Chapter 5

Distributed Swarm Formation Using Mobile Agents

Yasushi Kambayashi, Ryotaro Oikawa and
Munehiro Takimoto

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.81028

Abstract

This chapter presents decentralized control algorithms for composing formations of swarm robots. The robots are connected by communication networks. They initially do not have control program to compose formations. Control programs that implement our algorithm are introduced later from outside as mobile software agents. Our controlling algorithm is based on the pheromone communication of social insects such as ants. We have implemented the ant and the pheromone as mobile software agents. Ant agents control the robots. Each ant agent has partial information about the formation it is supposed to compose. The partial information consists of relative locations with neighbor robots that are cooperatively composing the formation. Once the ant agent detects an idle robot, it occupies that robot and generates the pheromone agent to attract other ant agents to the location for neighbor robots. Then the pheromone agent repeatedly migrates to other robots to diffuse attracting information. Once the pheromone agent reaches the robot with an ant agent, the ant agent migrates to the robot closest to the location pointed by the pheromone agent and then drives the robot to the location. We have implemented simulators based on our algorithm and conducted experiments to demonstrate the feasibility of our approach.

Keywords: mobile agents, formation control, multiple robots, ant colony optimization, swarm intelligence

1. Introduction

In the last two decades, we have witnessed mobile robot systems that consist of multiple robots such as robots playing soccer games. In performing a soccer task, the team would consist of robots with several roles such as defenders, midfielders, forwards, and a goalkeeper. If their roles are fixed to specific robots, they would have to always move around pursuing

a ball in order to cover their positions. On the other hand, if each robot can have different roles, the robots closest to the most convenient positions could achieve suitable roles without futile movement. Thus, the control manner based on multiple roles contributes to suppressing total execution cost and energy consumption. The control manner based on multiple roles, however, requires switching several programs on each robot depending on the circumstance shared by all the robots, which makes the implementation of the system difficult. Introducing mobile agents into multiroot systems solves this problem.

Mobile agent system achieves both of the ease of implementation and the execution efficiency. Each role is assigned to a specific mobile agent, and each agent migrates to the robot that is closest to the suitable position. The assignment of a role does not involve any physical movement. Furthermore, our model assumes that any agents on robots commit suicide as soon as they finish their tasks, and agents are supposed to migrate to idle robots rather than busy ones. Those control manners can decrease the total execution cost and energy consumption of not only a single task but also several tasks in multirobot systems. We have shown the advantages of mobile agents in robot systems for several applications. They are searching targets [1, 2], transporting objects to a designated collection area [3], clustering robots [4], and serialization of robots [5, 6].

In this chapter, we focus our attention on the formation control of multiple mobile robots. It is one of the most important issues for multiple mobile robots in many circumstances. Especially when individual robot has limited abilities and a given set of tasks requires collective actions, the formation control plays a vital role in a multirobot system. For example, robots may aggregate for coordinated search, that is, rescue, collectively transporting large objects, exploring and mapping unknown area, or maintaining formations for area defense or flocking [7]. In these formation algorithms, each robot needs to have the information about the entire formation. Practically, however, it is not common, for each robot knows the entire information. In this chapter, we propose distributed formation algorithms using mobile agents.

We assume that in a field, there is a fleet of robots that may be not only used for the specific task such as the formation but also shared by various different tasks because any mission is introduced into the fleet of robots by mobile agents through Wi-Fi network. For example, some robots may work for rescue, when other robots may work for mapping the area, and furthermore, these tasks may change dynamically and also the other ones may be idle state. Our goal is to propose methods that work well in such an environment and have high robustness that enables the entire robot system to continuously work even if some robots break down. The robots have capabilities of moving, and sensing length and angle to other robots, while they do not have a capability of sensing their absolute locations such as GPS. Thus, each robot must manage its coordinates as well as the coordinates of other robots in some way.

Our multirobot system is designed to use robot resources efficiently, that is, utilizing idle robots, cooperating with inaccurate devices that may have sensor errors and movement errors, composing arbitrary formations that can have various densities inside and continuing the task when some robots break down except for computers and communication devices. Furthermore, we also propose approaches that can compose a formation without any distortion even if some robots get to be out-of-order including the computers and communication devices, although the formation may be incomplete.

In this chapter, we present a pheromone-based approach toward the multirobot system that has no central control. In our robot control system, we have implemented ants and pheromones as mobile software agents. We call the mobile software agents that drive the mobile robot ant agents and call the other agent that provides communication means pheromone agent. Each ant agent on the robot requires only local neighbor information of the formation. Each ant agent has the information of relative positions of neighbor ant agents in the target formation and attracts the neighbor ant agents to the positions with their driving robots using pheromone agents. Pheromone agents diffuse to surrounding robots through migration. The ant agent not only attracts other ant agents but also is attracted by other ant agents. The bidirectional attraction results in settling all the ant agents with their driving robots down to the objective positions in the target formation without either knowledge of the whole shape of the formation. Since the attracted ant agent repeatedly migrates among the robots in the direction to the robot nearest to the target position, and then drives the robot to the target position, the pheromone-based approach contributes to suppressing energy consumption of the robots and decreasing the duration time for convergence.

The structure of the balance of this chapter is as follows. Section 2 gives the background and related works. Section 3 describes about the mobile software agents in general as preliminary information. We have extensively used an optimization method, namely ant colony optimization (ACO); therefore, in Section 4, we describe the ACO-inspired behaviors, which our mobile software agents have. In Section 5, we present a pheromone-based distributed formation control algorithm. Finally, we conclude our discussion in Section 6. The contents of this chapter are partially presented at the 10th Jubilee IEEE International Symposium on Applied Computational Intelligence and Informatics, the 13th European Conference on Multi-Agent Systems, and the Intelligent Systems Conference 2017. They are found in [8–11].

2. Background and related works

We have proposed a framework for controlling intelligent multiple robots using higher-order mobile agents [12–14]. The framework helps users to construct intelligent robot control software by migration of mobile agents. Since the migrating agents are of higher-order, a mobile agent can move into another agent so that it introduces a new feature or overrides a specific feature. The control software can be hierarchically assembled while they are running. Dynamically extending control software by the migration of mobile agents enables us to make base control software relatively simple and to add functionalities one by one as we know the working environment. Thus, we do not have to make the intelligent robot smart from the beginning or make the robot learn by itself. They can send intelligence later as new agents. We have implemented a team of cooperative search robots in order to show the effectiveness of our framework. At the same time, we have demonstrated that our framework contributes to energy saving of multiple robots [1, 12]. Based on this approach, our control algorithms are also based on mobile agents and therefore have effectiveness of improving system efficiency and suppressing energy consumption.

Recently, algorithms that are inspired by behaviors of social insects such as ants to communicate to each other by an indirect communication called stigmergy are becoming popular [15, 16].

Upon observing real ants' behaviors, Dorigo, Gambardella, Birattari, and Stützle found that ants exchanged information by laying down a trail of a chemical substance called pheromone so that other ants can follow the trail. They adopted this strategy, known as ant colony optimization (ACO), to solve various optimization problems such as the traveling salesman problem (TSP) [16]. Deneubourg et al. have originally formulated the biology-inspired behavioral algorithm that simulates the ant corps gathering and brood sorting behaviors [17]. Wang and Zhang proposed an ant-inspired approach along this line of research that sorts objects with multiple robots [18]. Lumer and Faiesta have improved Deneubourg's model and proposed a new simulation model that is called ant colony clustering [19]. Their method could cluster similar objects into a few groups.

Mizutani et al. have proposed and implemented pheromones in ACO as mobile agents [4]. In their system, both ants and pheromones are mobile agents. Ant agents repeatedly migrate among robots for searching free robots corresponding to objects. Once an ant agent finds a free robot, it drives that robot to form a cluster of robots. The pheromone agent is generated by the ant agent when its driving robot reaches a cluster, and repeatedly migrates among robots for guiding other ant agents to drive robots to the cluster. Shintani et al. have improved the mobile agent-based algorithm to serialize multiple robots while they are self-collected [5, 6]. Our approaches are extensions of the mobile agent-based clustering. Each agent that drives a robot is guided by the pheromone agents.

In the formation research area with which this chapter deals, many kinds of swarm controlling methods have been proposed. Behavior-based strategy defines simple behaviors or motion primitives for each robot. Balch and Arkin proposed the behavior-based motor scheme control [20]. Virtual structure considers the entire formation as a rigid body. The motion of each robot is translated from the motion of the virtual structure, which is determined by the definition of the dynamics of virtual structure. Lewis and Tan proposed one of the pioneering works of virtual-structure strategy [21]. In leader-following strategy, some robots are selected as leaders, and the other robots follow the leaders. Das et al. proposed one of the most popular control techniques using a feedback linearization control method in leader-following strategy [22].

Some methods have a limited class of shapes that they can form. Unsal and Bay proposed a formation method that forms shapes of rings and circles [23]. In this approach, robots issue beacons and instruct other robots to remain the position and keep a certain distance. Mamei et al. proposed similar method that composes a crude polygon and uses message hop count instead of a proximity sensor [24]. Kondacs proposed an approach that can form a large class of shapes that is inspired by biological agents that grow and die, and this approach can self-repair [25]. They use global and local compilation to automatically generate an agent. Stoy and Nagpal present a related approach to 3D self-assembly using self-reconfigurable modular robot, where individual modules must remain connected [26]. Although this system generates a wide variety of formations, it cannot create solid shapes because agents may block each other and create internal holes that agents cannot reach. To avoid this, they focus on scaffolds and porous shapes only. Gordan et al. proposed a method that forms arbitrary shape sharing common coordinate system [27]. The method, however, involves significant central computation. Although Rubenstein and Shen proposed a method that can self-repair, it can form only polar shapes [28].

Jimming Cheng, Winston Cheng, and Nagpal proposed a formation generation algorithm using contained gas model in which robots act like a particle in a container [7]. This method can construct almost arbitrary shapes, but it cannot compose a formation that does not have space inside the shapes such as line formation. Moreover, although it can keep suitable distance between robots, the density of the robots inside the formation cannot control, which means that it is not impossible to increase density partially intentionally. Our approach assumes that the shape of the formation is given and the motions of robots are determined by pheromone.

In our formation approach, mobile agents are main actors to compose a formation, and robots are just physical vehicles maneuvered by the mobile agents. The robots are only means where mobile agents move to the target locations in the formation. Therefore, any traditional approaches can be implemented on our mobile agent framework and obtain the effectiveness of its improving efficiency and saving energy. Furthermore, a mobile agent has no volume, so that mobile agents do not block any other mobile agents. Thus, our approaches can form not only line drawings but also arbitrary solid shapes and even formations with vacant area inside.

3. Mobile software agent

Object management group (OMG) has defined a set of common facilities of agents, which is called mobile agent system interoperability facility (MASIF). MASIF defines agents as computer programs that work for humans or organizations. If we give a unified notation of agents without restricting it to computer programs, it can be said that agents are independent subjects that can affect an environment through observing and taking actions for it [29, 30]. The subject means a human, a robot, or software that can be regarded as an individual object. Furthermore, the independency of the subject means that the subject can take actions based on its own experiments and knowledge for its circumstance. That is, agents are independent and intelligent subjects that interact with the environment utilizing their own intelligence and actions and making decisions based on the observation.

Recently, object-oriented programming becomes a dominant programming paradigm, where data representation and operations for it are abstracted, and only interfaces for the operations are given. Software agents can be considered as extended objects that have abilities to achieve some missions or tasks. Generally, the software agents have some features as follows:

- Autonomy: They can independently take actions.
- Reactivity: They can take actions along changes of the environment.
- Proactivity: They can take appropriate actions considering the context or intention.
- Learning capability: They can take actions based on knowledge and experiments.
- High-level communication: They can communicate with agents or humans through specific or natural languages.
- Cooperation: They can attain cooperation with other agents or humans.

- Character: They have names, features, or objects that are distinguished from other agents.
- Mobility: They can migrate among computer through a network.

Mobile agents are a kind of software agents but can independently and actively migrate to another computer through a network while continuing its computations. The mobility enables the agents to asynchronously perform tasks and to suppress communication costs [29, 30].

Figure 1 shows a model of mobile agent system. In the model, a mobile agent on computer 1 is able to migrate to computer 2 directly through a network and continuously execute its own tasks on computer 2.

Efficient use of network band is one of the advantages of the mobile agent model. Mobile agents are able to migrate to another computer, locally access its resources, and come back to the original computer with only the result of computations. Also, mobile agents are able to utilize information of the destination load. In intermittently communicating environments, mobile agents are able to continue their own tasks even while disconnecting the communications. We can summarize the advantages of mobile agents as follows.

Suppressing communication costs: In server-client model, communication cost often becomes high because the communication between a server and a client has to keep connection. In mobile agent model, the connections are restricted to migrations of the agents among computers. That is, the connection is not necessary after the agents migrate to another machine. This feature of mobile agents contributes to suppressing communication costs.

Suppressing network load: The connection feature of mobile agents mentioned above also contributes to suppression of network loads.

Asynchronous processing: Each mobile agent is an autonomous entity, and therefore, its behaviors follow asynchronous processing.

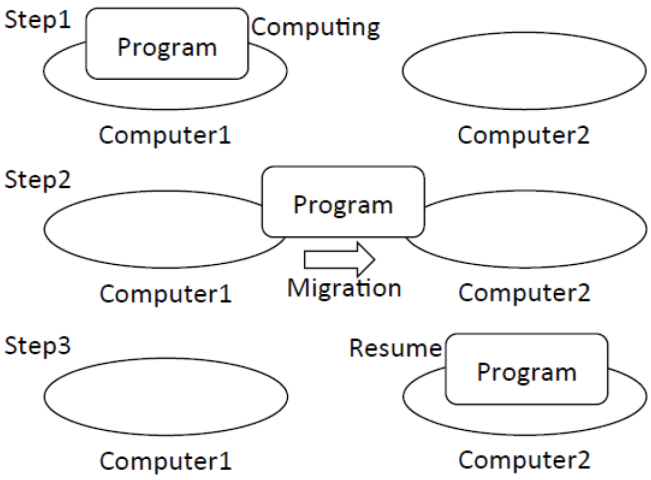

Figure 1. Mobile agent model.

Balancing loads: Each mobile agent can select the destination of migration autonomously. This means that it can contribute to load balancing, if it preferentially selects an idle machine as a destination.

An agent defined by MASIF has two kinds of states. They are (1) execution states and (2) attribute states. The execution states mean the states of context for executing any programs, such as counter or an execution stack. Attribute states are derived from the mobile agent's own data such as variables or arrays, which is decided by its mission, i.e., program. A mobile agent has to transfer not only program code but also these states to another site through migrations. This means that they have to be saved in a file as persistent data in order to transfer them to a destination site, and then, restored in loading the mobile agent's code at the destination.

Mobile agent systems often use a server system to implement their mobility. We also use server-based systems, AgentSpace and MobileSpace that were developed by Ichiro Sato. MobileSpace, which is an extension of AgentSpace, enables mobile agents to be composed hierarchically. Both systems are built using Java programming language. In Java, programs are not allowed to refer the program counter or the execution stack in a virtual machine. Thus, mobile agents implemented in Java are able to transfer only the values of their instance variables. AgentSpace and MobileSpace are middle wares for distributed computation implemented in Java language, which are free open source programs. These systems consists of an API part that makes construction of mobile agent applications easy and a run-time system part that gives services for execution and migration to mobile agents. These API and run-time system, which are shared by all the mobile agents, enable each mobile agent to be efficient and independent of the others [31]. AgentSpace and MobileSpace transfer their attribute states to the destination site with class files, but not execution sates. Some other systems also transfer execution states, which require costly process for each migration. We use AgentSpace to implement our agent-based distributed robot formation control systems.

4. ACO-inspired behavior

Our mobile agents have behavior inspired by ant colony optimization (ACO). In this section, we describe the details of the ACO-inspired behavior. A pheromone communication such as ant's route exploration can be modeled as a communication of multiagent system, for which various methods are studied. ACO is one of the most practical multiagent systems. ACO has been applied to solving a traveling salesman problem or network routing problem [32, 33]. ACO is a kind of metaheuristics that is used to obtain quasioptimal solutions independently of specific computational conditions. The metaheuristics try to resolve problems by introducing the following two notions:

Probabilistic action selection of an artificial ant agent: The random factor enables an agent not to repeat the same sequence of actions even if the same information is used. Namely, the controlling manner of agents enables agent to widely explore when the agents cannot decide which options to select, and to intensively explore when agent has confidence for which option to select.

Pheromone evaporation: This imitates the behavior of physical pheromones. Pheromones emitted by each agent evaporate over time. This prevents old information remain for a long time. Thus, the notion of pheromone evaporation keeps information fresh in ACO algorithm.

4.1. Basic concept of pheromone

In the science of ecology, pheromone is defined as chemical substance emitted by living things, which contributes to communications among the same species. For example, some pheromones attract other individuals that detect that pheromone, and some pheromones notify other individuals around them of danger. For both cases, the properties of evaporation and diffusion of pheromones are important. Evaporation contributes to decreasing older information over time. This property enables newer information to constantly be major. Diffusion contributes to expanding pheromones from their birth point to their surroundings. This property enables pheromones to propagate their information at some distance.

4.2. Pheromone communication

Ants maintain a social system by sharing some roles as if they are one individual. In the system, each individual takes actions for a whole swarm, and occasionally they devote their life to defend their swarm. Although each individual behavior seems to be randomly decided based on their judgment depending on various distributed and local factors, the whole swarm seems to suitably behave from the macroperspective. Many ants have restricted capability of sight, and in most cases, depend on the stimulus from tactile organ. Thus, they communicate not through visual information but through chemical signals sensed by the tactile organs. These signals are called pheromone and social insects such as ants communicate each other using the pheromone instead of languages or visual methods.

In ant colony, ants organize and control a swarm, using dozens of kinds of pheromones. Each individual emits a specific pheromone considering each situation. Then, distributed information through the pheromones is shared by other individual ants. Since this process is performed with time and spatial locality, it contributes to communications between individuals. Also, the shared information behaves a swarm organized by individuals as if each individual knows the optimal behaviors, so that the ant colony exhibits semioptimal behaviors.

4.3. Foraging of ants

Foraging is one of the most important tasks for worker ants to maintain their colony. The foraging process consists of exploring and transporting foods. Generally, the worker ants are divided into exploring ants and transporting ants. Once exploring ants find a food, they go back to colleagues in their nest through putting guidepost pheromone on their return path. After that, transporting ants go to the food following the guidepost pheromone using their tactile organs. Once transporting ants reach the food, they immediately catch the food and take it to their nest. At this time, transporting ants also put pheromones on their traces such as exploring ants. Although this sequence of actions seems to be simple, and indeed, the rules that each ant follows are simple. In seeing them as a swarm, however, the swarm shows a kind of intelligent behavior that seems to explore the shortest path between the nest and food.

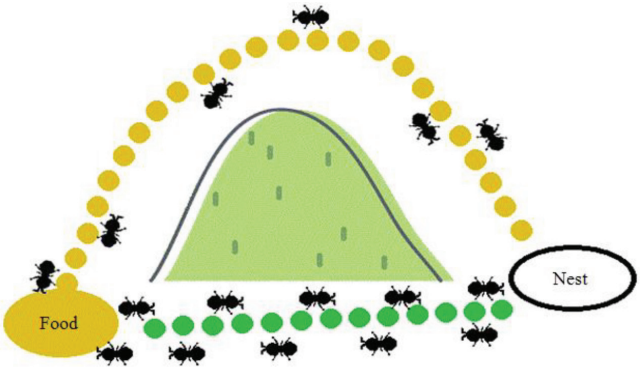

Figure 2. Ants take shorter path after initial search phase.

The observed intelligent behaviors for a swarm are collectively called swarm intelligence. For example, consider a branch path as shown in **Figure 2**. The first ant randomly selects a path at the branch. The second ant tends to select the path through which the first ant walked, because the first ant put pheromones on the path. Notice that the ants that select the shorter path reach a target earlier than the ants that select the longer path. Pheromones on the longer path evaporate earlier than pheromones in the shorter path, and then pheromones on the longer path spontaneously disappear, while pheromones in the shorter path become more stronger. As a result, the path that ants intensively select is the shortest path in high probability. Thus, strengthening phenomenon is positive feedback of route selection, which is the mechanism of ants' route selection. This mechanism achieves the optimization of detecting the shortest path without individual intelligence.

Suppose an ant k (in m ants) is at a junction i. Then, the ant chooses the direction to a junction j in a probabilistic way as follows:

$$p^k(i,j) = \frac{[\tau(i,j)]^\alpha [\eta(i,j)]^\beta}{\sum_{l \in N^i} [\tau(i,l)]^\alpha [\eta(i,l)]^\beta} \tag{1}$$

here, $\eta(i,j)$ represents the reciprocal of the distance between two junctions i and j. α and β are nonnegative constants, and their values are determined by whether one attaches importance to global information or local information. $\tau(i,j)$ represents the information added by ants, that is, the pheromone value on the route between two junctions i and j. This value is updated in the following rules, when one iteration is done.

$$\tau(i,j) \leftarrow (1-\rho)\tau(i,j) + \sum_{k=1}^{m} \Delta \tau^k(i,j) \tag{2}$$

$$\Delta \tau^k(i,j) = \begin{cases} 1/L^k & if (i,j) \in T^k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

here, T^k represents the set of all routes from source to destination, and L^k represents the collective distance from source to destination. In ACO algorithms, the pheromone value added to

a specific route is determined by the reciprocal of the total distance and the pheromone also disappears over time, where ρ represents the evaporation rate so that newly added pheromone has more influence than old pheromone additions. Thus, after many of iterations, one can expect to obtain quasioptimal route from the nest to the location of food.

4.4. Ants as multiagent system

As mentioned above, there are no supervisors that lead a whole swarm in ant colony systems. Although a queen ant is central character in a colony, the queen ant does not control worker ants and an individual worker ant that has multiple roles takes actions based on its locally perceiving information. On the other hand, a colony constantly maintains the state of ant colony system in certain quality as a whole.

In other words, each individual takes actions based on local action rules, changing the environment using pheromones, which accomplish route exploration indirectly. The notion of communication through mutual interactions based on the change of an environment is called stigmergy in ethology. This word, which roots from stigma (means sign) and ergon (means work), means obtaining a next action sign from work. That is, stigmergy is an indirect communication that tunes cooperative operation among individuals. In ant colony, the process where ants emit pheromone based on perceived local information and other ants perceive it feedback the change of an environment around ants to the ants, so that it enables the ants to adapt to complicated situations. **Figure 3** shows the system of ants as multiagents.

The generality and flexibility of ant colony are often applied to the design of artificial systems. The systems are called multiagent systems, which are actively studied in fields such as artificial intelligence, artificial life, robots engineering, and other various computer science fields. Certainly, our distributed formation control system is a typical multiagent system.

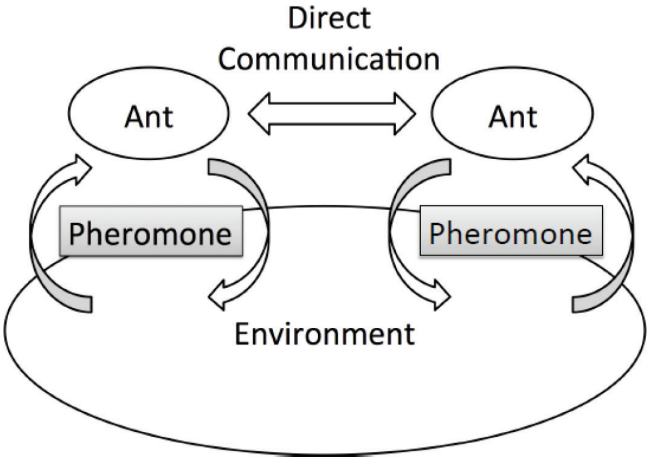

Figure 3. Ant as multiagent.

5. Pheromone-based formation

In this section, we present a decentralized control algorithm to compose any formations of mobile swarm robots. In this approach, mobile agents imitate pheromone communication of ants. The swarm robots are expected to compose any formations. The robots are connected by communication networks and not necessary intelligent; they initially do not have any control program to compose formations. Control programs that implement our algorithm are introduced later from outside as mobile software agents, which are able to migrate from one robot to another robot connected by the network. Our controlling algorithm is based on the indirect pheromone communication of social insects, that is, ants. We have implemented the ant and the pheromone as mobile software agents. Each ant agent has partial information about the formation and drives a robot based on the information to compose the formation. The partial information consists of relative locations of neighbor robots to cooperatively compose the target formation. Once the ant agent detects an idle robot, it occupies that robot and generates the pheromone agent to attract the specific ant agents to the location with neighbor robots. The pheromone agent repeatedly migrates to other robots to diffuse attracting information. Once the pheromone agent reaches the robot with an ant agent, the ant agent migrates to the robot closest to the location pointed by the pheromone agent, and then, drives the robot to the location. We have implemented a simulator based on our algorithm and conducted experiments to demonstrate the feasibility of our approach.

5.1. System overview

Our system consists of robots and two kinds of mobile software agents. We assume that the robots have simple capabilities of movement such as rolling wheels, measuring distance, and angle through an optical camera and other sensors. In addition to the mobile capability, the robots have some communication devices with which they can communicate each other through a communication network such as a wireless LAN.

All the controls for the mobile robots are achieved through the mobile agents. They are ant agents (AAs) and pheromone agents (PAs). In our algorithm, AAs attract each other to suitable positions as in **Figure 4** without any central coordinate that directs the other robots to compose a formation. In this figure, robots are represented as circles, and the neighbor locations of each robot are represented as stars to which vectors relative to the robot point, where vectors are represented as black arrows. The neighbor locations are supposed to be occupied by other robots to compose the formation. Colored arrows represent attracting force by pheromone. We assume that the target formations are represented as a set of positions of robots. For example, a line formation F_{line} whose length is 30 is represented by four robots such as F_{line} = {(0, 0), (0, 10), (0, 20), (0, 30)}.

Each AA takes charge of one position in a formation and drives a robot to the position. We assume that all the AAs are sent from the outside user machine to one of the reachable robots, and each AA does not know its own absolute position but has partial information of the formation. For example, when the target formation is a circle and AA *A*'s neighbors are B and *C* as shown **Figure 5**, *A* has only information of *B* and *C* represented as colored arrows in the figure ($\boldsymbol{P}_{AB}$ and $\boldsymbol{P}_{AC}$).

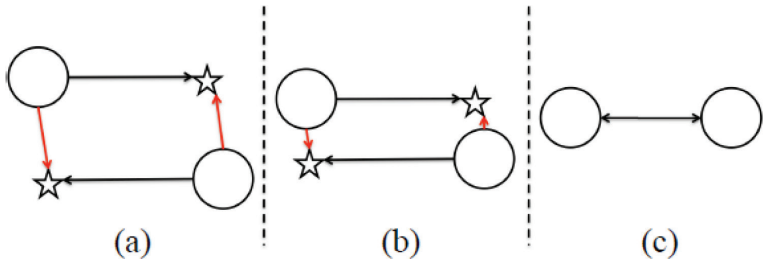

Figure 4. The process of composing a formation.

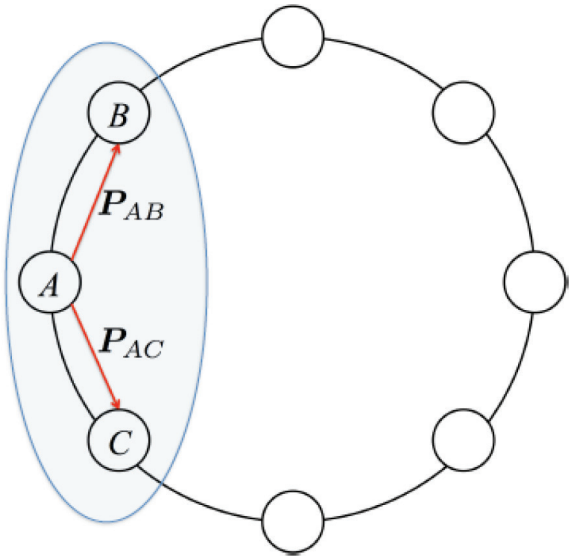

Figure 5. Local shape information of the ant agent *A*.

For simplicity, we assume that each robot can obtain a common direction by using devices such as compass. Each robot can measure an angle to another robot based on that common direction. We assume that when an agent that was on robot *A* migrates to robot *B*, robot *B* has a different coordinate (direction) from *A*'s as shown in **Figure 6**. As shown in the figure, θ_A is the angle to robot *B* from robot *A* and θ_B is to *A* from *B*, and dotted arrows represent robot's angles. When the agent on *A* migrates to *B*, the agent translates all the vector data it has such as the target location vector to adjust to *B*'s coordinate. To do this, we calculate the difference θ_r of angles between the two robots as follows:

$$\theta_r = 180 - (\theta_B - \theta_A) \tag{4}$$

In other words, the agent rotates its data vectors that are based on *A*'s coordinate by the difference angle θ_r.

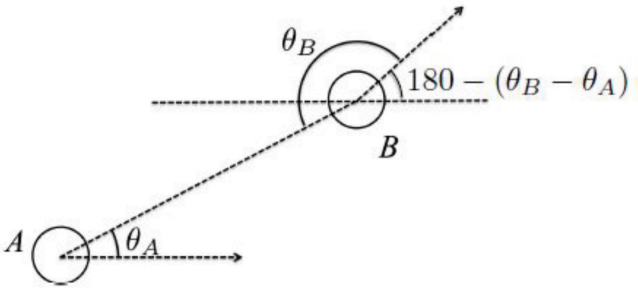

Figure 6. Rotation angle.

5.2. Ant agent

In this section, we describe the algorithm for AAs (Algorithm 1). AAs can migrate to robots through communication network. First, AAs traverse robots scattered in the field one by one to search idle robots. Once an AA finds an idle robot, the AA occupies that robot and generates PAs. The PA has vector values that point to the positions of neighbor AAs to compose the formation. The positions of neighbor AAs mean the positions of robots to compose the given formation. Neighbor AAs are supposed to drive robots to those positions. As long as an AA resides on an idle robot, it generates PAs at some intervals, which migrate to other robots to diffuse the information for attraction. Each PA has data $\boldsymbol{V}_t$ from the attractee to the neighbor location of its attractor. Once AA corresponding to the attractee receives the PA, the AA migrates to the robot closest to the objective location guided by the PA and then drives that robot to the location. If an AA simultaneously receives n PAs with vector $\boldsymbol{V}_i$, vector $\boldsymbol{V}_t$ held by the AA is computed as follows.

$$V_t = \frac{1}{n}\sum_{i=1}^{n} V_i \tag{5}$$

Algorithm 1 Ant Agents' behavior

```
if Current robot is idle then
    Generate PAs
    if The AA received PAs from other AAs then
        Synthesize vectors from PAs and directions
        Move to the synthesized location or migrate to a neighbor robot
    end if
else
    Migrate to another robot through network
end if
```

For example, the objective formation is a line with three robots, which are currently located as shown in **Figure 7**. In this figure, the stars represent the positions for AAs to occupy and are eventually the positions occupied by the robots. AA A is supposed to drive one robot to its corresponding position, and colored arrows represent vectors to the positions. A synthesizes

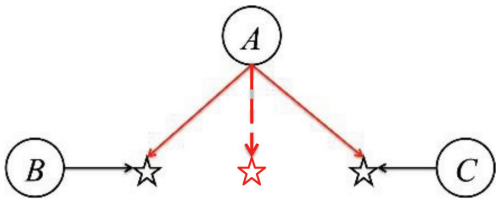

Figure 7. An example of vector synthesis.

the two vectors, resulting in new vector represented by the dashed arrow. The colored star is the position to where AA A must physically drive the robot. Once the AA finds some other robots within the view range, the AA judges whether to migrate to one of them by comparing $\boldsymbol{V}_t$ with the vector $\boldsymbol{V}_{another-to-dst}$ from the another robot to the destination, which is computed by using vector $\boldsymbol{V}_{another}$ to the another robot as follows:

$$\boldsymbol{V}_{another-to-dst} = \boldsymbol{V}_t - \boldsymbol{V}_{another} \tag{6}$$

If another robot is idle and $|\boldsymbol{V}_t|/2 > |\boldsymbol{V}_{another-to-dst}|$, the AA migrates to it instead of driving the current robot along $\boldsymbol{V}_t$. Notice that all the AAs are attracted each other along the guidance of PAs, so that they may be attracted with the total strength of them. This means that if an AA migrates to another robot when $|\boldsymbol{V}_t| > |\boldsymbol{V}_{another-to-dst}|$, the robot could overrun, so that it may waste extra time for convergence. The conditional migration of AA enables it to move to a robot closer to the destination without physically driving its current robot, which decreases not only convergence time but also the total energy consumption. Thus, the vector values play important roles in our system. In an absolute coordinate system, it is easy to compare or synthesize them. Each AA has, however, only an individual relative coordinate system. Therefore, PAs store angle values of coordinate system of the attractor AA that generated them and the attractee AA receiving the PAs adjusts its coordinate system by using angle values of them. Let θ be the angle value that the AA has, n be the number of the PAs that the AA receives, and θ_i be an angle value that ith PA has. Once an AA receives a series of PAs with those angle values, the AA updates its angle value as follows:

$$\theta = \frac{1}{n+1}\left(\theta + \sum_{i=1}^{n} \theta_i\right) \tag{7}$$

In this manner, all the individual coordinate systems are averaged and AA's driving direction converges to a certain angle. Therefore, the final location and the direction of the formation depend on the initial locations of the robots and the initial angles of the AAs.

5.3. Pheromone agent

In this section, we describe the algorithm for PAs (Algorithm 2). Each PA is responsible for guiding the specific AA to the location to occupy in the formation. A PA generated by an attractor AA seeks its attractee AA by traversing robots. Once the PA finds the attractee AA, the PA repeats the following behaviors (**Figure 8**):

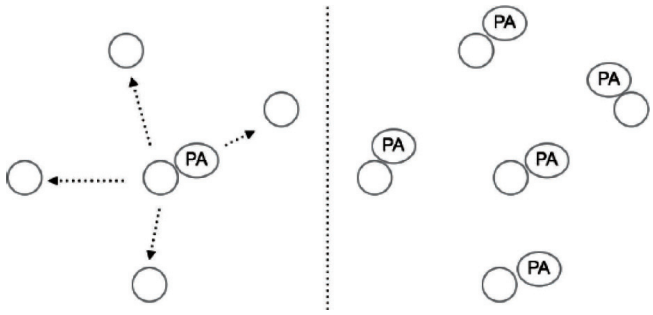

Figure 8. PAs clone themselves and migrate to neighbor robots.

1. PAs clone themselves on the current robots.
2. PAs migrate to all the other robots within the view range of the current robots.

Algorithm 2 Pheromone Agents' behavior

```
if Current robot has an AA then
  if The AA isn't one of attractees of this PA then
    Clone itself and migrate to neighbor robots
  else
    Stop migration
  end if
else
  Clone itself and migrate to neighbor robots
end if
if Certain time has passed from when generating then
  Defuse itself
end if
```

Since PAs are repeatedly generated on the same attractor AA, some of them may reach the same attractee AA through different routes at the same time. In this case, the attractee AA picks up the last PA and discards the other ones, in order to follow the latest information. Also, each PA has "time to live" and fades away after a certain time has elapsed as real pheromones evaporate, so that the information diffused by PAs is kept fresh. The information of PAs includes identifiers of the attractor and attractee AAs and a vector value pointing to the neighbor location of the attractor. The attractor AA's id is used to check if the current robot is already engaged in a destination to reach. The vector value has to be updated so that it keeps on pointing to the same location, whenever the PA migrates to another robot. The current vector $\boldsymbol{V}_t$ is updated using vector $\boldsymbol{V}_n$ to the destination robot of the migration as follows:

$$\boldsymbol{V}_t \leftarrow \boldsymbol{V}_t - \boldsymbol{V}_n \tag{8}$$

Figure 9 shows an example process where a PA updates V_t along with the migration path (*A*, *B*, *C*, *D*). Initially, the PA is on robot *A* and $\boldsymbol{V}_t$ points to the location to occupy

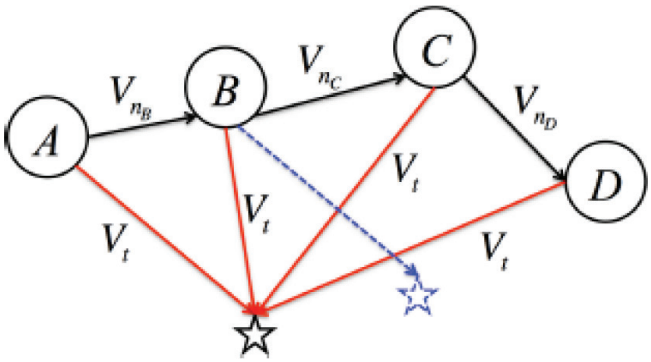

Figure 9. PAs update a target vector when PAs migrate.

represented by the solid line star. Once it migrates to robot *B*, V_t would point to the location represented by the dotted line star. In order to make V_t keep pointing to the black star, the PA has to adjust V_t through the calculation: $V_t \leftarrow V_t - V_{nB}$, where V_{nB} is the vector to robot *B* from robot *A*. In the same manner, the PA repeats adjustments of V_t on robot *B*, *C*, and *D*, so that V_t still points to the black star on robot *D*.

5.4. Experimental results

In order to demonstrate the correctness and effectiveness of our method, we have implemented a simulator and conducted numerical experiments on it. In the experiments, we assume the following conditions.

- Robots are randomly placed in an 800 × 800 square field in the simulator.
- The reachable range of Wi-Fi network for each robot is 150 units.
- Each robot can move 2 units in each step in the simulator.
- Interval time of generating PAs is 1 step.
- Duration time of PAs is 10 steps.
- The initial locations and angles of robots are randomly determined without overlapping.
- Each robot is represented as a circle on the grid field.
- Robots that have an AA are emphasized by large circle.

We confirmed that robots converge to the target formation as shown in **Figure 10** where the location and angle of the formation is determined by an initial condition. **Figure 11** shows the result of the formation that represents the letter A in different initial conditions from **Figure 10**. As shown in the figures, the initial conditions determine the location and the angle of the resulted formation.

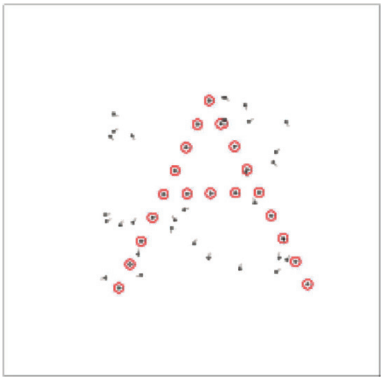

Figure 10. Formation of letter A.

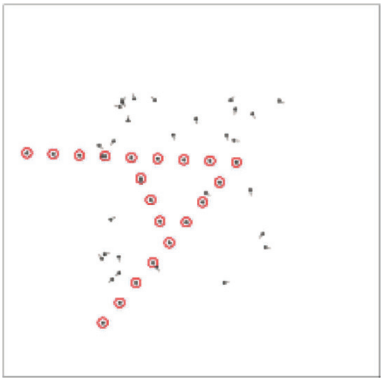

Figure 11. Formation of letter A (2).

Migrations to other robots contribute to reducing the time for convergence and the total length of movements of robots. We have measured the convergence time for the different numbers of robots between 20 and 50 with the interval 10. The objective shape is a circle whose radius is 150 units, and it consists of 20 robots.

When the number of robots is 20, all the robots must participate in the formation, and there is not any idle robot. On the other hand, when we increased the number of robots to 50, AAs can use extra 30 idle robots for migration. In order to measure quantitatively the degree of correspondence with the objective ideal formation, we calculated the average distance D as defined in Eq. (9). We calculated the barycenter for both actual and ideal points, calculating all relative coordinate of robots from the barycenter. Then, we average all the distances between actual and ideal relative coordinates $\boldsymbol{A}_i$ and $\boldsymbol{I}_i$, calculating the average distance D by the following equation:

$$\mathrm{D} = \frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{I}_i - \boldsymbol{A}_i\right\| \tag{9}$$

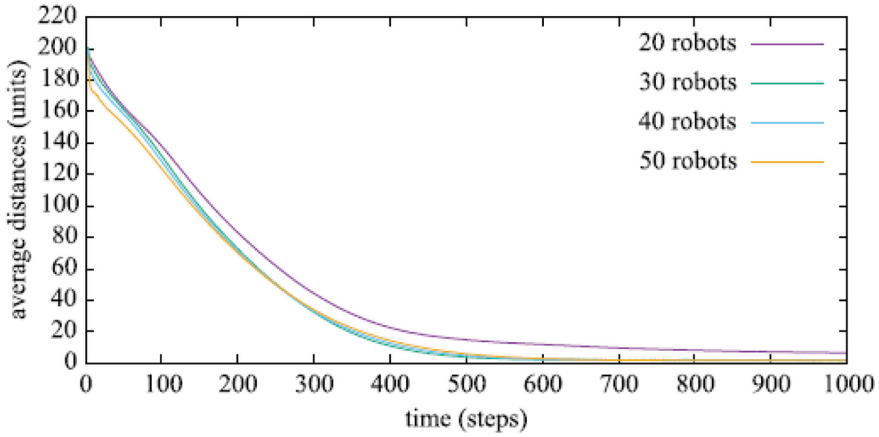

Figure 12. Time for convergence for the different number of robots.

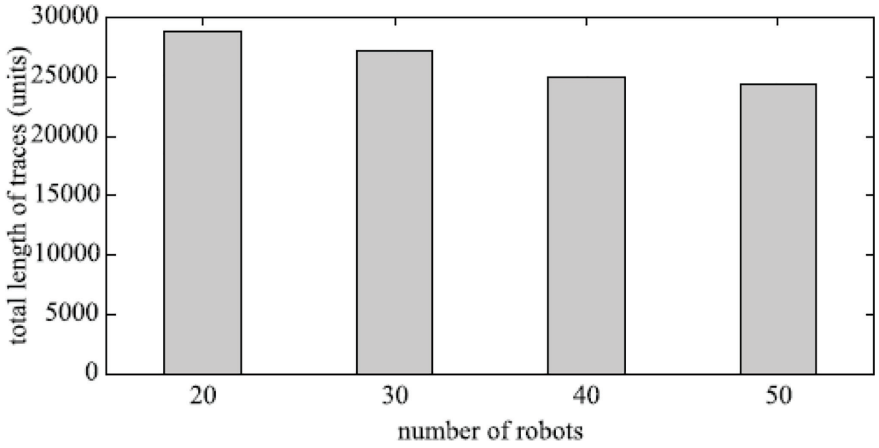

Figure 13. Total length of movements for different number of robots.

$\boldsymbol{I}_i$ is the ideal relative coordinate of the ith robot from the barycenter and $\boldsymbol{A}_i$ is the actual relative coordinate of ith robot from the barycenter. In the equation, n is the number of AAs that compose a given shape. **Figures 12** and **13** show the processes of convergence and the total length of movements until the time steps 1000. The horizontal axes are the duration time and the numbers of robots, respectively, and the vertical axes are the average distance from actual AAs' current locations to the ideal locations in the formation and total length of movements of all the robots with AAs, respectively. As shown in **Figure 12**, the migration can reduce the duration time for convergence, and if the number of idle robots is more than enough, which is over 10, the measures of duration time for convergence are almost identical. Also, as shown in **Figure 13**, if the number of robots is more than enough, the extra idle robots can contribute to reducing the total length of movements of robots. To show the effectiveness of migration from

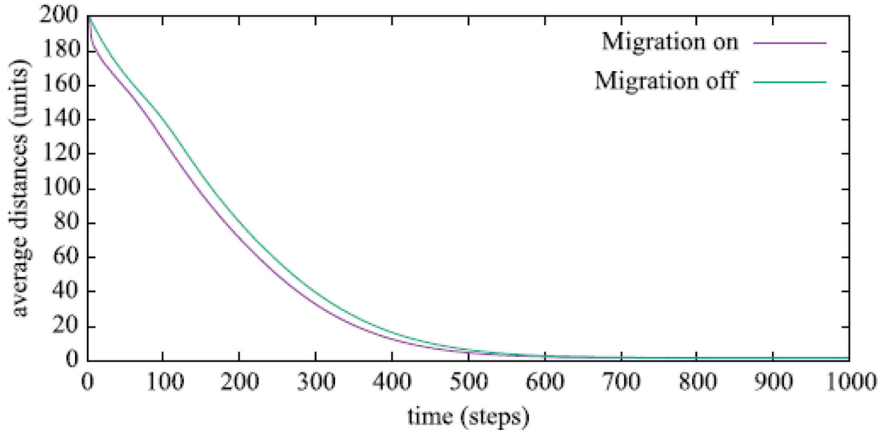

Figure 14. Time for convergence compared to migration off.

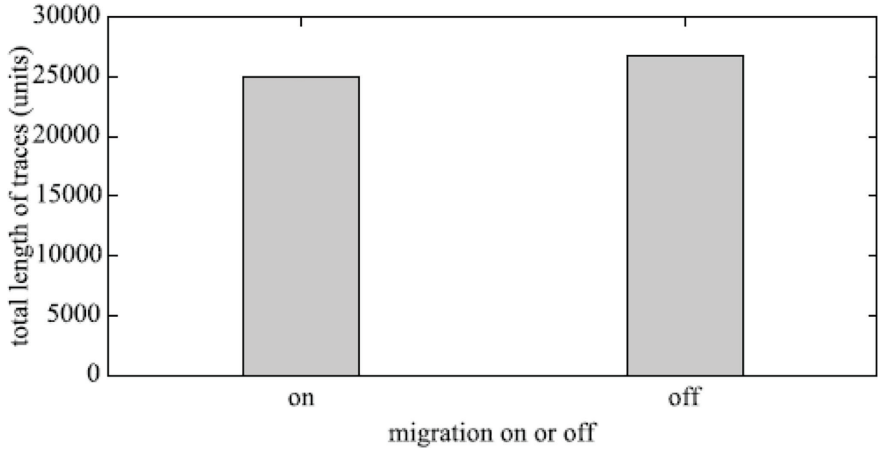

Figure 15. Total length of movements compared to migration off.

another aspect, we fixed the number of robots to 40. Under such a condition, we measured the duration time for convergence and length of movements and compared with those with no migration. As shown in **Figures 14** and **15**, the time for convergence and total length of movements were reduced by the migration.

We have examined the relation between the duration time for convergence and the number of AAs composing the formation, where we fixed the target formation to a circle with the radius of 150 and set the number of robots to 40, and under this condition, we changed the number of AAs composing the formation to 10, 20, and 30. The reason why we set the number of robots to 40 is that 10 extra idle robots are just enough to take advantage of migration as shown in **Figure 12**. **Figure 16** shows that the duration time for convergence increases as the

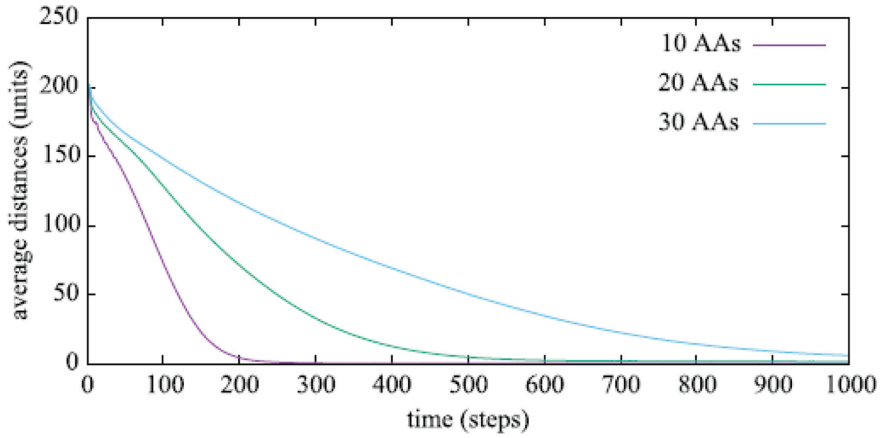

Figure 16. Time for convergence for different number of AAs.

number of AAs increases. In our algorithm, the location of formation is determined by the initial locations of robots. Once one AA moves to compose a formation, the whole formation is affected. The change of the location of an AA is observed by neighbor AAs and then the neighboring AAs adjust their locations. The changes of the locations of the neighbor AAs cause further adjustments of others. In this way, the change of the location of an AA is repeatedly propagated to the other AAs. At this time, for mutual-dependence relation, reverse directional propagation is also caused. Eventually, the whole formation settles down to a certain location after oscillation for some time. Thus, we can expect that as the target formations become larger, the number of time steps for propagating the changes of locations increases, so that the duration time for convergence also increases.

6. Conclusions and future works

In this chapter, we presented formation control methods based on mobile software agents that have the following features: (1) it does not have central computation; (2) it can efficiently use robot resources; (3) it can reconcile with sensor errors or movement errors; (4) it makes swarm robots compose arbitrary formations with various density inside the formations; and (5) it let swarm robots continue the task when some of them break down except for computers and communication devices because mobile agent can migrate other unbroken robots.

Our approaches make accurately arrange mobile robots to the proper places in a given formation. Upon composing the formation, each robot is not reserved for a specific position in the formation. Instead, each mobile agent is reserved for the position. This means that the mobile agent-based control manner simulates a system where each robot has a specific role in distributed environments, although any robot does not actually have its own role. The property of our approaches enables users to easily implement sophisticated formation without sacrificing scalability and robustness as a distributed system.

In order to implement our approach, we have used decentralized algorithms for the formation control for swarm robots using mobile agents. In the algorithm, we introduced two kinds of mobile software agents; that is, ant agents and pheromone agents. Ant agents generate pheromone agents that have local information about the formation to guide other ant agents to the appropriate locations to form the target formation. Once pheromone agents are generated, pheromone agents clone themselves and migrate to other robots to find target ant agents. When ant agents receive pheromone agents, which have information to guide the ant agents, the ant agents move to the locations that the pheromone agents point to. Eventually the whole robots compose the target formation. In order to show the feasibility of our algorithm, we have implemented a simulator on which we have conducted numerical experiments. We have shown that the algorithm can achieve not only the distributed formation control but also suppressing energy consumption through the experiments.

Our approach can continue tasks when robots get in trouble and cannot move, because even from the troubled robots, ant agents can migrate to other trouble-free robots. Once ant agents are lost, however, our current approaches cannot complete the formation. As a future work, we will give a solution to address this problem. For example, since the pheromone-based approach can compose a formation with some lacks of robots, ant agents corresponding to the lost positions could also be recognized by their neighbor ant agents through observing the pheromone agents to attract the lost ones. At this time, the neighbor agents could create new ant agents for the lost positions.

Acknowledgements

This work is partially supported by Japan Society for Promotion of Science (JSPS), with the basic research program (C) (No. 26350456, 26750076 and 17K01304), Grant-in-Aid for Scientific Research (KAKENHI), and Suzuki Foundation.

Author details

Yasushi Kambayashi[1]*, Ryotaro Oikawa[2] and Munehiro Takimoto[2]

*Address all correspondence to: yasushi@nit.ac.jp

1 Nippon Institute of Technology, Miyashiro, Japan

2 Tokyo University of Science, Noda, Japan

References

[1] Nagata T, Takimoto M, Kambayashi Y. Cooperatively searching objects based on mobile agents. In: Transaction on Computational Collective Intelligence XI, Vol. 8065 of Lecture Notes in Computer Science. 2013. pp. 119-136

[2] Abe T, Takimoto M, Kambayashi Y. Searching targets using mobile agents in a large scale multi-robot environment. In: Proceedings of the First KES International Symposium

on Agent and Multi-Agent Systems (KES-AMSTA 2011), Vol. 6682 of Lecture Notes in Artificial Intelligence. 2011. pp. 211-220

[3] Shibuya R, Takimoto M, Kambayashi Y. Suppressing energy consumption of transportation robots using mobile agents. In: Proceedings of the 5th International Conference on Agents and Artificial Intelligence (ICAART 2013). SciTePress; 2013. pp. 219-224

[4] Mizutani M, Takimoto M, Kambayashi Y. Ant colony clustering using mobile agents as ants and pheromone. In: Proceedings of the Second Asian Conference on Intelligent Information and Database Systems Applications of Intelligent Systems (ACIDS 2010), Lecture Notes in Computer Science 5990. Springer-Verlag; 2010. pp. 435-444

[5] Shintani M, Lee S, Takimoto M, Kambayashi Y. Synthesizing pheromone agents for serialization in the distributed ant colony clustering. In: ECTA and FCTA 2011—Proceedings of the International Conference on Evolutionary Computation Theory and Applications and the Proceedings of the International Conference on Fuzzy Computation Theory and Applications [Parts of the International Joint Conference on Computational Intelligence (IJCCI 2011)]. SciTePress; 2011. pp. 220-226

[6] Shintani M, Lee S, Takimoto M, Kambayashi Y. A serialization algorithm for mobile robots using mobile agents with distributed ant colony clustering. In: Knowledge-Based and Intelligent Information and Engineering Systems, Vol. 6881 of Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2011. pp. 260-270

[7] Cheng J, Cheng W, Nagpal R. Robust and self-repairing formation control for swarms of mobile agents. In: AAAI'05 Proceedings of the 20th National Conference on Artificial Intelligence. Vol. 1. AAAI Press; 2005. pp. 59-64

[8] Oikawa R, Takimoto M, Kambayashi Y. Distributed formation control for swarm robots using mobile agents. In: Proceedings of the Tenth Jubilee IEEE International Symposium on Applied Computational Intelligence and Informatics. 2015. pp. 111-116

[9] Oikawa R, Takimoto M, Kambayashi Y. Predictive distributed formation control for swarm robots using Mobile agents. Transactions on Automatic Control and Computer Science. 2016;**61**(1):83-90

[10] Oikawa R, Takimoto M, Kambayashi Y. Composing Swarm Robot Formations Based on their Distributions Using Mobile Agents, Multi-Agent Systems and Agreement Technologies. EUMAS 2015, Vol. 9571 of Lecture Notes in Computer Science. Springer; 2016. pp. 108-120

[11] Yajima H, Oikawa R, Takimoto M, Kambayashi Y. Practical formation control of swarm robots using Mobile agents. In: Proceedings of the Intelligent Systems Conference. 2017. pp. 1002-1009

[12] Takimoto M, Mizuno M, Kurio M, Kambayashi Y. Saving energy consumption of multi-robots using higher-order mobile agents. In: Proceedings of the First KES International Symposium on Agent and Multi-Agent Systems (KES-AMSTA 2007), Vol. 4496 of Lecture Notes in Artificial Intelligence. Springer-Verlag; 2007. pp. 549-558

[13] Kambayashi Y, Takimoto M. Higher-order mobile agents for controlling intelligent robots. International Journal of Intelligent Information Technologies (IJIIT). 2005;**1**(2):28-42

[14] Kambayashi Y, Ugajin M, Sato O, Tsujimura Y, Yamachi H, Takimoto M, et al. Integrating ant colony clustering to a multi-robot system using mobile agents. Industrial Engineering and Management Systems. 2009;**8**(3):181-193

[15] Dorigo M, Birattari M, Stützle T. Ant colony optimization–artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine. 2006;**1**(4):28-39

[16] Dorigo M, Gambardella LM. Ant colony system: A cooperative learning approach to the traveling salesman. IEEE Transactions on Evolutionary Computation. 1996;**1**(1):53-66

[17] Deneubourg J, Goss S, Franks NR, Sendova-Franks AB, Detrain C, Chreien L. The dynamics of collective sorting: Robot-like ant and ant- like robot. In: Proceedings of the First Conference on Simulation of Adaptive Behavior: From Animals to Animats. MIT Press; 1991. pp. 356-363

[18] Wand T, Zhang H. Collective sorting with multi-robot. In: Proceedings of the First IEEE International Conference on Robotics and Biomimetics. 2004. pp. 716-720

[19] Lumer ED, Faiesta B. Diversity and adaptation in populations of clustering ants, from animals to animats 3. In: Proceedings of the 3rd International Conference on the Simulation of Adaptive Behavior. MIT Press; 1994. pp. 501-508

[20] Balch T, Arkin RC. Behavior-based formation control for multirobot teams. Robotics and Automation, IEEE Transactions on. 1998;**14**(6):926-939

[21] Anthony Lewis M, Tan K. High precision formation control of mobile robots using virtual structures. Autonomous Robots. 1997;**4**(4):387-403

[22] Das AK, Fierro R, Kumar V, Ostrowski JP, Spletzer J, Taylor CJ. A vision-based formation control framework. Robotics and Automation, IEEE Transactions on. Oct 2002;**18**(5): 813-825

[23] Unsal C, Bay JS. Spatial self-organization in large populations of mobile robots. In: Proceedings of 1994 9th IEEE International Symposium on Intelligent Control. 1994. pp. 249-254

[24] Mamei M, Vasirani M, Zambonelli F. Experiments of morphogenesis in swarms of simple mobile robots. Applied Artificial Intelligence. 2004;**18**(9-10):903-919

[25] Kondacs A. Biologically-inspired self-assembly of two-dimensional shapes using global-to-local compilation. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence; August 9-15, 2003; Acapulco, Mexico. 2003. pp. 633-638

[26] Sty K, Nagpal R. Self-reconfiguration using directed growth. In: Proceedings 7th International Symposium on Distributed Autonomous Robotic Systems. 2004. pp. 1-10

[27] Gordon N, Wagner IA, Bruckstein AM. Discrete bee dance algorithm for pattern formation on a grid. In: 2003 IEEE/WIC International Conference on Intelligent Agent Technology. 2003. pp. 545-549

[28] Rubenstein M, Shen WM. A scalable and distributed approach for self-assembly and self-healing of a differentiated shape. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2008. pp. 1397-1402

[29] Wooldridge M. An Introduction to Multiagent Systems. 2nd ed. Chichester, West Sussex, UK: John Wiley & Son; 2009

[30] Bellifemine F, Caire G, Greenwood D. Developing Multi-Agent Systems with Jade. Chichester, West Sussex, UK: John Wiley & Son; 2007

[31] AgentSpace—A Mobile Agent System. Available from: http://research.nii.ac.jp/~ichiro/

[32] Dorigo M, Gambardella LM. Ant algorithms for discrete optimization. Artificial Life. 1999;**5**(2):137-172

[33] Dorigo M, Stützle T. Ant Colony Optimization. Cambridge, MA, USA: MIT Press; 2004