Hindawi
Complexity
Volume 2021, Article ID 6698421, 17 pages
https://doi.org/10.1155/2021/6698421

Research Article

Swarm Robot Exploration Strategy for Path Formation Tasks Inspired by *Physarum polycephalum*

Yandong Luo, Jianwen Guo, Zhenpeng Lao, Shaohui Zhang, and Xiaohui Yan

School of Mechanical Engineering, Dongguan University of Technology, No. 1 Daxue Road, Songshanhu Lake, Dongguan, Guangdong, China

Correspondence should be addressed to Jianwen Guo; guojw@dgut.edu.cn

Received 13 December 2020; Revised 17 February 2021; Accepted 4 May 2021; Published 19 May 2021

Academic Editor: Ning Cai

Physarum polycephalum, a unicellular and multiheaded slime mould, can form highly efficient networks connecting separated food sources during the process of foraging. These adaptive networks exhibit a unique characteristic in that they are optimized without the control of a central consciousness. Inspired by this phenomenon, we present an efficient exploration and navigation strategy for a swarm of robots, which exploits cooperation and self-organisation to overcome the limited abilities of the individual robots. The task faced by the robots consists in the exploration of an unknown environment in order to find a path between two distant target areas. For the proposed algorithm (EAIPP), we experimentally present robustness tests and obstacle tests conducted to analyse the performance of our algorithm and compare the proposed algorithm with other swarm robot foraging algorithms that also focus on the path formation task. This work has certain significance for the research of swarm robots and *Physarum polycephalum*. For the research of swarm robotics, our algorithm not only can lead multirobot as a whole to overcome the limitations of very simple individual agents but also can offer better performance in terms of search efficiency and success rate. For the research of *Physarum polycephalum*, this work is the first one combining swarm robots and *Physarum polycephalum*. It also reveals the potential of the *Physarum polycephalum* foraging principle in multirobot systems.

1. Introduction

In robotics, the process of searching for a specified target in an unknown environment by a group of robots is important for solving various problems in unknown environments, such as navigation [1–3], searching the whole area to solve the area coverage problem [4] and realism the whole area's the simultaneous localization and mapping (SLAM) [5]. In fact, exploration is the most important class of applications in the area of robotics because many real-life applications belong to this class, such as search [6] and rescue [7], area coverage [8], and military actions [9]. However, collective central place foraging by superorganismal social insect colonies elegantly and scalably solves the problem of resource collection in an uncertain environment [10, 11]. Accordingly, engineers have drawn inspiration from social insects to design swarm robotics systems that collectively solve foraging-like tasks in a parallel mode [12–14].

This paper focuses on a special case of exploration for swarm robotics: the path formation. The goal of this task is to find a collective path of robots between two locations. Most of the research studies on exploration address the process of exploration, which is easy to realize by some robots with complex functions. However, the exploration tasks addressed in swarm robot with limited cognitive abilities are not always easy to realize it. This kind of exploration task poses the following challenges: (1) The robots not only need to coordinate to find item sources in an unknown environment but also need to self-organize to form a path that connects item sources. (2) Each robot has limited cognitive abilities without any complicated centralized control, communication, and positioning. This means that the realization of the entire task can only rely on the perception and interaction between the robot and the environment. (3) Meanwhile, compared with the entire area, the communication range between robots is also very limited. When the

robot is far away from other robots, the robot will lose communication with the group, which will cause the swarm system search efficiency to decrease. Therefore, robots not only need to search efficiently but also need to continuously maintain communication with the group. The foraging behaviour of a slime mould from nature gives us a new way for solving these challenges.

The foraging behaviour of a particular primitive single-cell slime mould, called *Physarum polycephalum*, is to expand into its surroundings to form a staggered and efficient network connecting food sources. During the process of foraging, it shows excellent capabilities in terms of network construction and optimization without the central control [15, 16]. Many studies are based on observations of biological organisms, with the goal of simulating their intelligent behaviour from physical, chemical, biological, and other perspectives for application to various practical problems, such as wireless sensor networks (WSNs) [17] and the travelling salesman problem (TSP) [18]. Such a foraging behaviour without central control also meets the requirements of a multirobot system.

So, we draw inspiration from the foraging behaviour of *Physarum polycephalum* and present the first proposal that combines this behaviour with the requirements of path formation tasks for swarm robots. And then we designed four mechanisms to imitate the foraging principle of *Phloris polycephalum* and to design a self-organizing exploration algorithm inspired by *Physarum polycephalum*, called EAIPP. Our algorithm can lead multirobot to overcome the limitations of very simple individual agents to accomplish the task of exploring an unknown environment to find a specified target and connect them. In simulation experiments, we use the same environment that was used to test the *Physarum*-inspired multiagent system (P-MAS) [19] to verify that our algorithm is equally effective in solving maze problems and exhibits search characteristics similar to the foraging behaviour of *Physarum polycephalum*. In addition, the proposed algorithm is validated, and tests of related performance aspects (such as robustness testing and obstacle testing) are performed through simulations. Meanwhile, we also compare the performances with other existing multirobot path formation algorithms. These algorithms include PFIAS algorithm [20], our previously proposed TSASR algorithm [21], and a neural network evolution algorithm applied to multirobot path formation (NNEA) [22]. The test results show that the method proposed here achieves higher efficiency and a better success rate than the other algorithms.

Our results are of potential interest to both swarm engineers and behavioural ecologists, in that they demonstrate the sufficiency of very simple individual agents to generate sophisticated collective behaviour, as well as its scalability, and reveal the potential of the *Physarum polycephalum* foraging principle in multirobot systems. Although Jones et al. [23] have proposed a multiagent system to simulate the foraging behaviour of *Physarum polycephalum*, that work is realized by simulation of virtual multiagent particles in a virtual environment full of chemical nutrients. The virtual multiagent system continuously explores the area and releases agents' secrete trail (that is a chemical substance similar to the pheromone secreted by ants) to share the local environment. But for the multirobot system, there are more challenges: (1) The robot operating environment is not composed of chemical nutrients, and the robot cannot release chemical nutrients to induce other individuals to move. (2) In the swarm robot system, collisions will affect the overall operation effect, while the virtual multiagent system can overlap in the same position without the impact of collisions. This work is the first one combining multirobot systems and *Physarum polycephalum*.

2. Related Work

2.1. Physarum polycephalum. The slime mould *Physarum polycephalum* is a unicellular, multinucleate protist that relies on reactive navigation to explore its environment. All studied behaviours concerning adaptive network formation are exhibited in the active vegetative stage of its complex life cycle, called the "plasmodium" [24]. From a macroperspective, the feeding behaviour of *Physarum polycephalum* is to grow and expand its shape in accordance with its environment to form a network structure, as shown in Figure 1. This decentralized foraging behaviour of *Physarum polycephalum* is driven by the structural composition of the network. The network consists of a gel-like outer layer and an interior cytoplasmic fluid (also called the protoplasm). The outer layer houses an actin-myosin cytoskeleton that generates periodic contractions of the tube walls [25]. The contraction amplitude and frequency generally increase or decrease throughout the organism when it encounters an attractant (such as food) or a repellent (such as a dry area), respectively. The contractions from the gel-like outer layer drive the protoplasm flow, which traverses the entire individual to the extension zone and leads the organism to avoid areas that are not beneficial for growth or have already been explored [26]. The protoplasm promotes the expansion and exploration of *Physarum polycephalum* in the extension zone. This is a type of feedback behaviour that proceeds without central control.

Despite the lack of central control, *Physarum polycephalum* can still show complex and intelligent foraging behaviour through the above principles. For example, *Physarum polycephalum* can explore a maze environment and gradually form a shortest path connecting the exit and entrance of the maze, as previously reported in Nature [15]. Similarly, a study published in Science has shown that *Physarum polycephalum* can connect various food sources via a network of veins whose efficiency, cost, and robustness resemble and even surpass those of human-made rail networks [16]. To explore the intelligence mechanism of *Physarum polycephalum*, many researchers have established a similar model system by simulating its foraging behaviour and network construction process. The corresponding studies can be divided into several categories, as follows: (1) Tero et al. [27] built a mathematical model of *Physarum polycephalum* to solve mazes and construct networks by means of the Poiseuille law and Kirchhoff's law. (2) Gunji et al. [28] simulated the morphological behaviour of *Physarum polycephalum* and the formation of various

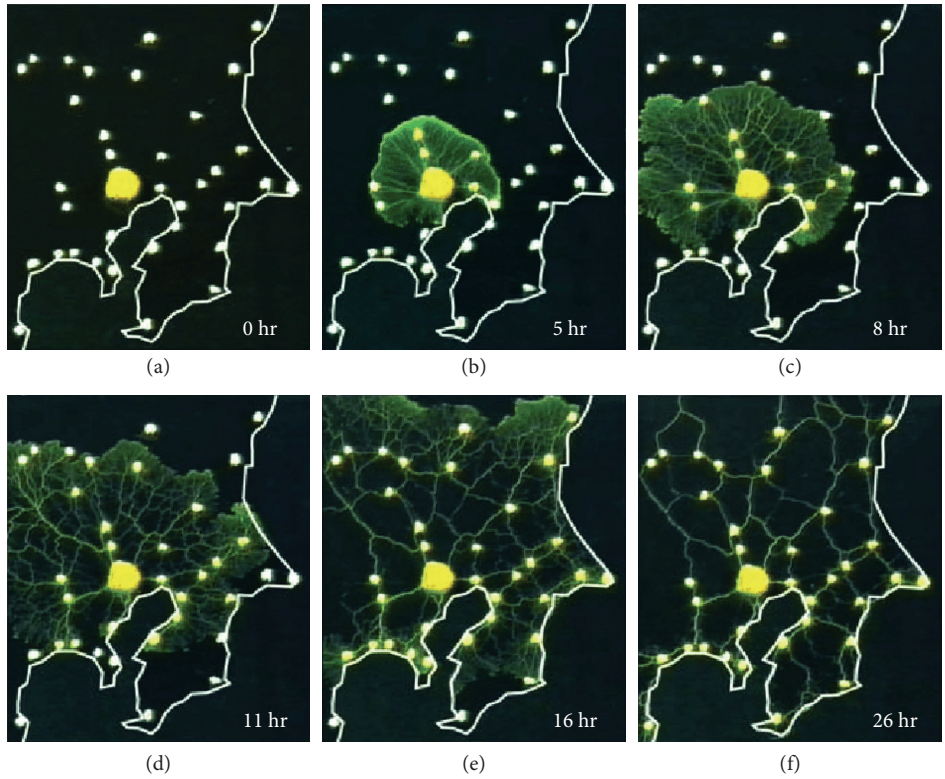

Figure 1: Network formation in *Physarum polycephalum* [16].

networks from the perspective of a cellular automata model. (3) Jones [23] established a multiagent model to simulate the dynamic evolution process and characteristics of *Physarum polycephalum*. Based on these models, many exploration problems can be effectively solved, including the TSP [29], path optimization problems [30], and maze solving problems [31]. These studies have fully demonstrated that the noncentral consciousness-driven network building ability exhibited by *Physarum polycephalum* can achieve remarkable performance in various exploration problems. Meanwhile, a multiagent system model inspired by *Physarum polycephalum* has also revealed the potential of using the mechanism of *Physarum polycephalum* in distributed systems.

2.2. Swarm Robotics. In swarm robotics, although a large number of swarm algorithms were used to swarm robot exploration problem, little attention has been paid to the path formation tasks under the exploration tasks. Even for exploration tasks, many algorithms still rely on centralized control and the accurate positioning system, and even the principal idea of swarm robotics of distributed decision making is neglected [32]. The path formation strategies for swarm robotics can be divided into the following categories: (1) Strategy based on a priori experience: Sperati et al. [22] and Motoaki et al. [33] trained a group of robots by a neural network model so that the robot explored the entire area and formed a path between the two specified locations. (2) Inspired by the behaviour induced by pheromones released by ant colonies, landmark-based navigation and path integration have also been exploited [13, 20, 21]. Each robot forms a communication network that extends in the environment, always keeping the connection with other robots. These methods often rely on a behaviour-based architecture. (3) Similarly, inspired by the pheromone trails observed in many ant species, many researchers tried to use different ways and implementing a robotic system that replicates the ants' foraging behaviour suffers from the complex problem of finding an alternative to the pheromones. For example, Mayet et al. investigated the use of fluorescent paint to simulate pheromones, given that the fluorescence activates and fades away with similar dynamics [34]. And a recent study investigated the use of computer (base control station, BCS) as a device to implement virtual pheromones. This computer stores the pheromone information, and the robots spread and sense such information while passing by [35]. This kind of strategy requires more complex sensors to assist the robot's perception of pheromones, including virtual pheromones.

The above these three kinds of studies are representative of a common methodology in the path formation tasks for swarm robotics and the researches about *Physarum polycephalum*. Most of the research studies start from a biological example, distill its relevant features, and transposes the identified mechanisms in the robotic system. In this paper, by combining the characteristics exhibited by *Physarum polycephalum* with the requirements of multirobot path formation tasks, we propose to simulate the

foraging behaviour of *Physarum polycephalum* by using multiple robots with limited functions to complete complex path formation tasks. Meanwhile, we compare the proposed algorithm with the group robot path formation strategy mentioned in (1) and (2) above in the experimental part. For the third type of strategy, we did not add it to our comparative experiment. This is because our main research in this paper is the path formation strategy of individual with extremely limited functions, and the strategy induced by virtual pheromone often requires more complicated sensors.

3. Task Model, Robot Model, and Challenges

3.1. Task Model. The task that we have chosen as a test bed to analyse our control algorithm is described as follows. A group of robots must search an unknown environment and form a path between two objects, called the nest and food. In our test bed, as described in Figure 2, we choose two special fixed robots as the nest, called the Robot_Nest, and the food, called the Robot_Food. These two special robots are far away from each other and cannot be directly sensed. The difficulty of the task can be modified by varying the distance between these two special objects and by placing obstacles in the environment.

3.2. Robot Model. For the robots, we use the marXbot robot [36] as the model. The marXbot is a ground-based robot that moves by means of a combination of wheels and tracks, as shown in Figure 3. And it has various sensors and actuators. But we focus on the multirobot collaborative search problem with limited robot capabilities. So, we have limited the functions of the robot's sensors and only use the basic functions as follows: (1) communication sensor of range R; (2) ranging sensor that detects the distances and the direction to robots within the communication radius; (3) proximity sensor, which is only used for obstacle avoidance in the bounded environment; and (4) simple movement by adjusting the speed of the two wheels.

3.3. Challenges. According to the above description of the task and robot model, the proposed control algorithm has the following challenges:

(1) The robot generally cannot acquire or calculate its own position. Even if it can obtain its own position, the robot cannot broadcast its position information to other individuals to induce them to search in a specific area or stop searching in a certain area in time because of the lack of a global communication system.

(2) The communication connectivity of a swarm system with only short-range communication capabilities will be greatly reduced and result in the failure of the search task when using some random search algorithms [37].

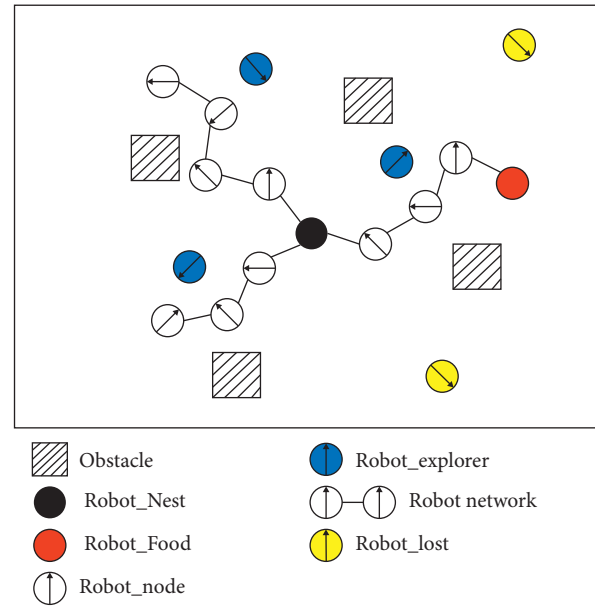

Figure 2: A network constructed by our control algorithm.

4. Algorithm Introduction

In this section, we first establish the mapping relationship between the foraging principle of *Physarum polycephalum* (as described in Section 2) and the multirobot system, as shown in Figure 4. We define three robot behaviour states and four execution mechanisms to implement our algorithm.

4.1. Behavioural States. Each robot is an independent execution unit, and we define three behaviour states, namely, Robot_Node, Robot_Explorer, and Robot_Lost, in accordance with different behaviour rules.

Robot_Node: the network formed by *Physarum polycephalum* is formed through the connection of multiple intersecting tubes. To simulate the formation of such a network, we first need to simulate the tubes. We refer to the locations where the tubes in the network formed by *Physarum polycephalum* intersect as nodes, and we define the Robot_Node state of a robot to simulate such a node (as shown in Figure 5). A robot that is in this state stops moving. In this way, two robots in the Robot_Node state can indirectly form an invisible connection through communication, similar to a tube in the *Physarum polycephalum* network. Multiple Robot_Nodes will form a network similar to that of *Physarum polycephalum*. Our previous work has also proven that such a network constructed by multiple robots provides an effective means to address the exploration problem for multiple robots with extremely simple functions and a limited communication range [13, 21].

Robot_Explorer: during the foraging process of *Physarum polycephalum*, the frequency and amplitude of the periodic contractions of the tube walls will increase when the organism encounters an attractant

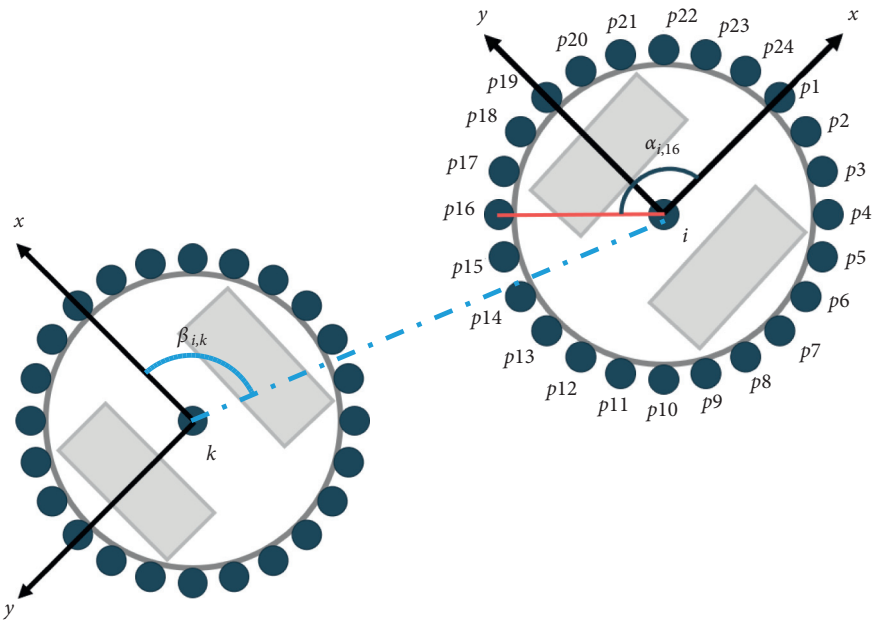

FIGURE 3: The kinematic model of the robots. The x direction identifies the front of the robot, $p_1 - p_{24}$ are the proximity sensors, $\alpha_{i,j}$ is the angle of the j^{th} proximity sensor, and $\beta_{i,k}$ is the angle between the i^{th} robot and the k^{th} robot.

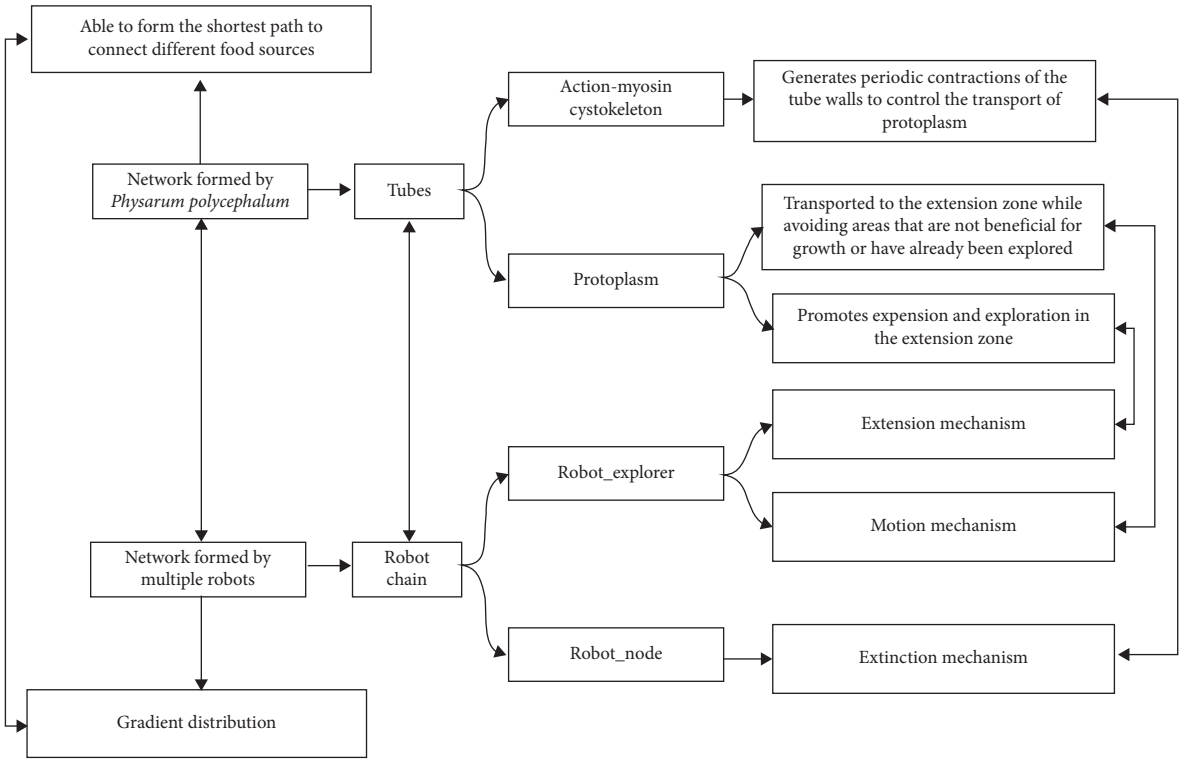

FIGURE 4: The mapping relationship between our controller and the foraging principle of *Physarum polycephalum*. Double arrows indicate correspondence.

(such as food), which attracts the protoplasm, and the protoplasm will engulf and digest newly encountered food sources [38]. By contrast, a repellent area (such as a dry area) is less attractive to the protoplasm. In other words, the foraging behaviour of *Physarum polycephalum* can be attributed to the transport of protoplasm to the extension zone. Accordingly, we define the Robot_Explorer state of a robot to simulate the protoplasm flow in *Physarum polycephalum*. As shown in Figure 5(b), a Robot_Explorer is guided by

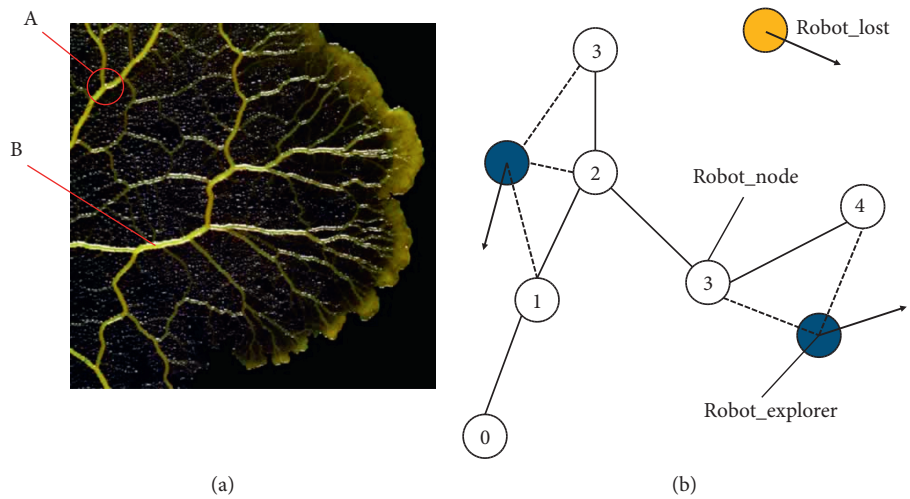

Figure 5: The networks constructed by *Physarum polycephalum* and swarm robots. (a) Created by *Physarum polycephalum*. A: a location where tubes intersect, called a node; B: one of the tubes that make up the network. (b) Created by swarm robots: the dotted and solid lines represent connections to the valid communication neighbours of a Robot_Explorer and a Robot_Node, respectively; the numbers represent the gradient value of each Robot_Node; and the arrows point in the direction of each robot's current movement.

the robot network constructed by Robot_Nodes, similar to the transport of protoplasm in the *Physarum polycephalum* network. Through transitions from the Robot_Explorer state to the Robot_Node state, the network constructed by the robots can constantly expand. In contrast, the robot network will retreat from areas that are not suitable for expansion or areas where the Robot_Food is not present. The specific transitions between the Robot_Node and Robot_Explorer states are described in detail in Sections 4.2.2 and 4.2.4.

Robot_Lost: different from the protoplasm of *Physarum polycephalum*, the robots have certain distinct physical properties. When multiple robots gather in a crowded area, collisions may occur, and a robot may lose communication with other robots. We define a robot that has lost communication with any Robot_Node as a Robot_Lost. In this state, a robot will walk randomly around the environment until it meets up with a robot in the Robot_Node state again. The specific random movement can be described as follows: the robot keeps moving in a straight line. At the same time, when encountering the boundary of the environment or with other robots in the Robot_Lost state, the robot changes the direction of movement to avoid collisions.

4.2. Execution Mechanisms. Our control algorithm is based on a behaviour-based architecture. Robots in different states will perform different execution mechanisms. All execution mechanisms are inspired by *Physarum polycephalum* as described below.

4.2.1. Gradient Distribution. The most prominent ability of *Physarum polycephalum* is its ability to form an efficient network structure. This ability enables it to connect to a food source along the shortest path and even solve the maze problem. Inspired by this phenomenon, we use a simple gradient distribution mechanism for the Robot_Nodes to find the shortest path when the Robot_Food can be connected by different paths.

The implementation of the gradient distribution mechanism can be described as follows: the Robot_Nest and Robot_Food each continuously broadcast a message with a fixed value of 0, indicating that they are gradient sources. Each robot in the Robot_Node state that receives this message will traverse the messages of neighbouring Robot_Nodes, the Robot_Nest, and the Robot_Food to obtain the minimum gradient value X, set its own gradient value to $X+1$, and then broadcast that gradient value. We refer to the Robot_Node with the minimum gradient X as the parent node (P_Node), whereas a corresponding node with a gradient of $X+1$ is called a child node (C_Node). The effect of this mechanism is shown in Figure 6. Since the Robot_Nest and Robot_Food are both gradient sources, two types of gradient values from different sources will be established in the robot network, as shown in Figure 6. To distinguish these two types of gradient values, we associate each of them with a different Source_Type. When each Robot_Node calculates its own gradient value, it will also update its own Source_Type to be the same as that of the neighbour with the minimum gradient value.

The purpose of gradient formation is to create a long-range sense of distance across the swarm system, thus

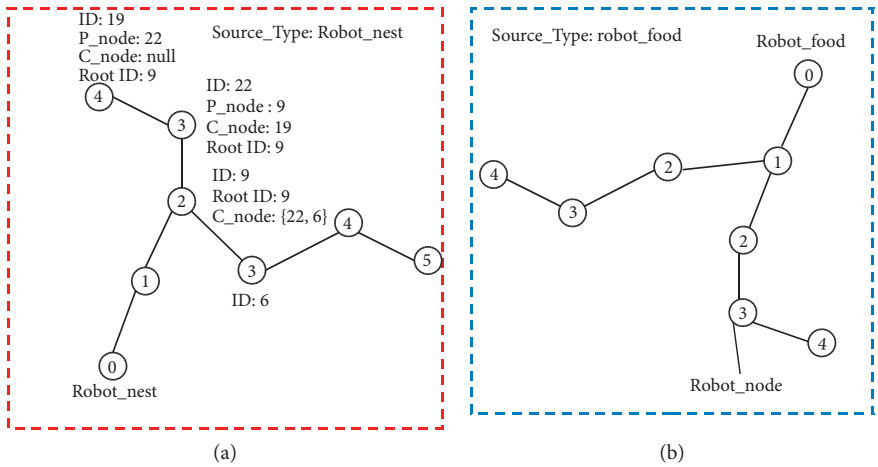

Figure 6: An illustration of the gradient mechanism. During the search process, two independent network structures will gradually be formed, as shown in the blue dotted box and red dotted box, where the numbers represent the gradient value of each Robot_Node.

providing a rough measure of the distance from the nest to the end of the network. In this way, the shortest path can be clearly and easily obtained when multiple paths are connected to the Robot_Food. Meanwhile, with the help of such gradient formation in the robot network structure, Robot_Explorers can determine their directions of movement in accordance with changes in gradient and whether they are at the end of the robot chain.

Moreover, based on the gradient distribution, each Robot_Node can also calculate the Root_ID of the branch to which it belongs. Specifically, each Robot_Node determines whether there are multiple child nodes surrounding it. If so, the robot will set the Root_ID to its own ID and broadcast it. Otherwise, it will read the Root_ID of the parent node and broadcast it. An example of this identification process is shown in Figure 6. In this way, the Robot_Nodes can know which branches of the network they are on, thereby ensuring that a Robot_Node that leaves the network due to the extinction mechanism will not rejoin the same branch chain. More details will be given in Section 4.2.2 (Algorithm 1).

4.2.2. Extinction Mechanism. When *Physarum polycephalum* senses an attractant, such as food, via specific binding to receptor molecules present on the outer membrane surface, this causes cytoplasm to flow towards the attractant. As it moves, *Physarum polycephalum* leaves behind a thick mat of nonliving, translucent, extracellular slime. This extracellular slime will cause the *Physarum polycephalum* to avoid foraging in the same area again [26]. This prevents repeated searches in areas that have already been explored. Similarly, we design an extinction mechanism for the Robot_Nodes to achieve dynamic control of the search direction of the robot system to avoid it becoming trapped in an undesirable part of the environment, such as a dead end in the maze problem.

The implementation of the extinction mechanism can be described as follows: the Robot_Node at the end of each network branch determines whether any Robot_Explorer is present nearby within a certain period of time (denoted by PEROID_TIME_1). If no Robot_Explorer is perceived throughout PEROID_TIME_1, this may mean that the branch is too long or has extended in the wrong direction, preventing any Robot_Explorer from moving to the end of the branch in a sufficiently short time. In this case, the Robot_Node is transformed into the Robot_Lost state and then joins in the network again in a different position once it encounters another branch. By contrast, if a Robot_Explorer is sensed during PEROID_TIME_1 but no child node generation is detected, this may mean that the network has extended to the edge of the environment. In this case, the robot will broadcast a message to tell each Robot_Explorer that receives the message to retreat along the branch and move to other branches. Moreover, if a Robot_Node that receives this message has the same Root_ID as the source of the message, it will also forward the message. After the Robot_Explorers have all left, the Robot_Node at the end of the branch will begin monitoring again. If no Robot_Explorer appears within a certain period of time (denoted by PEROID_TIME_2), the Robot_Node will transition into the Robot_Explorer state and also move back along this branch towards other branches. This mechanism is continuously applied until the shrinking branch chain retains only one Robot_Node to avoid repeated exploration (Algorithm 2).

4.2.3. Motion Mechanism. In Section 2, we analyzed the foraging principle of *Physarum polycephalum*, which can be summarized as follows: under the periodic contractions of the tube walls, protoplasm is transported to the extension zone, where it promotes the movement and growth of the organism in the direction of food. Accordingly, we design a movement mechanism to simulate the transport of protoplasm through the pipeline.

The extension zone to which the protoplasm is transported is near the food source. In our experiments, the robots have no a priori knowledge about the Robot_Food location. However, based on the gradient distribution mechanism described above, Robot_Explorers can guide the extension of the robot swarm by reading the gradient values

```
loop
gradient_value (self) ← GRADIENT_MAX
SOURCE_TYPE(self) ← Robot_Food
//Store the minimum gradient value from Robot_Nest in the neighbor neighbor_min_gradient_1 ← GRADIENT_MAX
//Store the minimum gradient value from Robot_Nest in the neighbor neighbor_min_gradient_2 ← GRADIENT_MAX
for all neighbors n do
  if SOURCE_TYPE (n) ← Robot_Nest then
    if (then neighbor_min_gradient_1 > gradient_value (n))
      neighbor_min_gradient_1 ← neighbor_min_gradient_1
  else
    if (then neighbor_min_gradient_2 > gradient_value (n))
      neighbor_min_gradient_1 ← neighbor_min_gradient_2
//Update the gradient and SOURCE_TYPE
if neighbor_min_gradient_1 ≠ GRADIENT_MAX then//exist the gradient from Robot_Nest
  SOURCE_TYPE (self) ← Robot_Nest
  gradient_value (self) ← neighbor_min_gradient_1 + 1
  transmit gradient_value (self)
else if neighbor_min_gradient_2 ≠ GRADIENT_MAX then
  SOURCE_TYPE (self) ← Robot_Food
  gradient_value (self) ← neighbor_min_gradient_2 + 1
  transmit gradient_value (self)
```

Algorithm 1: Gradient mechanism. GRADIENT_MAX is infinity.

```
counter_l ← 0//counterl is used to record the time when Robot_Explorer does not exist around
counter_2 ← 0//counter2 is used to record the time when Robot_Explorer exists around
Flag_Contraction ← False//Flag_Contraction is used to determine whether this branch is in a contracted state.
loop
if C_Node = null then//this Robot_Node is at the end of the branch.
  for all neighbors n do
    if state(n) = Robot_Explorer then
      counter_2 ++
      counter_1 = 0
      break
  if Do not exist Robot_Explorer in all neighbors n then
    counter_l ++
    counter_2 = 0
//Robot_Explorers can expand on this branch but they do not exist at the end for a long time.
  if counter_l > PERIOD_TIME_1 and Flag_Contraction = False then
    state ← Robot_Lost
//No child nodes generated for a long time
  if counter_2 >PERIOD_TIME_2 and Flag_Contraction = False then
    Flag_Contraction = true
    transmit Flag_Contraction (self)//pass the message that this branch is in a contracted state.
    state ← Robot_Explorer
//Robot_Explorers cannot expand on this branch and they do not exist at the end of it.
  if counterl. >PERIOD_TIME_1 and Flag_Contraction = true then
//Keep the last Robot_Node as a marker. Do not repeat the search in this area.
  if Root_ID(self) ≠ ID(P_Node) then
      state ← Robot_Explorer
else//not at the end of branch
  for all neighbors n do
    if ID (n) = C_Node and Root_ID (n) = Root_ID (self) then
      Flag_Contraction (self ← Flag_Contraction (C_Node)
      transmit Flag_Contraction (self)
      break
```

Algorithm 2: Extinction mechanism.

broadcast by the Robot_Nodes and moving to the positions with the largest gradient values, that is, the ends of the network branches. In this way, the Robot_Explorers can constantly swarm to areas that have not yet been explored. We propose a mathematical model that can be used by a Robot_Explorer to calculate the direction for its next step based on the gradient distribution, as shown in Figure 7.

The model is divided into two terms, as shown in equation (1). The first term, $\overrightarrow{P}_{n,i}$, represents the direction of the "extension zone," which is the end of the network branch. For this term, we are inspired by the research by Ke et al. [39]. These authors modelled the chemical queues as a scalar field using singular potentials located at each local food source and proposed a mathematical model for calculating the potential strengths from different food sources for *Physarum polycephalum*. We modify the formula to allow a Robot_Explorer to calculate the attractive or repulsive forces from the gradient distribution of the current surrounding Robot_Nodes, as shown by the red and purple arrows in Figure 7. The second term, $\overrightarrow{P}_{o,i}$, represents the obstacle avoidance direction, which is the direction pointing away from a nearby obstacle as calculated from the data and directions of all proximity sensors.

$$\overrightarrow{P}_{\mathrm{sum},i} = (1 - V_{\max})\overrightarrow{P}_{n,i} + V_{\max}\overrightarrow{P}_{o,i}, \tag{1}$$

where

$$\begin{aligned} \overrightarrow{P}_{n,i} &= \sum_{n=1}^{N} \gamma (G_n - \overline{G}) \ln D \overrightarrow{R}_n, \\ \overrightarrow{P}_{o,i} &= \left(\frac{\sum_{j=1}^{S} -V \overrightarrow{P}_j}{S} \right). \end{aligned} \tag{2}$$

Here, $\overrightarrow{P}_{\mathrm{sum},i}$ represents the direction of the next step of robot i, considering the influences from neighbouring robots, $\overrightarrow{P}_{n,i}$, and obstacles, $\overrightarrow{P}_{o,i}$. N represents the number of neighbouring Robot_Nodes, including the Robot_Nest. γ represents the contraction state of the branch, as computed in accordance with the pseudocode for the extinction mechanism. If the value of Flag_Contraction is false, then γ is 1; otherwise, γ is -1. G_n is the gradient value at the n^{th} neighbouring Robot_Node or Robot_Nest, and $\overline{G}$ is the mean of the G_n values. D represents the distance between Robot_Explorer i and its n^{th} neighbour. $\overrightarrow{R}_n$ is the direction vector of the n^{th} neighbour. S represents the number of proximity sensors. V represents the reading of the j^{th} proximity sensor, which takes values in the range $[0, 1]$. The value increases as the robot approaches an object. $\overrightarrow{P}_j$ is the direction vector of the j^{th} proximity sensor. In addition, we use $V_{\max}$ and $(1 - V_{\max})$ as weighting factors to balance the influence of $\overrightarrow{P}_{n,i}$ and $\overrightarrow{P}_{o,i}$ on the direction of movement. $V_{\max}$ denotes the maximum value of V. In this way, the closer the robot is to an obstacle, the more the robot's movement is dominated by obstacle avoidance. Otherwise, the main guidance is provided by the neighbouring Robot_Nodes.

After obtaining its next direction of motion $\overrightarrow{P}_{\mathrm{sum},i}$, a robot can calculate the differential speed of its two wheels to move in the desired direction as shown in Algorithm 3.

4.2.4. Extension Mechanism. We use the extension mechanism to implement the state transition from the Robot_Explorer state to the Robot_Node state in order to expand the interrobot communication network, similar to the expansion of the *Physarum polycephalum* network.

When a Robot_Explorer moves to the end of a network branch, it will determine whether the current branch can be expanded and whether the distance between its current position and that of the Robot_Node at the end of the branch meets the requirements for extension (Algorithm 4).

4.3. Algorithm Flow. The probabilistic finite state machine (PFSM) of the individual robot behaviours is presented in Figure 8. All robots are initially in the Robot_Lost state and are located either at random positions in the exploration environment or within a specific area. Typically, each robot does not initially perceive the Robot_Nest or Robot_Food and therefore performs a random walk until it senses one of them. At this time, the robot transitions from the Robot_Lost state to the Robot_Explorer state and begins to move in accordance with the motion mechanism (see Section 4.2.3). The Robot_Nest or Robot_Food can be regarded as a root of the robot network. When a Robot_Explorer meets the requirements of the expansion mechanism, the robot will either start a new branch or follow an existing branch. Thus, the robots will gradually form two independent and unconnected network structures with the Robot_Nest and Robot_Food as the sources. Through the gradient distribution mechanism, we can ensure that each robot is aware of which robot network it is part of. Moreover, each Robot_Node at the end of a branch will selectively leave that branch depending on the presence of Robot_Explorers in its surroundings (see Section 4.2.2) and thus begin to expand other chains. The process of joining/leaving a branch is fundamental to the exploration of the environment, as it allows the formation of new branches in unexplored areas. During the above process, when a robot loses communication with other Robot_Nodes due to collision or other reasons, it will transition into the Robot_Lost state and perform a random walk until it perceives a Robot_Node or the Robot_Nest again. As the robot network continues to expand, the two independent networks continue to approach each other until these two networks become connected and merge to form a network connecting both the Robot_Nest and the Robot_Food. At this time, the task is complete.

5. Experiments and Discussions

The goal of our experimental activity was to evaluate our algorithm under different experimental conditions and to compare them with other algorithms. Section 5.1 utilizes a

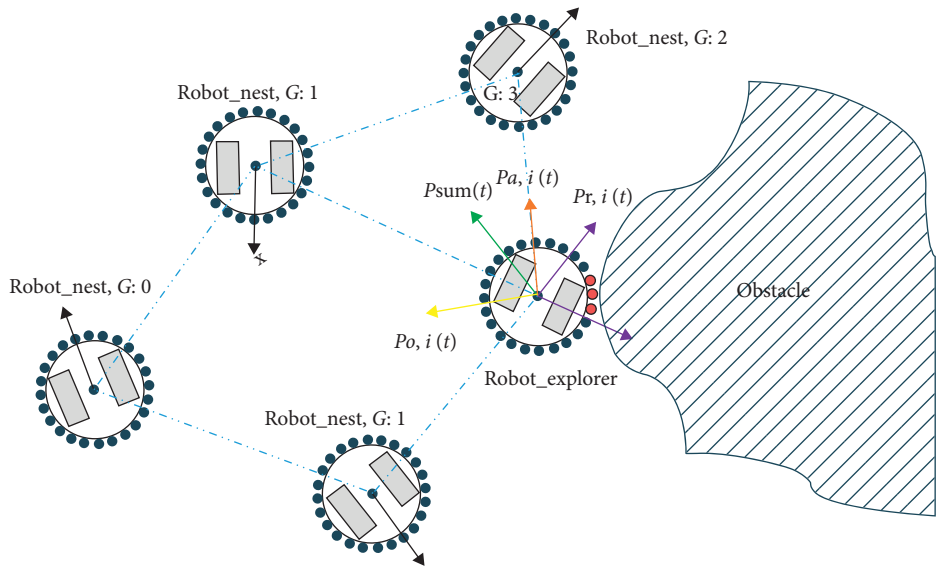

Figure 7: Example of the motion mechanism. The dash-dotted lines represent the interrobot communication network. The purple arrows $\overrightarrow{P}_{r,i}(t)$ represent the repulsive forces of Robot_Nodes with a gradient value of 1 on Robot_Explorer i in the t^{th} time step. The red arrow $\overrightarrow{P}_{a,i}(t)$ represents the attractive force of a Robot_Node with a gradient value of 2. $\overrightarrow{P}_{o,i}(t)$ is the obstacle avoidance direction. $P_{\text{sum}}(t)$ is the sum of all the above vectors.

Input: $\overrightarrow{p}_{\text{sum}}$:the next direction of motion;
//calculate the angle between the direction of the current direction and the $\overrightarrow{p}_{\text{sum}}$ direction
angle ← atan2 ($\overrightarrow{p}_{\text{sum}}.y$, $\overrightarrow{p}_{\text{sum}}.x$)
Left_Velocity ← 0
Right_Velocity ← 0
if angle >0 **then**
  Left_Velocity = m_fWheelVelocity * (1 − 2*angle/π);
  Right_Velocity = m_fWheelVelocity
else
  Left_Velocity = m_fWheelVelocity
  Right_Velocity = m_fWheelVelocity * (1 − 2*angle/π);
Set wheel speed (Left_Velocity, Right_Velocity)

Algorithm 3: Vector to wheel velocity.

(1) loop
(2) N_Num ← 0//N_Num will store the number of Robot_Nests and Robot_Nodes in neighbors.
(3) Object_N ← MAX_DISTANCE//Object_N will store the neighbor object.
(4) **for** all neighbors n **do**
(5) **if** state(n) = Robot_Node **or** state(n) = Robot_Nest **then**
(6) N_Num = N_Num++
(7) Object_N ← n
(8) **if** N_Num = **1** **then**//only one Robot_Node or Robot_Nest can be sensed.
(9) **if** Flag_Contraction(Object) ≠ TRUE **then**//the branch status requirements.
(10) **if** measured_distance(Object_N) > B_UL **then**//the distance requirements.
(11) state ← Robot_Explorer

Algorithm 4: Extinction mechanism. B_UL represents the distance between the Robot_Nodes.

maze environment to explore the ability of maze solving and path reconfiguration of our algorithm. In Section 5.2, we compare the performance of our algorithm with the PFIAS algorithm [20] and our previously proposed TSASR algorithm [21], including the efficiency, success rate, and stability. Meanwhile, we also do a serve of robustness tests to analyse the stability of our algorithm. Above all the experiments were performed on an ARGoS simulation platform [40].

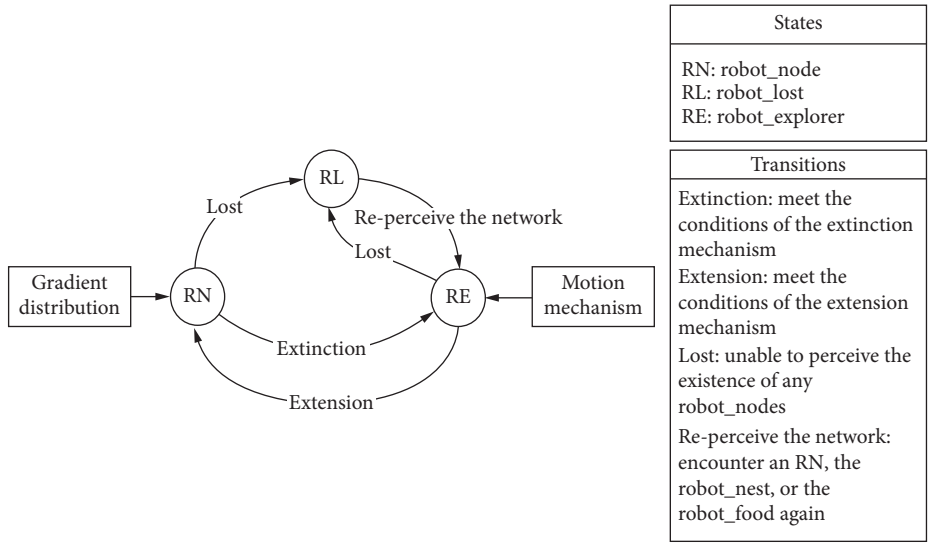

Figure 8: The probabilistic finite state machine (PFSM) of the individual robot behaviours.

5.1. Experiments in a Maze Arena. Our algorithm is a multirobot exploration algorithm inspired by the foraging behaviour of *Physarum polycephalum* with outstanding performance in solving maze problems. So we refer to the experimental maze environment used to test the P-MAS algorithm [20] and design an identical experimental maze environment on our simulation platform to verify the feasibility of our algorithm for solving the maze problem. To more closely simulate the foraging behaviour of *Physarum polycephalum*, we also limit the behaviour of the proposed algorithm in this experimental environment. For example, when a Robot_Explorer or Robot_Lost perceives the Robot_Food, it will not respond unless at least one Robot_Node also exists nearby. In this way, the robot network will be formed only with the Robot_Nest as the source, not with both the Robot_Nest and Robot_Food as sources. This weakens the performance of our algorithm to some extent because robot branches with the Robot_Food as the source have a more direct effect on achieving the exploration task.

The parameters used in the simulation experiment (Table 1) are different from those in Section 4.1. Moreover, all robots are collectively placed in a designated area (waiting area). Therefore, all of the robots need to leave this waiting area in accordance with the Lennard–Jones (LJ) model before they can begin to perform the search task. The process can be described as follows. Initially, the robots wander in the waiting arena (as shown in Figure 9(a)). Once some of the robots perceive the Robot_Nest, these robots switch to the "exiting" state, while in this state, the robots are attracted to the Robot_Nest, maintain a safe distance from other exiting robots based on the LJ model, and broadcast their distances from the Robot_Nest to nearby robots. In the meantime, other robots detect the presence of robots in the "exiting" state and switch to the same state. In this way, within a few steps, the entire swarm will transition to the "exiting" state, with the exception of the landmark robot. The robots that cannot perceive the Robot_Nest directly are attracted to their neighbours that are broadcasting the shortest distances to the landmark. The net effect is that the entire swarm rushes towards the exit of the waiting area, where the Robot_Nest is located, as shown in Figures 9(a)–9(c).

5.1.1. Simulation Process and Analysis. In Figure 9, we show the simulation process with a group size of 70. After the robots follow the LJ method to leave the waiting area as described above, the robots follow our algorithm to explore the area, continuously form a robot network, and eventually connect to the Robot_Food. Once the Robot_Food is found, the robots will continue to explore other areas that have not been explored. If there are multiple paths connected to the Robot_Food, the gradient mechanism (see Section 4.2.1) in our algorithm will allow the robots to determine the optimal path based on the gradient values. It is worth noting the behaviour observed in the red boxes in Figures 9(b) and 9(c). At the time represented in Figure 9(b), a large number of robots are gathered at the dead end marked by the red box. However, in Figure 9(c), the number of robots in the dead end has obviously decreased, indicating that they have moved to other areas for exploration. This phenomenon can be attributed to our extinction mechanism (see Section 4.2.2).

5.1.2. Group Size Tests. In these tests, we also use the maze arena, again with an obstacle-free environment. A summary of the results is given in Figure 10, where the success rates for each condition from left to right are 54%, 57%, 64%, 68%, 77%, 81%, and 88%. Figure 10 reports the corresponding completion times, and the performance of our algorithm improves as the group size increases. Specifically, for group sizes from $N = 40$ to $N = 55$, the search efficiency and success rate are low because the number of robots is too small to

Table 1: Parameter settings for simulation experiments in a maze arena.

Symbol	Meaning	Units	Value
$L \times L$	Search space	m × m	10×10
R	Communication radius	m	1.5
R	Detection radius	m	1.5
B_UL	Distance between the Robot_Nodes in a branch	m	$0.7^*R < B\_UL < 0.8^*R$
D	Distance between the Robot_Nest and Robot_Food	m	7.07
N	System size	Count	{40, 45, 50,55,60,65,70}
O	Number of obstacles	Count	Non
Speed	Robot movement speed	cm/s	20

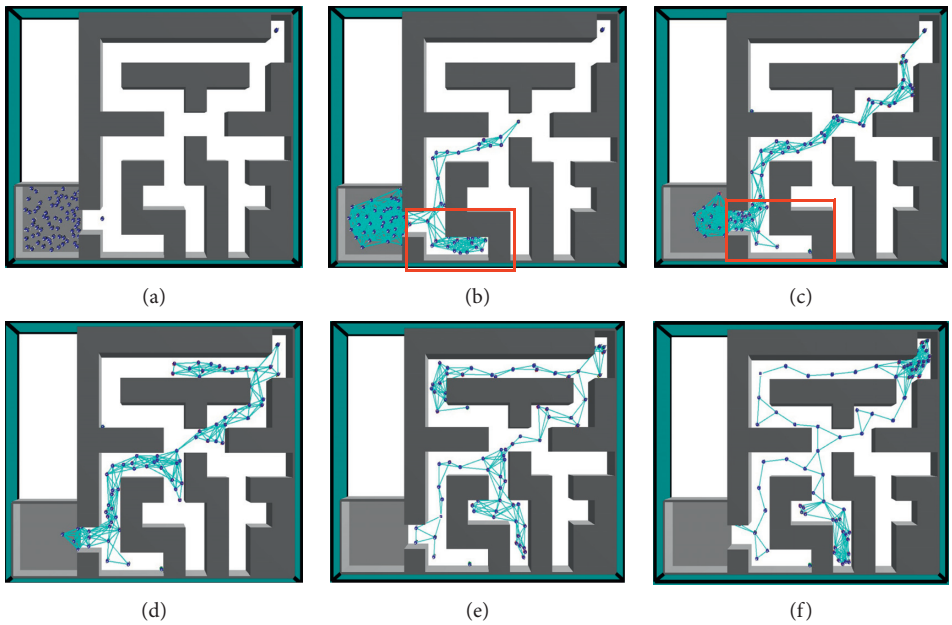

Figure 9: The simulation process. (a) 0 time steps. (b) 708 time steps. (c) 1418 time steps. (d) 2126 time steps. (e) 2835 time steps. (f) 3544 time steps.

effectively form a path connecting the Robot_Nest and Robot_Food. As the size of the group continues to grow, the maze environment tends to become crowded with moving robots, as shown in Figure 9, and the robot system can explore a larger area. Therefore, the performance gradually improves.

5.2. Experiments in a Square Arena. We employ a bounded arena of size 5 m × 5 m. The task consists in forming a path between the Robot_Nest and the Robot_Food in the environment. And the parameters used in the simulation experiment were the same as those used to test the PFIAS algorithm [20] and TSASR [21], as shown in Table 2. Based on these, we analyse the performance of the proposed algorithm and compare it with the PFIAS algorithm [20], TSASR [21], and NNEA [22]. Among them, NNEA is a neural network algorithm. We train the neural network in the experimental scenario where D is 1.8 m and N is 15 count. The parameters of the neural network controller—connections weights, biases, and time constants—are obtained using a genetic algorithm. The

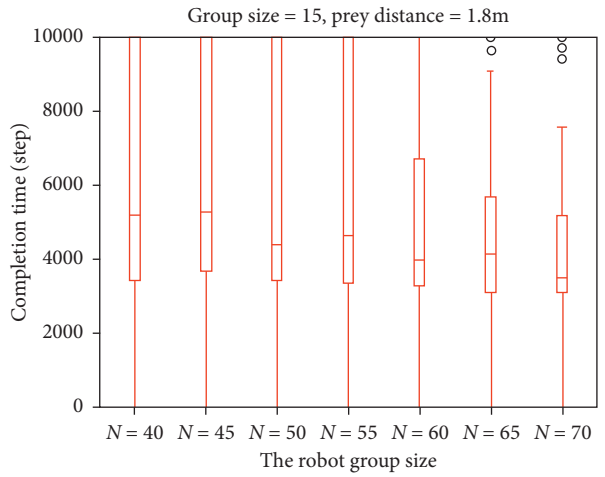

Figure 10: Box-and-whisker plots (Becker et al. 1998 [41]) of efficiency tests (100 observations per box) for seven robot group sizes in the maze arena. The boxes represent the interquartile ranges of the data, while the horizontal bars inside the boxes indicate the median values. The whiskers extend to the most extreme data points within 1.5 of the interquartile range from each box. The empty circles represent outliers.

Table 2: Parameter settings for experiments in a square arena.

Symbol	Meaning	Units	Value
$L \times L$	Search space	m × m	5×5
R	Communication radius	m	{0.6}
R	Detection radius	m	{0.6}
B_UL	Distance between the Robot_Nodes in a branch	m	0.7*R < B_UL < 0.8*R
D	Distance between the Robot_Nest and Robot_Food	m	{1.0, 1.2, 1.40, . . ., 3.0}
N	System size	Count	{5, 10, 15, 20}
O	Number of obstacles	Count	{0, 5, 10, . . ., 30}
Speed	Robot movement speed	cm/s	5

evolutionary process lasts 500 generations. For these four kinds of algorithm tests, we set the maximum allowable completion time to 1000 s at 10 steps/s, corresponding to a maximum of 10,000 steps. If the Robot_Food is not perceived during this time period, the test is terminated, and the experiment is considered a failure. For each combination of parameters, the experiment is repeated 100 times.

5.2.1. Difficulty Tests. To assess the performance with small groups, we conducted an experiment employing 15 robots in an obstacle-free environment. In this experiment, we measure the completion time when we vary the distance D from the Robot_Nest to the Robot_Food in the interval from 1 to 3 m. In addition, we compare the proposed algorithm with our previous work (the TSASR algorithm [21]), the PFIAS algorithm [20], and the NNEA algorithm [22]. The results are shown in Figures 11 and 12. Specifically, we analyse the search efficiency of the four different algorithms as the distance increases in Figure 11.

First, for the behaviour-based architecture algorithms (TSASR, PFIAS, and the EAIPP algorithm proposed in this paper). It is apparent that, for each distance, the green box is generally lower than the blue box, while the red box is generally at the lowest position. This indicates that the TSASR algorithm proposed in our previous work has higher efficiency than the PFIAS algorithm, while the algorithm proposed in this paper has the highest efficiency among these four algorithms. It is worth noting that the efficiency of the TSASR algorithm is higher than the efficiency of the algorithm proposed in this paper when the distance D is in the range of [1.0, 1.8] m. However, as the distance increases, the efficiency of the algorithm proposed in this paper gradually exceeds that of the TSASR algorithm. This observation demonstrates the advantage of the proposed algorithm for long-distance tasks. In the data for $D = [2.4, 2.6, 2.8, 3.0]$, it is observed that the blue and green boxes abruptly shorten and even disappear. This is not because the search with the PFIAS algorithm becomes faster or more stable. Instead, the search success rate of the PFIAS algorithm is too low for tasks at these distances, and all failed experiments are recorded as requiring 10000 steps in Figure 11. In Figure 12, we show the number of tests in which the search task is successfully completed, divided by the total number of tests (100 observations per experimental condition). Compared with the other algorithms, our proposed algorithm has the highest success rate, especially for long-distance tasks.

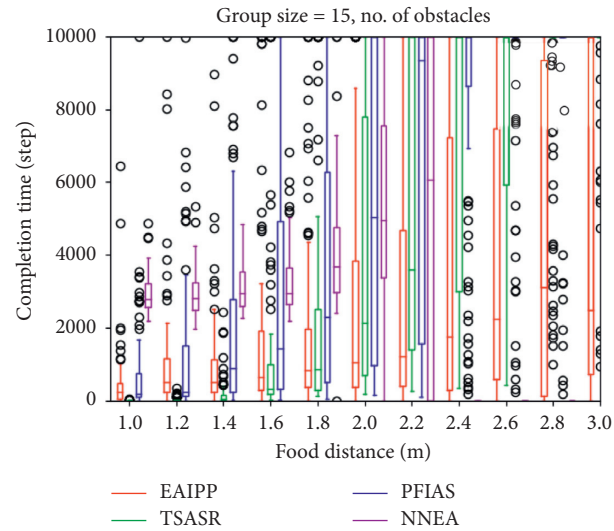

Figure 11: Box-and-whisker plots (Becker et al. 1998 [41]) showing 100 evaluations per box of the completion times with varying Robot_Nest-to-Robot_Food distances for a group of 15 robots in an environment without obstacles. See Figure 10 for an explanation of box-and-whisker plots.

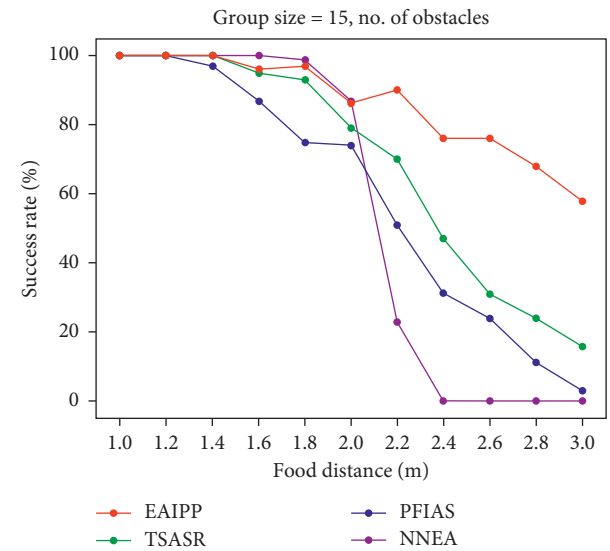

Figure 12: Effect of task difficulty on the success rate for a group size of $N = 15$.

Secondly, for neural network evolution algorithm (NNEA) based on prior knowledge, it can be clearly seen that the stability of the purple box is in the time range of 2000–4000 when the data are $D = [1.0, 1.2, 1.4, 1.6]$. Although the efficiency of the NNEA algorithm represented by the purple box is lower than the efficiency of the other three algorithms at this time, the success rate of the NNEA algorithm is actually stable and higher than the other three algorithms as shown in Figure 12. However, as the distance continues to increase, the success rate of the NNEA algorithm drops sharply. This also caused the purple box to disappear in Figure 11. The reason why the success rate of the NNEA algorithm drops sharply is that the NNEA algorithm analyzed in Figure 11 is based on the training of the neural network with the distance $D = 1.8$ m. Therefore, when the distance D is greater than 1.8 m, the prior knowledge formed by the neural network will no longer play a better role in the formation of the path. Compared with the other three algorithms, the NNEA algorithm has poor adaptability to the change of distance D.

5.2.2. Obstacle Tests. In order to access the performance in the presence of obstacles, we tested our algorithm by the standard arena with a random configuration of obstacle cylinders. In this test, we also compared with the PFIAS algorithm [20], the TSASR algorithm [21], and the NNEA algorithm [22].

Figure 13 shows the results of an obstacle experiment for a group of 15 robots. The distance D is 3 m in all cases. As the number of obstacles increases, the difficulty of the task increases in several ways. First, finding the Robot_Nest or Robot_Food becomes more difficult because it might be hidden behind obstacles. Second, obstacles between robots block communication. Therefore, the existence of obstacles may cause communication failure between robots, thereby reducing the communication efficiency of the group system. Third, it might be impossible to form a straight path connecting the Robot_Nest and Robot_Food. This increases the length of the shortest path and presents difficulties in the alignment and movement strategies. The results show that in general, our controller is capable of coping with obstacles. Although the completion time increases with more complex configurations, the task can still be completed efficiently and stably. By contrast, the TSASR and PFIAS algorithms perform worse than EAIPP, as also proven by the effect of obstacles on the task success rates shown in Figure 14. It is clear from the data shown in Figure 14 that the success rate of the EAIPP algorithm remains above 90% in every case. In contrast, there is a dramatic drop in the success rates of the PFIAS and TSASR algorithms. Among these four algorithms, the most obvious decline in search success rate and efficiency is still in the NNEA algorithm based on a priori knowledge. Although the NNEA algorithm has a higher success rate and stability than the other three algorithms when the distance is $D = 1.8$ m, it also has a significant decrease in the overall search effect when the number of obstacles changes. The reasons for this phenomenon are similar to those described in Section 5.1.1. The neural network is trained in an environment without obstacles, so the

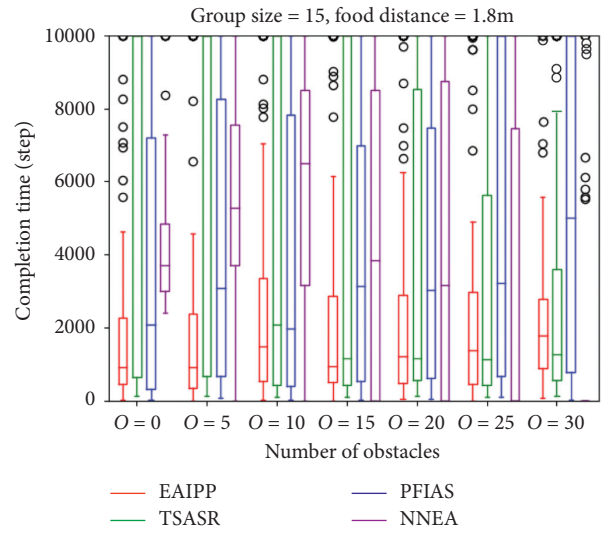

Figure 13: Box-and-whisker plots (Becker et al. 1998 [41]) of obstacle tests (100 observations per box) for a group of 15 robots and a food distance of 1.8 m. See Figure 10 for an explanation of box-and-whisker plots.

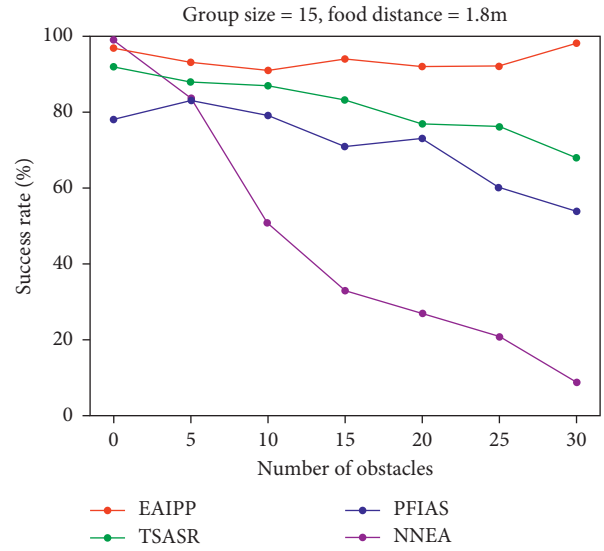

Figure 14: The effect of obstacles on the success rate of the task.

robots do not have a good ability to deal with the environment of obstacles. To achieve a better search effect in an obstacle environment, it is necessary to train the neural network in the obstacle environment. This also means that algorithms based on prior knowledge are poorly compatible with changing environments because different experimental environments require different neural networks.

5.2.3. Robustness Tests. We generate different noise data for different sensors and add them to the corresponding sensors. Specifically, we vary the noise in the directions to other robots perceived by the range-and-bearing system (Figures 15 and 16) and in the differential wheel speed (Figure 17).

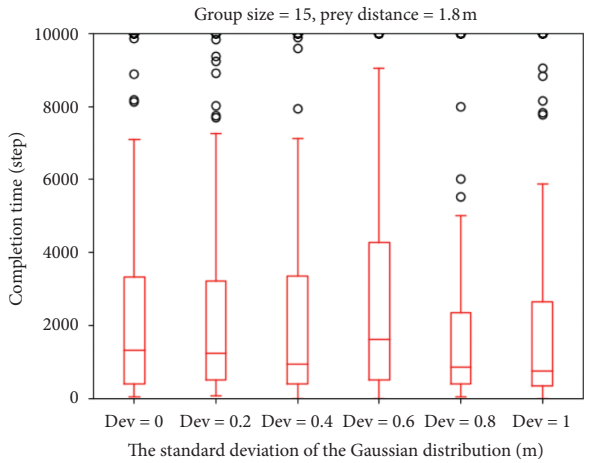

Figure 15: Box-and-whisker plots (Becker et al. 1998 [41]) of robustness tests for direction perception in the range-and-bearing system (100 observations per box) for a group of 15 robots and a food distance of 1.8 m. See Figure 10 for an explanation of box-and-whisker plots.

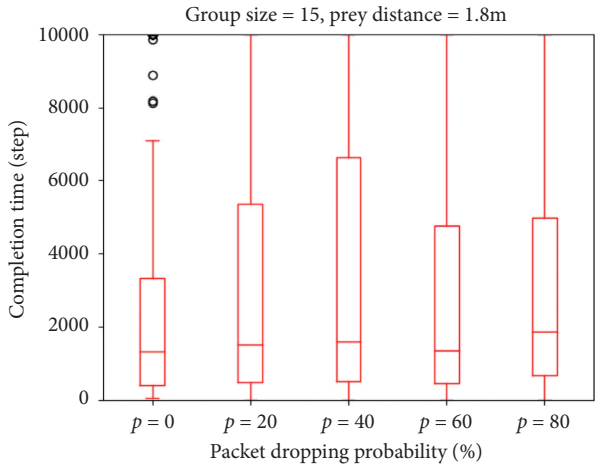

Figure 16: Box-and-whisker plots (Becker et al. 1998 [41]) of robustness tests for the packet dropping probability in the range-and-bearing system (100 observations per box) for a group of 15 robots and a food distance of 1.8 m. See Figure 10 for an explanation of box-and-whisker plots.

The noise is varied as follows:

(1) Regarding the directions to other robots, noise is implemented as a random vector added to the vector joining two communicating robots in the range-and-bearing system. For this random vector, the azimuth is chosen uniformly from the range [0 : 2PI], and the length is drawn from a Gaussian distribution. By adjusting the value of the standard deviation, we can obtain random vectors of different lengths. These random vectors are added to the perceived directions between robots. In Figure 15, we show the impact of the standard deviation of this random noise on the search task. It is clear that our algorithm can still maintain stable performance in the case of directional interference.

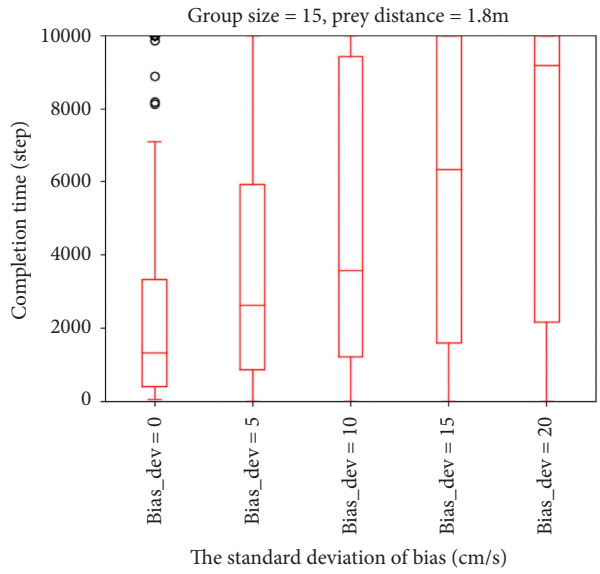

Figure 17: Box-and-whisker plots (Becker et al. 1998 [41]) of robustness tests for the differential steering of the robot wheels (100 observations per box) for a group of 15 robots and a food distance of 1.8 m. See Figure 10 for an explanation of box-and-whisker plots.

(2) Regarding packet loss, we assume that the robots have some probability of packet loss during communication with others. In this experiment, we specify the probability that a packet will be lost even though the robot should have received it. The results are shown in Figure 16, where the success rates for each probability condition from left to right are 92%, 86%, 83%, 89%, and 91%. We can see that although losing packets has an impact on the search speed to a certain extent, our algorithm is still able to complete the search task. In addition to the results in Figure 16, we also test the performance when the packet loss rate reaches 100%. The results show that the success rate drops to 0% in this case. This is because the group system can no longer exchange any information.

(3) Regarding the differential steering drive control, we increase the noise by controlling the differential wheel actuation using equation (3). In these tests, we analyse the effect of different standard deviations of the bias of differential steering, from 0 cm/s to 20 cm/s. Figure 17 shows the experimental results, where the success rates for each condition from left to right are 92%, 87%, 78%, 63%, and 45%. From this figure, we can see that differential wheel noise has a great impact on our algorithm. The time required to complete the task increases linearly with the standard deviation of the bias, and the success rate of task completion also plummets.

$$a = f^{*}(w + b), \tag{3}$$

where a is the actual actuation value, f is a random factor drawn from a Gaussian distribution, w is the ideal wheel actuation, and b is a random bias drawn from a Gaussian distribution.

6. Conclusion and Future Work

Path formation is a more challenging problem in multirobot exploration and navigation. The robots not only need to coordinate to find item sources in an unknown environment but also need to self-organize to form a path that connects item sources. In this paper, we succeed in designing a new method inspired by *Physarum polycephalum* for multirobot systems to perform path formation tasks. First of all, by analyzing the foraging mechanism of *Physarum polycephalum*, we can know that *Physarum polycephalum* grows through the surroundings and gradually forms an efficient tube network to connect different foods. It is worth noting that this entire process is the result of the feedback effect of *Physarum polycephalum* on its surrounding environment without any central control. Inspired by the characteristics of these *Physarum polycephalum*, we combine these characteristics with the swarm robots to solve the path formation task. The main contributions of our work are as follows: (1) For swarm robotics, we not only pioneered the multirobot path formation strategy but also analyzed and compared most of the existing multirobot path formation strategies, including strategies based on neural network training (NNEA algorithm [22]) and strategies based on behavioural state structure (PFIAS [20] and TSASR [21] algorithms). By comparison, our proposed algorithm has more obvious advantages in search efficiency and success rate. (2) For the researches about *Physarum polycephalum*, although a large number of methods were used to simulate the foraging principle of *Physarum polycephalum*, even some multiagent approaches were proposed to uncover the mysteries of *Physarum polycephalum* [19, 23]. However, in these studies, overall task completion was mainly driven by a computer mimicking the release of similar pheromones. In this paper, we revealed for the first time the feasibility of Polycephalus polycephalum and we also consider the robots' collision constraints and physical properties for multiagent system to simulate the foraging of *Polycephalus polycephalum* rather than only focus on the virtual agent individual and centralized computing.

In future work, we intend to enhance the proposed algorithm to improve the efficiency of the robot system and its coverage for area exploration and to test its ability to adapt to changes in dynamic environments.

Data Availability

The data used to support the findings of this study are included within the article. The experimental data such as success rate and completion time used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant number: 61703102), the National Natural Science Foundation of China (grant number: 51975121), the Department of Education of Guangdong in China (grant numbers: 2017KZDXM082 and 2018KTSCX224), and the Industrial-Academic-Research Cooperation Demonstration Base of DGUT-HengLi Town.

References

[1] A. Pandey, N. Bej, R. Kumar, A. Panda, and D. R. Parhi, "Type-2 fuzzy controller (T2FC) based motion planning of differential-drive pioneer P3-DX wheeled robot in V-REP software platform," *A Journey towards Bio-inspired Techniques in Software Engineering*, Springer, Cham, Switzerland, pp. 47–57, 2020.

[2] A. Pandey, A. K. Kashyap, D. R. Parhi, and B. K. Patle, "Autonomous mobile robot navigation between static and dynamic obstacles using multiple ANFIS architecture," *World Journal of Engineering*, vol. 16, no. 2, 2019.

[3] A. Pandey, V. S. Panwar, M. E. Hasan, and D. R. Parhi, "V-REP-based navigation of automated wheeled robot between obstacles using PSO-tuned feedforward neural network," *Journal of Computational Design and Engineering*, vol. 7, no. 4, pp. 427–434, 2020.

[4] M. Mohammad, P. Akshar, G. N. Vishnu, and K. R. Guruprasad, "Simultaneous exploration and coverage by a mobile robot," *Control Instrumentation Systems*, vol. 581, 2020.

[5] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[6] A. Hong, O. Igharoro, Y. Liu, F. Niroui, G. Nejat, and B. Benhabib, "Investigating human-robot teams for learning-based semi-autonomous control in urban search and rescue environments," *Journal of Intelligent & Robotic Systems*, vol. 94, pp. 669–686, 2019.

[7] C. Sampedro, A. Rodriguez-Ramos, H. Bavle, A. Carrio, P. de la Puente, and P. Campoy, "A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, pp. 601–627, 2019.

[8] X. Liu and Y. Tan, "Adaptive potential fields model for solving distributed area coverage problem in swarm robotics," *Lecture Notes in Computer Science*, Springer, Cham, Switzerland, pp. 149–157, 2017.

[9] J. M. Gregory et al., "Enabling intuitive human-robot teaming using augmented reality and gesture control," 2019, http://arxiv.org/abs/1909.06415.

[10] O. Olsson, J. S. Brown, and K. L. Helf, "A guide to central place effects in foraging," *Theoretical Population Biology*, vol. 74, no. 1, pp. 22–33, 2008.

[11] C. Detrain and J.-L. Deneubourg, "Collective decision-making and foraging patterns in ants and honeybees," *Advances in Insect Physiology*, vol. 35, pp. 123–173, 2008.

[12] A. Reina, R. Miletitch, M. Dorigo, and V. Trianni, "A quantitative micro–macro link for collective decisions: the shortest path discovery/selection example," *Swarm Intelligence*, vol. 9, pp. 75–102, 2015.

[13] Y. Luo, J. Guo, Z. Zeng et al., "Robot chain based self-organizing search method of swarm robotics," in *Proceedings of the International Conference on Intelligent Computing*, Wuhan, China, August 2018.

[14] L. Pitonakova, R. Crowder, and S. Bullock, "The information-cost-reward framework for understanding robot swarm foraging," *Swarm Intelligence*, vol. 12, no. 1, pp. 71–96, 2018.

[15] T. Nakagaki, H. Yamada, and Á. Tóth, "Maze-solving by an amoeboid organism," *Nature*, vol. 407, no. 6803, p. 470, 2000.

[16] A. Tero, S. Takagi, T. Saigusa et al., "Rules for biologically inspired adaptive network design," *Science*, vol. 327, no. 5964, pp. 439–442, 2010.

[17] W. Tang, K. Zhang, and D. Jiang, "Physarum-inspired routing protocol for energy harvesting wireless sensor networks," *Telecommunication Systems*, vol. 67, no. 4, pp. 745–762, 2018.

[18] Y. Liu, X. Feng, H. Yu, and F. Luo, "Physarum dynamic optimization algorithm based on energy mechanism," *Journal of Computer Research and Development*, vol. 8, p. 14, 2017.

[19] Y. Liu, C. Gao, Z. Zhang et al., "A new multi-agent system to simulate the foraging behaviors of Physarum," *Natural Computing*, vol. 16, no. 1, pp. 15–29, 2017.

[20] S. Nouyan, A. Campo, and M. Dorigo, "Path formation in a robot swarm," *Swarm Intelligence*, vol. 2, no. 1, pp. 1–23, 2008.

[21] Y. Luo, J. Guo, G. Ye et al., "Toward target search approach of swarm robotics in limited communication environment based on robot chains with elimination mechanism," *International Journal of Advanced Robotic Systems*, vol. 17, no. 3, 2020.

[22] V. Sperati, V. Trianni, and S. Nolfi, "Self-organised path formation in a swarm of robots," *Swarm Intelligence*, vol. 5, no. 2, pp. 97–119, 2011.

[23] J. Jones, "The emergence and dynamical evolution of complex transport networks from simple low-level behaviours," *International Journal of Unconventional Computing*, vol. 6, p. 2, 2010.

[24] C. Gao, C. Liu, D. Schenz et al., "Does being multi-headed make you better at solving problems? a survey of Physarum-based models and computations," *Physics of Life Reviews*, vol. 29, 2018.

[25] K. Alim, N. Andrew, A. Pringle, and M. P. Brenner, "Mechanism of signal propagation in *Physarum polycephalum*," *Proceedings of the National Academy of Sciences*, vol. 114, no. 20, pp. 5136–5141, 2017.

[26] C. R. Reid, T. Latty, A. Dussutour, and M. Beekman, "Slime mold uses an externalized spatial "memory" to navigate in complex environments," *Proceedings of the National Academy of Sciences*, vol. 109, no. 43, pp. 17490–17494, 2012.

[27] A. Tero, R. Kobayashi, and T. Nakagaki, "A mathematical model for adaptive transport network in path finding by true slime mold," *Journal of Theoretical Biology*, vol. 244, no. 4, pp. 553–564, 2007.

[28] Y.-P. Gunji, T. Shirakawa, T. Niizato, and T. Haruna, "Minimal model of a cell connecting amoebic motion and adaptive transport networks," *Journal of Theoretical Biology*, vol. 253, no. 4, pp. 659–667, 2008.

[29] J. Jones and A. Adamatzky, "Computation of the travelling salesman problem by a shrinking blob," *Natural Computing*, vol. 13, no. 1, pp. 1–16, 2014.

[30] J. Jones, "A morphological adaptation approach to path planning inspired by slime mould," *International Journal of General Systems*, vol. 44, no. 3, pp. 279–291, 2015.

[31] A. Adamatzky, "Slime mold solves maze in one pass, assisted by gradient of chemo-attractants," *IEEE Transactions on Nanobioscience*, vol. 11, no. 2, pp. 131–134, 2012.

[32] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, 2020.

[33] M. Hiraga, T. Yasuda, and K. Ohkura, "Evolutionary acquisition of autonomous specialization in a path-formation task of a robotic swarm," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 22, no. 5, pp. 621–628, 2018.

[34] R. Mayet, J. Roberz, T. Schmickl, and K. Crailsheim, "Antbots: a feasible visual emulation of pheromone trails for swarm robots," *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 84–94, 2010.

[35] M. S. Talamali, T. Bose, M. Haire, X. Xu, J. A. R. Marshall, and A. Reina, "Sophisticated collective foraging with minimalist agents: a swarm robotics test," *Swarm Intelligence*, vol. 14, no. 1, pp. 25–56, 2020.

[36] M. Bonani, V. Longchamp, S. Magnenat et al., "The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research," in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Taipei, Taiwan, October 2010.

[37] N. R. Hoff III, "Multi-robot foraging for swarms of simple robots," Doctoral thesis, Harvard University, Cambridge, MA, USA, 2011.

[38] A. Dussutour, T. Latty, M. Beekman, and S. J. Simpson, "Amoeboid organism solves complex nutritional challenges," *Proceedings of the National Academy of Sciences*, vol. 107, no. 10, pp. 4607–4611, 2010.

[39] L. Ke, K. Thomas, L. F. Rossi, and C.-C. Shen, "Slime mold inspired protocol for wireless sensor networks," in *Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, IEEE, Venezia, Italy, October 2008.

[40] C. Pinciroli, V. Trianni, R. O'Grady et al., "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[41] R. A. Becker, J. M. Chambers, and A. R. Wilks, "The new S language, A programming environment for data analysis and graphics," *The Economic Journal*, vol. 1, no. 401, 1988.