[1, 2]Yong Xu
[1]Danfeng Li
[2]Malathy Batumalay
[2]Choon Kit Chan
[2]Long Jiang

Multi-robot Source Navigation Method Based on Coordination Graph Monte Carlo Tree Search

Journal of Electrical Systems

Abstract: - To address the dynamic cooperative multi-robot sequential decision problem, we propose a multi-robot source navigation method based on coordination graph Monte Carlo tree search. The Monte Carlo tree search online planning is designed with the algorithm system's structure in mind. We combine the upper confidence bound strategy with the coordination graph, which represents multi-robot cooperative communication, and solve the coordination graph using the max-sum algorithm to select joint actions. This paper proposes a time series prediction model for optimizing multi-robot movement trends using historical data, which is then used to solve the exploration and exploitation problem in Monte Carlo tree search. The experimental analysis and comparison show that our proposed method outperforms the comparison algorithms in terms of source search steps, task execution time, and source search success rate, as well as robustness and efficiency for multi-robot source search.

Keywords: Source navigation; Monte Carlo tree search; coordination graph; multi-robot; time series prediction.

I. INTRODUCTION

Multi-robot systems have advanced significantly in theory and practice as a result of modern industrial technology advancements, which encourage research into various types of robotic systems. For instance, Quan Yuan et al. proposed a Cooperative Game-based Motion Planning (CGP) method for multi-mobile robot path planning [1]. Belkacem Kada et al. proposed a continuous Distributed Consensus Control (DCC) scheme without control flutter [2], which solves the formation control problem of non-complete wheeled mobile robots by providing high tracking accuracy and smooth dynamic control inputs to agents. Zhang Jia et al. from Beijing University of Technology proposed an improved contract net auction algorithm [3] for designing and implementing a distributed multi-intelligent body detection task assignment and multi-robot system simulation platform. A multi-robot system is more than simply a collection of robots. In terms of overall system functionality, it is not as simple as scaling up individual robot functions. When compared to single robots, multi-robot systems are more functional, have greater redundancy, work more efficiently, and are more cost-effective [4-6].

With the rapid advancement of artificial intelligence, robots' intelligent characteristics are becoming more visible in the application of multi-robot fields. Robots have been used for source navigation in a variety of fields, including safety rescue, the petroleum industry, and underwater surveys [7, 8]. Source navigation refers to a group of searchers who follow a specific search plan to find and track down a target source of interest. Researchers are currently focusing more on multi-robot source navigation research, which has emerged as a dynamic research direction with promising applications in the field of robotics. However, in navigation operations, the movement of the target source and the search robot is often uncertain. Creating an efficient, stable, and low-cost multi-robot collaborative source search is critical, but it can be difficult.

The first and most important issue to address when transitioning from a single-robot to a multi-robot system is how to deal with the size of the joint action space. In a multi-robot system, each robot will have its own set of possible actions and observations, and the robots' relationships must be coordinated to achieve the common goal effectively. Most methods fail because the multi-robot problem has a large state and action space and requires a significant amount of training data to converge [9-14]. Traditional multi-robot path planning methods are both centralized and decentralized. Many recent studies [12, 13] have centered on decentralized policy learning, in which each robot develops its own strategy. Consider the possibility of multiple robots collaborating. Furthermore, there are methods for teaching robots to predict the behavior of other robots [9, 14], but this method is often difficult to scale as the team size increases.

[1]School of Artificial Intelligence and Smart Manufacturing, Hechi University, Hechi 546300, China
[2]INTI International University, Nilai 71800, Malaysia
*Corresponding Author: Choon Kit Chan.

The majority of cases involve some form of centralized learning, in which the sum of all robots' experiences is used for a common aspect of the training problem [15, 16], such as network output, value advantage calculation, and so on. Parameter sharing has been used to achieve faster and more stable training by sharing the weights of specific layers of a neural network when learning network outputs are centralised. For example, in the actor-critic approach [14], the network's critic output is typically trained centrally using parameter sharing because it applies to all robots in the system and has previously been used to train robot cooperation. When dealing with partially observable environments, centralized learning can be useful because it combines all robot observations into a single learning process [17-19]. As can be seen, having a source navigation method to coordinate robot actions in order to maximize the joint utility of multi-robot systems will be extremely beneficial.

In this study, we develop a coordination graph Monte Carlo tree search (CG-MCTS) algorithm, a multi-robot source navigation approach for dynamic collaboration, to address the issues mentioned above. The components of a multi-robot source navigation system combined with Monte Carlo tree search (MCTS) are examined first. Following that, an online MCTS planning approach using a coordination graph is proposed. Furthermore, the joint action selection is handled by a max-sum algorithm that combines an upper confidence bound strategy and a coordination network. Finally, a time-series prediction model that optimizes several robots' motion trends is developed to promote collaborative sourcing across multiple robots by incorporating historical data.

II. System Overview

A. Monte Carlo Tree Search

The MCTS algorithm is a decision-making method that requires understanding the problem and learning how to solve it. It will predict the best possible outcome and provide a solution. To determine the computer's strategy, hundreds of thousands of neurons are linked in layers. To find near-optimal solutions, MCTS does not require the use of domain-specific knowledge-derived heuristics and is very effective in searching large search spaces [20], making it ideal for multi-robot systems.

To determine the best move from a set of moves, MCTS selects, expands, simulates, and updates nodes in the tree to arrive at the final solution. The MCTS starts with the root node and includes the following information: {I(n), s(n), a(n), p(n), R(n), N(n)}. Among them, I(n) represents the signal value of node n's position s(n), a(n) represents the action direction from node n's parent node to node n, p(n) represents node n's prior action selection probability, R(n) represents node n's cumulative reward, and N(n) represents node n's visit count.

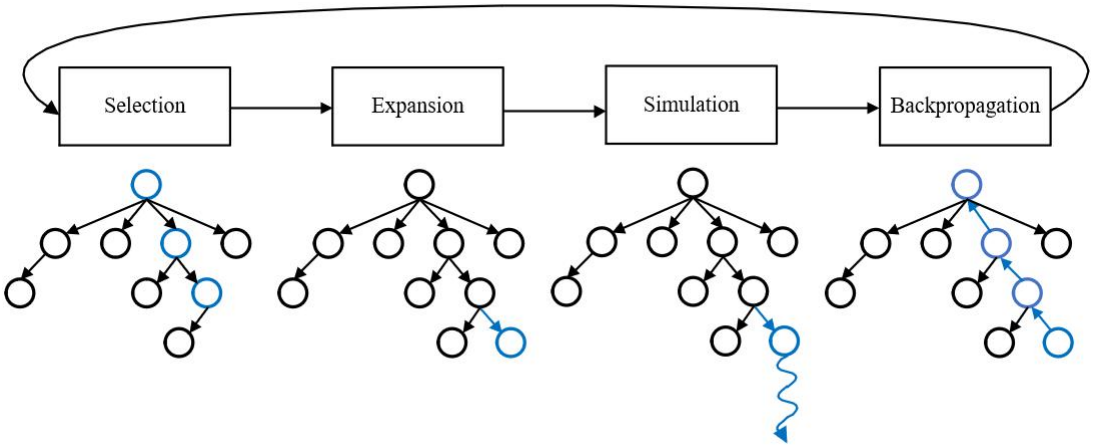


Figure 1: Block diagram of MCTS

Figure 1 shows a schematic diagram of MCTS, with the steps performed in each iteration being selection, expansion, simulation, and backpropagation. Repeat the preceding four steps until the maximum number of iterations is reached, then select the most confident node under the root node using the Upper Confidence Bound Apply to Trees (UCT) formula [21, 22], as shown in (1). This time step's output strategy is the good child node and its corresponding action a.

$$\boldsymbol{U} = \frac{\boldsymbol{R(n)}}{\boldsymbol{N(n)}} + \boldsymbol{c}\sqrt{\frac{\boldsymbol{\ln N(p)}}{\boldsymbol{N(n)}}} \tag{1}$$

N(p) represents the number of times the parent node of node n has been traversed.

B. System Framework

It remains difficult to develop a solution for multi-robot source navigation that strikes a balance between optimality and practical efficiency. The multi-robot path planning problem can be thought of as a sequential decision problem that each robot must solve at time t in order to reach its destination [23]. Figure 2 depicts the source navigation algorithm based on CG-MCTS. The algorithm is broken down into three sections: online planning with MCTS, joint action selection with an iterative maximum sum algorithm, and a temporal prediction model for optimizing multi-robot motion trends.

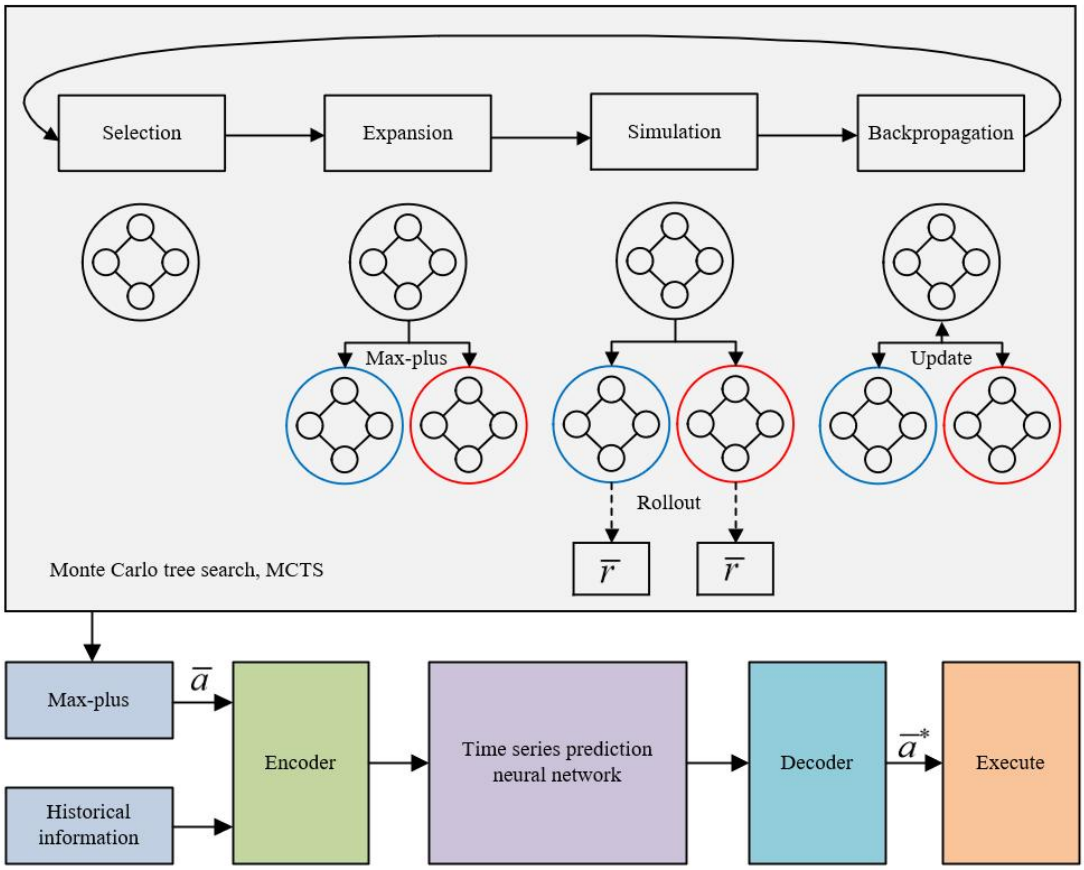


Figure 2. Structure of the source navigation algorithm

The system structure specifies what information must be shared between robots so that each robot can make decisions based on it. Furthermore, the robot only needs locally relevant information about its location and goals; it does not require global reference information. Multi-robot joint action selection can be optimized by iteratively sampling the action space at random, constructing a search tree based on the sampling results, solving the coordination graph with the maximum sum algorithm, and then learning the motion trends of multi-robots at historical moments.

In contrast to a single-robot system, a multi-robot system makes decisions for more than one robot. Although each robot will have its own set of possible actions and observations, this does not make full use of the valuable information obtained from other robots. As a result, in order to successfully complete the information source navigation goal, the robots' relationships must be coordinated. The system can determine which information is relevant to the entire multi-robot system, share that information, and finally facilitate efficient path planning for robots via joint training of the algorithm's three components.

III. COORDINATION GRAPH MONTE CARLO TREE SEARCH METHOD

A. Coordination Graph online planning

The coordination graph algorithm has made significant research advances in the fields of constraint satisfaction, probabilistic reasoning, and relational databases, including connection queries applied to computational relational data centers [24] and dynamic programming in robotics [25, 26]. The first consideration in multi-robot systems is to choose globally optimal joint actions that maximize the group's immediate value.

Assume a group of robots. Each robot j selects an action A_j, denoted by U as $\{A_j, ..., A_g\}$. Each robot j has a local function, Q_j, which represents its contribution to the overall benefit function. Robots collaborate to maximize:

$$Q = \sum_j Q_j \tag{2}$$

and each robot's Q_j can be influenced by its own and other robots' behavior. Define U Scope $[Q_j] \epsilon U$ as the multi-robot collection whose behavior affects Q_j. Each Q_j can be decomposed into a linear combination of functions that require fewer robots, further reducing the algorithm's complexity.

The system aims to choose a joint action that maximizes $\sum_j Q_j(a)$. Indeed, because Q_j relies on the actions of multiple robots, the robots must coordinate their actions. A coordination graph can be used to represent the system's coordination requirements, with each node representing a robot and an edge connecting two robots if they must coordinate their actions to optimize Q_j. Figure 3A depicts a coordination graph, as:

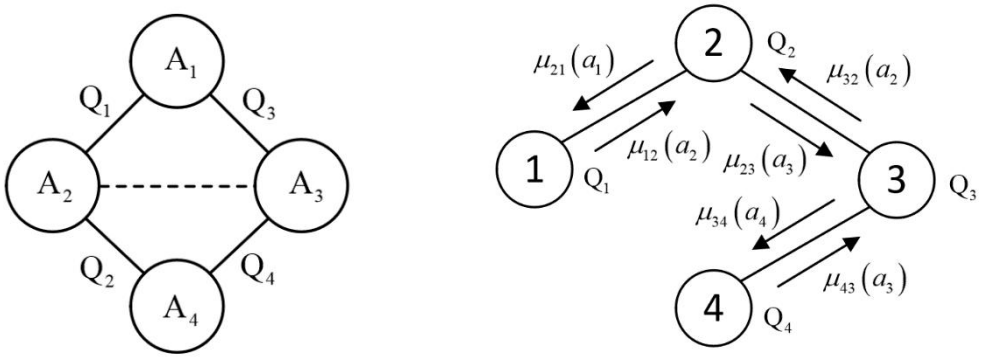
$$\mathrm{Q} = Q_1(a_1, a_2) + Q_2(a_2, a_4) + Q_3(a_1, a_3) + Q_4(a_3, a_4) \tag{3}$$

The key idea is to maximize one variable at a time rather than combining all functions and then maximizing.

When maximizing a_x, only those involved do so. For example, to optimize A_4, first optimize robot 4. The functions Q_1 and Q_3 are irrelevant, so we get:

$$Q = \max_{a_1, a_2, a_3} Q_1(a_1, a_2) + Q_3(a_1, a_3) + \max_{a_4} [Q_2(a_2, a_4) + Q_4(a_3, a_4)] \tag{4}$$

It is clear that in order to select the best A_4, the robot must understand the values of A_2 and A_3. In fact, it is calculating the conditional policy of robots 2 and 3's possible actions. Robot 4 can use a new function $e_4(a_2, a_3)$ to generalize the value it provides to the system in various situations. e_4 represents the addition of a new communication dependency between A_2 and A_3, denoted by the dotted line in Figure 3A.



(A) (B)

Figure 3. Diagram of coordination graph: (A) Schematic diagram of coordination graph; (B) Schematic diagram of coordination graph messaging.

When the problem is reduced to calculating:

$$Q = \max_{a_1, a_2, a_3} Q_1(a_1, a_2) + Q_3(a_1, a_3) + e_4(a_2, a_3) \tag{5}$$

it is clear that robot 4 has been eliminated.

The following step is to select robot 3, there is:

$$Q = \max_{a_1, a_2} Q_1(a_1, a_2) + e_3(a_1, a_2) \tag{6}$$

$$\text{where } e_3(a_1, a_2) = \max_{a_3} [Q_3(a_1, a_3) + e_4(a_2, a_3)].$$

Next, choose robot 2, have $e_2(a_1) = \max_{a_2} [Q_1(a_1, a_1) + e_3(a_1, a_2)]$. Finally, robot 1 can choose a_1 as the action that maximizes $e_1 = \max_{a_1} e_2(a_1)$. As a result, reversing the process yields the optimal action set. Increasing selection e_1 selects action $a_1{}^*$ for robot 1. To satisfy robot 1's selection, robot 2 must choose action $a_2{}^*$ to maximize $e_2(a_1{}^*)$, forcing robots 3 and 4 to choose their own actions.

The coordination graph online planning algorithm will track a set of functions F, starting with $\{Q_1, ..., Q_g\}$. The algorithm then goes through the following steps: (1) Select a non-eliminated robot A_i. (2) Take all ranges of F, including $e_1, ..., e_i$ of A_i. (3) Define a new function $e = \max_{a_i} \sum_j e_j$ and incorporate it into F. Finally, by sending messages backwards, the optimal action selection is achieved.

B. Max-plus Algorithm

Following multi-robot online planning with the coordination graph, an MCTS was carried out using the Max-plus algorithm. In graphical models, the Max-plus algorithm is equivalent to belief propagation, and its time complexity varies linearly with the size of the coordination graph. It is better suited to real-time systems than variable elimination and can deal with multi-robot systems more effectively.

To begin, each node is defined to keep decomposition statistics. Given a potential state-related undirected coordination graph $= \langle v, \varepsilon \rangle$, the current global benefit of the coordination graph is decomposed as follows:

$$Q(\bar{a}) = \sum_{i \in v} Q_i(a_i) + \sum_{(i,j) \in \varepsilon} Q_{ij}(a_i, a_j) \tag{7}$$

Q_{ij} is the local profit function of robots i and j linked by edge (i, j), while Q_i is the individual profit function of robot i.

In probabilistic graphical models, a Max-plus algorithm for computing joint actions via message passing exploits the duality of computing maximal, posterior, and optimal joint actions in coordination graphs [27]. Each robot node in the coordination graph sends messages to its neighbor $j \in \Gamma(i)$. Messages from robot i are scalar-valued functions that accept robot j's action space, for example:

$$\mu_{ij}(a_j) = \max_{a_i} \left\{ Q_i(a_i) + Q_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus \{j\}} \mu_{ki}(a_i) \right\} \tag{8}$$

where$\Gamma(i)$ denotes the set of robot i's neighbors, and the robots exchange information until convergence or the maximum number of rounds is reached. Finally, each robot calculates its own best move, which is:

$$a_i^* = \underset{a_i}{argmax} \left\{ Q_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i) \right\} \tag{9}$$

C. Upper Confidence Bound Strategy Exploration

The Upper Confidence Bound (UCB) algorithm overcomes all of the limitations of exploration and exploitation-based strategies, including the requirement to understand breadth and suboptimality gaps [28]. This algorithm can take a variety of forms depending on the distributional assumptions made about the noise. The UCB algorithm operates on the principle of selective action in the face of uncertainty. Two things happen when you act optimistically. Regardless, optimism is justified, and the robot's behavior is optimal. Either the optimism is misplaced, and the bots take an action that they believe will result in a large payoff but does not. If this occurs frequently enough, the bot will determine the true payoff of the action and avoid it in the future.

The UCB method will be used to deduce and calculate all actions after a set number of cycles. However, with continuous iterations, the likelihood of being selected again in subsequent iterations decreases for actions with low estimated values and action nodes that have been chosen multiple times. The main issue in extending MCTS to coordination graphs is upper confidence bound strategy exploration. It consists of two distinct computation phases: message passing for each edge (8) and action selection for each node (9).

Figure 3B depicts a schematic diagram of message passing in the coordination graph, with edge and node exploration. When investigating similar income statistics Q_{ij}, record the corresponding paired action statistics N_{a_i,a_j}. The exploration strategy is to add rewards to (8), as shown below:

$$\mu_{ij}(a_j) = \max_{a_i} \left\{ Q_i(a_i) + Q_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus \{j\}} \mu_{ki}(a_i) + \sqrt[c]{\frac{log\,(N+1)}{N_{a_i,a_j}}} \right\} \tag{10}$$

To add node exploration rewards during action selection, keep individual action frequencies N_{a_i} constant and modify (9) as follows:

$$a_i^* = \underset{a_i}{argmax} \left\{ Q_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i) + \sqrt[c]{\frac{log\,(N+1)}{N_{a_i}}} \right\} \tag{11}$$

In formula (7), the joint action payoff $Q(\bar{a})$ can be factored on coordination graph nodes and edges, but there is no reward $\sqrt[c]{\frac{\log(N+1)}{N_{a_i,a_j}}}$. As a result, the node and edge exploration strategy described here is a heuristic alternative to the previous component-based exploration using variable elimination, which only considers nodes and edges in the coordination graph.

D. Time Series Prediction Model

Finding path planning and coordination solutions during the process of locating signal sources is critical to improving system performance in a multi-robot system. Current methods can be classified as either centralized or decentralized. The centralized approach requires the use of a central unit to collect information from all robots and organize the best path for each robot, which takes up a significant amount of computing resources.

Decentralized approaches, in which each robot estimates or communicates the future trajectories of other robots via broadcast or distance-based communication, are gaining popularity as systems scale [13]. When many adjacent robots communicate simultaneously and equally, it can lead to redundant communication, which consumes computing power and has a negative impact on the team's overall performance. Furthermore, due to limited bandwidth, large data volumes, and environmental interference, reliable and continuous communication is not guaranteed. It is difficult to ensure training process convergence in a fully decentralized framework without a priority plan, so a temporal prediction model is needed to actively measure the motion trends of multiple robots during the sourcing process.

Time series data are used in a variety of research areas. There are numerous techniques for developing predictive models, which can be broadly divided into two categories: traditional statistical techniques and machine learning techniques [29-31]. Using appropriate techniques and time series characteristics, an accurate predictive model will be obtained. Time series forecasting accuracy has recently been improved using deep neural networks, such as Long Short-Term Memory neural networks (LSTM) [32, 33].. Because the dynamic behavior of various complex systems can be described by temporal patterns of interactions between system components, the problem of predicting the movement trend of multi-robots must be addressed. This paper investigates both the robot's external factors and its own state changes, and then incorporates this historical data into the time series prediction model to obtain the best action decision.

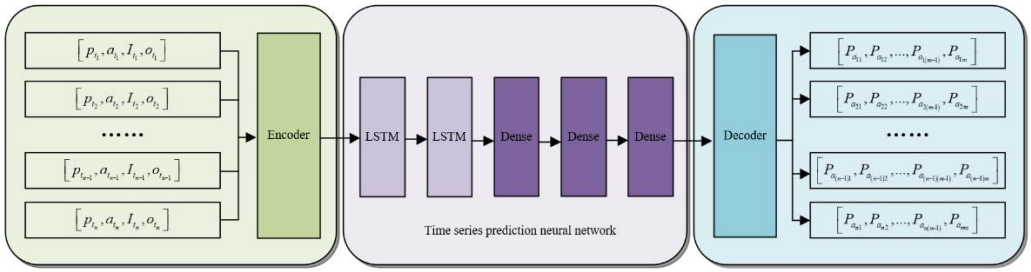


Figure 4. Structural diagram of time series prediction model

Figure 4 depicts the time series prediction model structure diagram, which consists of an encoder, a time series prediction neural network, and a decoder. The encoder's primary function is to encode each robot's state information over time. The state information includes the robot's position p_t at the time, the action selection a_t, the signal strength I_t of the current position, the angle information o_t from the target position, and the state at time t. The symbols for information are $\{p_t, a_t, I_t, o_t\}$. The sequential neural network consists of two layers of LSTMs and three layers of fully connected networks. The decoder essentially solves the sequential neural network output to determine the multi-robot system's optimal joint action selection at the current time. The time series prediction model's optimization goal is to identify a set of optimal solutions that allow multi-robot actions to move as close to the target direction as possible. The loss function used during training is as follows:

$$Loss = \sum_{i=1}^{n} \sum_{j=1}^{m} P_{a_{ij}} \times D_{a_{ij}} \tag{12}$$

$$D_{a_{ij}} = \begin{cases} 10 & d(s, a_{ij}) > d(s) \\ -5 & d(s, a_{ij}) = d(s) \\ -20 & d(s, a_{ij}) < d(s) \end{cases} \tag{13}$$

Where $P_{a_{ij}}$ is the probability that robot i will choose action j, d(s) is the distance from the current state to the target, and $d(s, a_{ij})$ is the distance after taking action a_{ij} in the current state.

IV. Experiment and Analysis

A. Experiment Settings

In the experimental scenario presented in this paper, multiple robots work together to locate the target signal source. The signal field is contained within a discrete grid composed of multiple robots and a signal source[34]. Robots perform actions in the environment while also collecting data on its state. During the source-finding process, robots communicate with one another in order to exchange useful information about the location of the target signal source. During the source-seeking task, rewards will be given based on the robot's proximity to the signal source, while penalties will be applied for collisions.

Each robot starts in a randomly selected unique cell in the grid world and performs discrete actions like moving a cell or remaining stationary. Certain actions, such as moving toward a wall or another robot, may be invalid at each time step. Experiments show that this approach produces more stable training than providing negative rewards to a robot that chooses invalid actions. Furthermore, the reward function follows the reward rules established in the majority of such experiments, in which the robot is penalized or rewarded at each time step, resulting in a policy of completing the task as quickly as possible. The reward for the robot is slightly higher than the punishment, and the corresponding reward is obtained when the robot is closer to the target position (experimental setting 1000), whereas the penalty is given when the robot is further away from the target position or the two robots are too close (experimental setting 100), which is required to encourage exploration.

Table 1. Network structure of the time series prediction model

Number of Network Layers	Number of Neurons	Activation Function
LSTM	512	-
LSTM	1024	-
Full connected	512	ReLU
Full connected	128	ReLU
Full connected	72	SoftMax

Table 1 shows the network structure of the time series prediction model. Following the superimposition of two layers of LSTM, a three-layer fully connected network is used to predict joint action probability. The repetition rate of information data is excessive due to the simultaneous sampling of adjacent time steps. To avoid this, the sampling interval is configured to sample historical data every six time steps. The Adam optimizer trains in 500 epochs with a batch size of 128. The encoder encodes each robot's position, action, signal strength, and angle information in the first 16 time steps at the current moment, which is then fed into the model; the decoder outputs the probability of each robot in each action direction. The robot is currently selecting a joint action.

B. Result Analysis

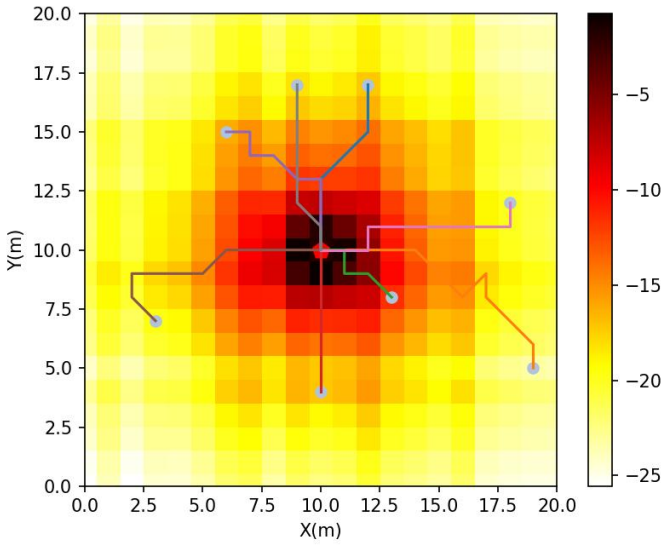


Figure 5. Diagram of multi-robot source navigation scenario

Figure 5 depicts a navigation path diagram from a multi-robot signal source navigation experiment. The eight robots' initial positions are randomly distributed around the signal source. The figure depicts how, by communicating and cooperating, each robot can move in the direction of the signal source and reach the desired signal source. It demonstrates that the multi-robot system can accurately exchange information obtained during the sourcing process via communication, as well as that the time series prediction model can capture historical data to improve the accuracy of end-to-end joint action prediction. As a result, the CG-MCTS source navigation method has increased stability and action selection precision.

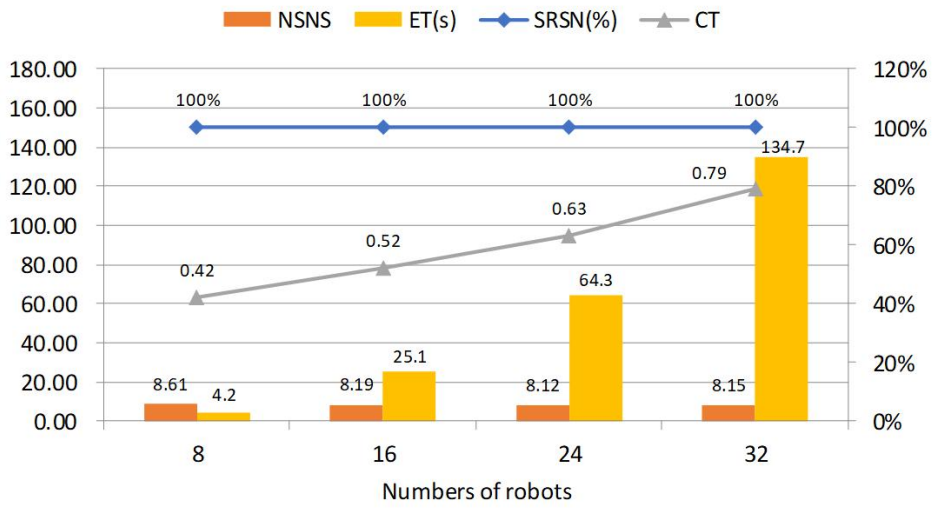


Figure 6. Test data graph of different numbers of robots

Figure 6 depicts a test data diagram of the number of source navigation steps (NSNS), execution time (ET), success rate of source navigation (SRSN), and collision times (CT) for various numbers of robots in a 20*20 environment. The graph shows that as the number of robots increases, the system's execution time grows exponentially, the number of collisions grows slowly, the number of source navigation steps decreases slowly, and the success rate of source navigation is 100%. The execution time increases exponentially due to the fixed size of the environment, the increased number of robots, and the time required for robot communication and actions. The number of collisions gradually increases, indicating that the number of robots has little effect on collisions and that the path planning algorithm proposed in this paper is effective. When faced with dynamic obstacles, the robots cooperate and have some obstacle-avoidance capabilities. The number of source navigation steps gradually decreases, indicating that as the number of robots increases, so does the amount of environmental information shared between robots, encouraging the robot to complete the source-finding task over a shorter distance to some extent. The success rate of source navigation is 100%, indicating that the algorithm is effective in multi-robot collaborative sourcing.

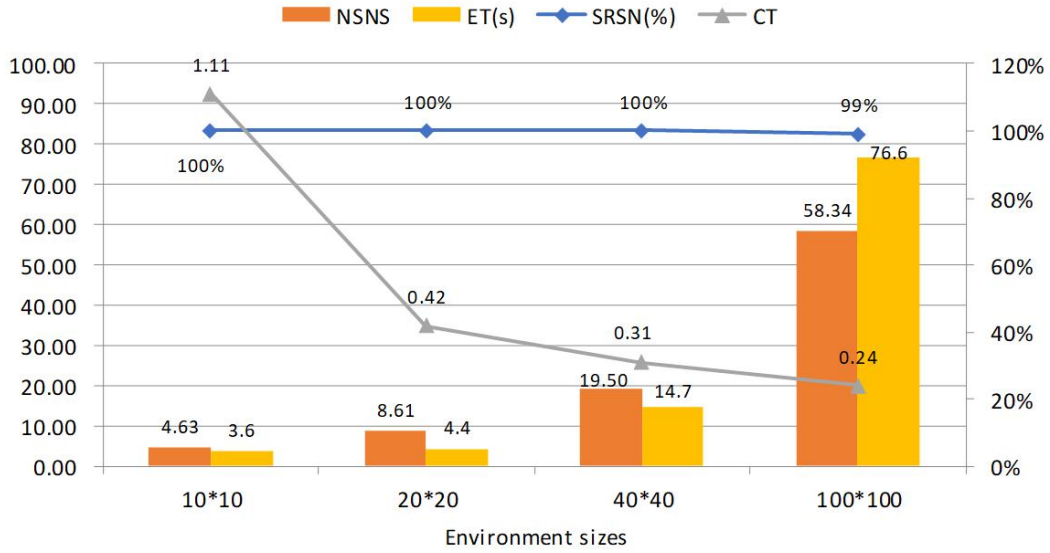


Figure 7. Test data graph of different environment sizes

Figure 7 depicts test data graphs of NSNS, ET, SRSN, and CT for various environment sizes when there are eight robots. The figure depicts how, as the environment grows larger, the distance for the robot to complete the source-finding task naturally increases, as does the number of steps and execution time of the source navigation, but the CT decreases and the SRSN increases. The CG-MCTS signal can be observed. The source navigation method, which is robust and reliable, can fully utilize the robot's historical motion information to promote multi-robot collaborative navigation and efficient decision-making. Furthermore, the figure demonstrates that the

SRSN is only 99% in a 100*100 grid environment. Because of the large size of the signal field and the small change in signal strength far from the signal source, the robot source search is required. It is simple to fall into a local optimal cycle, resulting in source seeking failure.

We compare the proposed CG-MCTS method with the Multi-Agent Monte Carlo Tree Search (MA-MCTS) method [35], the Safe Interval Path Planning (SIPP) method [36], and the Conflict-Based Search (CBS) method [37]. The number of robots in the same source environment is limited to eight, and 100 multi-robot source navigation simulation experiments are carried out at random using four methods. In each experiment, the robot's initial position is chosen at random, and the maximum number of iteration steps in a single experiment is 200. Each method's hyperparameters are the best values discovered through repeated experiments. Table 2 shows the results of the source-finding test. All four methods have source-finding success rates of more than 90%. In the experiment, the CG-MCTS proposed in this paper was able to locate the target signal source 100% of the time. It is more effective than the other three strategies.

Table 2. Table of sourcing tests for different algorithms

Algorithms	SRSN	Average NSNS	ET
CG-MCTS	100%	8.5	4.3
MA-MCTS	99%	13.0	9.5
SIPP	95%	24.0	5.0
CBS	90%	19.0	6.5

The average number of sourcing steps is calculated by taking the average NSNS in a given number of experiments. Table 2 shows that the average number of steps for the CG-MCTS algorithm is 8.5 steps, and the time spent finding the source is 4.3 seconds, the shortest of the four algorithms, indicating that the algorithm can efficiently complete the multi-robot search task. The MA-MCTS method allows the robot to efficiently search the available strategy space while focusing on strategies that benefit the entire system. The experimental result shows a relatively low number of source-seeking steps, with 13 steps. However, the MA-MCTS method is time-consuming because it executes the operations stored in the strategy, affecting system execution time. The SIPP and CBS methods have fairly stringent collision conflict requirements. When two or more robots attempt to enter the same grid at the same time, they will pause and wait for re-planning at the next opportunity, resulting in a relatively large number of steps. However, due to the low cost of node expansion, the two algorithms will have a short running time. In conclusion, the CG-MCTS method proposed in this paper outperforms other comparable algorithms in terms of sourcing success rate, sourcing step count, and task execution time. As a result, CG-MCTS is effective in multi-robot sourcing in a point source environment. Robustness and high efficiency are important characteristics.

V. CONCLUSIONS

To effectively complete the common goal of source navigation, multi-robot systems must coordinate each robot's relationship and solve the sequential decision-making problem of dynamic collaboration. As a result, we propose using the coordination graph to represent the system's coordination requirements, solving the coordination graph with the maximum sum algorithm to determine the robot's joint action selection, and developing a source navigation method based on the CG-MCTS. The algorithm is broken down into three parts: online planning with MCTS, an iterative maximum-sum algorithm for selecting joint actions, and a temporal prediction model for optimizing multi-robot motion trends. Finally, the experimental results show that, when compared to other multi-robot collaborative algorithms, the MCTS method based on coordination graphs can fully utilize both the source-seeking process's historical motion information and the signal strength information in the signal field, significantly improving the multi-robot coordination algorithm. It can improve the success rate of collaborative sourcing, reduce sourcing costs, and be used in multi-robot navigation.

Our findings suggest the possibility of multi-robot dynamic collaboration in source navigation decision-making. The primary research direction in the future will be how to improve positioning accuracy while maintaining source search efficiency, as well as how to conduct hardware experimental tests.

ACKNOWLEDGEMENT

This research was financially supported by First-class Discipline Construction Project of Hechi University, Guangxi Colleges and Universities Key Laboratory of AI and Information Processing (Hechi University), Education Department of Guangxi Zhuang Autonomous Region.

Funding Statement: This research was funded by the Natural Science Foundation of Guangxi Province under Grant No: 2023GXNSFAA026025, the Project of Ministry of Education supply and demand docking employment education under Grant No: 20230106490, the Research Project of Hechi University under Grant No: 2021XJD003, Special Research Project of Hechi University under Grant No: 2022YLXK003.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

REFERENCES

[1] Yuan, Q., Li, L., Wang, C., et al. (2020). Cooperative Game-based Approach for Local Path Planning of Multiple Mobile Robots in Opposite-direction Scenarios. 2020 Chinese Automation Congress (CAC): IEEE, 5464-5469.

[2] Kada, B., Balamesh, A.S., Juhany, K.A., et al. (2020). Distributed cooperative control for nonholonomic wheeled mobile robot systems. International Journal of Systems Science, 51(9), 1528-1541.

[3] Zhang, J., Ren, Q. (2021). Design and implementation of task allocation experiment in multi agent system. Research and Exploration in Laboratory, 40(07), 121-125.

[4] Xiao, L., Gong, J., Chen, J. (2020). Industrial robot control systems: a review. Proceedings of the 11th International Conference on Modelling, Identification and Control (ICMIC2019): Springer, 1069-1082.

[5] Ju, C., Kim, J., Seol, J., et al. (2022). A review on multirobot systems in agriculture. Computers and Electronics in Agriculture, 202, 107336.

[6] Farivarnejad, H., Berman, S. (2022). Multirobot control strategies for collective transport. Annual Review of Control, Robotics, and Autonomous Systems, 5, 205-219.

[7] Liu, B., Zhao, G., Liu, S., et al. (2020). Data source selection for UAVs' networked navigation system based on chaos particle swarm optimization. Electronics Optics & Control, 27(08), 64-68+74.

[8] Zhou, Y., Lai, J., Guo, X., et al. (2015). A research on all source navigation and positioning and its critical technology. China Satellite Navigation Conference (CSNC) 2015 Proceedings: Volume III: Springer, 801-808.

[9] Bae, H., Kim, G., Kim, J., et al. (2019). Multi-robot path planning method using reinforcement learning. Applied sciences, 9(15), 3057.

[10] Yang, Y., Juntao, L., Lingling, P. (2020). Multi-robot path planning based on a deep reinforcement learning DQN algorithm. CAAI Transactions on Intelligence Technology, 5(3), 177-183.

[11] Long, P., Fan, T., Liao, X., et al. (2018). Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. 2018 IEEE international conference on robotics and automation (ICRA): IEEE, 6252-6259.

[12] Foerster, J., Assael, I.A., De Freitas, N., et al. (2016). Learning to communicate with deep multi-agent reinforcement learning. Advances in neural information processing systems, 29.

[13] Khan, L.U., Saad, W., Han, Z., et al. (2021). Dispersed federated learning: Vision, taxonomy, and future directions. IEEE Wireless Communications, 28(5), 192-198.

[14] Lowe, R., Wu, Y.I., Tamar, A., et al. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems, 30.

[15] Devin, C., Gupta, A., Darrell, T., et al. (2017). Learning modular neural network policies for multi-task and multi-robot transfer. 2017 IEEE international conference on robotics and automation (ICRA): IEEE, 2169-2176.

[16] Foerster, J., Farquhar, G., Afouras, T., et al. (2018). Counterfactual multi-agent policy gradients. Proceedings of the AAAI conference on artificial intelligence.

[17] Zhu, P., Dai, W., Yao, W., et al. (2020). Multi-robot flocking control based on deep reinforcement learning. IEEE Access, 8, 150397-150406.

[18] Gupta, J.K., Egorov, M., Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16: Springer, 66-83.

[19] Wang, D., Deng, H. (2021). Multirobot coordination with deep reinforcement learning in complex environments. Expert Systems with Applications, 180, 115128.

[20] Browne, C.B., Powley, E., Whitehouse, D., et al. (2012). A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games, 4(1), 1-43.

[21] Liu, Y.-C., Tsuruoka, Y. (2016). Modification of improved upper confidence bounds for regulating exploration in monte-carlo tree search. Theoretical Computer Science, 644, 92-105.

[22] Duan, S., He, H., Xu, C., et al. (2022). A deep sequential monte-carlo tree search method for source navigation in unknown environments. Acta Electronica Sinica, 50(07), 1744-1752.

[23] Wang, D., Qi, H., Lian, B., et al. (2023). Resilient Decentralized Cooperative Localization for Multi-Source Multi-Robot System. IEEE Transactions on Instrumentation and Measurement.

[24] Girault, J.-T., Gayah, V.V., Guler, I., et al. (2016). Exploratory analysis of signal coordination impacts on macroscopic fundamental diagram. Transportation Research Record, 2560(1), 36-46.
[25] Siméon, T., Leroy, S., Lauumond, J.-P. (2002). Path coordination for multiple mobile robots: A resolution-complete algorithm. IEEE transactions on robotics and automation, 18(1), 42-49.
[26] Spensieri, D., Åblad, E., Bohlin, R., et al. (2021). Modeling and optimization of implementation aspects in industrial robot coordination. Robotics and Computer-Integrated Manufacturing, 69, 102097.
[27] Vlassis, N., Elhorst, R., Kok, J.R. (2004). Anytime algorithms for multiagent decision making using coordination graphs. 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat No 04CH37583): IEEE, 953-957.
[28] Zhu, J., Liu, J. (2023). Distributed multi-armed bandits. IEEE Transactions on Automatic Control.
[29] Karevan, Z., Suykens, J.A. (2020). Transductive LSTM for time-series prediction: An application to weather forecasting. Neural Networks, 125, 1-9.
[30] Zhang, Y., Wang, X., Tang, H. (2019). An improved Elman neural network with piecewise weighted gradient for time series prediction. Neurocomputing, 359, 199-208.
[31] James, S.C., Zhang, Y., O'Donncha, F. (2018). A machine learning framework to forecast wave conditions. Coastal Engineering, 137, 1-10.
[32] Li, Y., Zhu, Z., Kong, D., et al. (2019). EA-LSTM: Evolutionary attention-based LSTM for time series prediction. Knowledge-Based Systems, 181, 104785.
[33] Hua, Y., Zhao, Z., Li, R., et al. (2019). Deep learning with long short-term memory for time series prediction. IEEE Communications Magazine, 57(6), 114-119.
[34] Li, R.-G., Wu, H.-N. (2023). Multi-Robot Plume Source Localization by Distributed Quantum-Inspired Guidance With Formation Behavior. IEEE Transactions on Intelligent Transportation Systems.
[35] Zerbel, N., Yliniemi, L. (2019). Multiagent monte carlo tree search. Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems2309-2311.
[36] Yakovlev, K., Andreychuk, A., Stern, R. (2020). Revisiting bounded-suboptimal safe interval path planning. Proceedings of the International Conference on Automated Planning and Scheduling300-304.
[37] Lee, H., Motes, J., Morales, M., et al. (2021). Parallel hierarchical composition conflict-based search for optimal multi-agent pathfinding. IEEE Robotics and Automation Letters, 6(4), 7001-7008.