

[7] V. Honkote1, D. Ghosh, K. Narayanan1, A. Gupta1 and A. Srinivasan, "Design and Integration of a Distributed, Autonomous and Collaborative Multi-Robot System for Exploration in Unknown Environments," IEEE/SICE International Symposium on System Integration, Jan 2020.

[8] P. E. Hart, N. J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Transactions on Systems Science and Cybernetics, Volume: 4, Issue: 2, July 1968.