Distributed Swarm Formation
Using Mobile Agents

(モバイルエージェントを用い
群ロボット分散形成)

平成 29 年 3 月

及川 亮太郎
## Contents

1 Introduction ........................................... 6  
   1.1 Motivation ........................................ 6  
   1.2 Background and Related Work ....................... 9  
   1.3 Thesis Organization ................................ 11  

2 Preliminary ........................................ 13  
   2.1 Mobile Agent ...................................... 13  
      2.1.1 Agent and Environment .......................... 13  
      2.1.2 Definition of Mobile Agents .................... 15  
      2.1.3 Features of Mobile Agents ....................... 15  
      2.1.4 State of Mobile Agents .......................... 17  
      2.1.5 Environment of Mobile Agent ...................... 18  
   2.2 ACO inspired behavior ............................. 19  
      2.2.1 Basic Concept of Pheromone ..................... 19  
      2.2.2 Pheromone Communication ......................... 19  
      2.2.3 Food Collection of Ants ......................... 20  
      2.2.4 Ants as Multi-Agent System ....................... 20  
   2.3 Ant Colony Optimization ............................ 22  
   2.4 Ant Colony Clustering .............................. 23  

3 Leader Based Formation ............................. 25  
   3.1 System Overview .................................. 25  
   3.2 Guide Agent ........................................ 26  
   3.3 Node Agent ......................................... 28  
   3.4 Experimental Results .............................. 28  
      3.4.1 Migration ...................................... 29  
   3.5 Summary ............................................ 32  

4 Pheromone Based Formation ......................... 33  
   4.1 Pheromone Based Clustering ......................... 33  
      4.1.1 The Ant Colony Clustering ....................... 34
## List of Figures

2.1 The environment and agents. ................................. 13
2.2 Communication methods of server-client applications. ....... 16
2.3 Mobile agent model. ........................................... 16
2.4 AgentSpace. .................................................. 18
2.5 Ant proceed at a branch. ...................................... 21
2.6 Ant as multi-agent. ............................................ 21

3.1 Formation. .................................................... 27
3.2 Initial state. .................................................. 27
3.3 Barycenter. .................................................... 27
3.4 Translation. ................................................... 28
3.5 NA Location. .................................................. 28
3.6 The movement. ................................................ 28
3.7 Formation of letter A. ........................................ 29
3.8 Formation of letter A (2). .................................... 29
3.9 Effect of migration. ............................................ 31
3.10 Total length of traces for different condition. ................. 31
3.11 Effect of the number of robots. ............................... 32

4.1 The relation between agents and robots. ....................... 35
4.2 Traversals of mobile agents. ................................ 35
4.3 Pheromone walking. .......................................... 36
4.4 The behavior of picking up an object and randomly walking. 36
4.5 Synthesis of pheromones. ..................................... 37
4.6 Migration of pheromones. ..................................... 38
4.7 Synthesizing pheromone agents. ............................... 38
4.8 Simulated robots. ............................................. 39
4.9 The number of clusters and the time for convergence. ........ 40
4.10 The total length of traces. ................................... 40
4.11 The process of forming a shape. ............................. 42
4.12 Local shape information of ant agent A. ..................... 43
4.13 Rotation angle. .............................................. 43
4.14 An example of vector synthesis.  
4.15 PAs clone themselves and migrate to neighbor robot.  
4.16 PAs update a target vector when PAs migrate.  
4.17 Formation of letter A.  
4.18 Formation of letter A (2).  
4.19 Time for convergence for the different number of robots.  
4.20 Total length of movements for different number of robots.  
4.21 Time for convergence compared to migration off.  
4.22 Total length of movements compared to migration off.  
4.23 Time for convergence for different number of AAs.  
4.24 Comparison to the leader based approach.  
4.25 Influence of error.  
4.26 Formation of letter G.  
4.27 Time for convergence for circle and Letter G.  
4.28 Time for convergence for different number of neighbor AAs.  

5.1 Predicting the locations of robots.  
5.2 Formation of letter A.  
5.3 Formation of letter G.  
5.4 Formation of line.  
5.5 Formation of lattice.  
5.6 Comparison in letter A.  
5.7 Comparison in letter G.  
5.8 Comparison in line formation.  
5.9 Comparison in lattice formation.  
5.10 Comparison of letter A with interval 4.  
5.11 Comparison of letter A with interval 8.  
5.12 Comparison of letter A with interval 16.
List of Tables

5.1 Comparison of mileages. . . . . . . . . . . . . . . . . . . . . 60
5.2 Comparison of letter A with various interval. . . . . . . . . . . 61
Chapter 1

Introduction

1.1 Motivation

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 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. However, that control manner requires switching several programs on each robot depending on the changes of the environment shared by all the robots, which makes the implementation of the system difficult. Introducing mobile agents into multi-root systems solves this problem.

Mobile agent system achieves both of the ease of implementation and 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 are killed 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 multi-robot 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 paper, 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 situations. 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 multi-robot system. For example, robots may aggregate for coordinated search i.e. 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. However, practically, it is rare for each robot to be able to know the entire information. In this paper, we propose distributed formation algorithms using mobile agents.

We assume that in a field, there are several robots that may be used for not only the specific task such as the formation, but also may be shared by various different tasks because any mission is introduced into robots by mobile agents through WiFi 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 the coordinates of it and other robots in some way.

In this paper, we propose a method that can effectively use robots resources including idle ones, work with devices that may have sensor or movement errors, construct arbitrary shapes that can have various density inside and continue the task when some robots break down except for computers and communication devices. Furthermore, we also propose approaches that can form a shape without any distortion even if robots are broken completely including the computers and communication devices, although the shape may lack some parts.

First, we propose a leader based approach using mobile agent. To form shapes, we introduce a leader agent that manage a whole information of robots and gives locations to move as relative vector information to the other agents that drive robots. The algorithm calculates the locations suitable for efficiently composing a formation, so that the robots can suppress the duration time and energy consumption. In this algorithm, we introduce two kinds of mobile agents: the guide agent and node agents. The guide agent migrates among the robots in order to collect initial locations of the robots and calculates the optimal locations for the robots to move for the formation based on the conceptual barycenter of them. The node agents drive the robots to the locations that the guide agent calculates.

Seconds, as a basis of formation, we propose a clustering method that is inspired by pheromone communication of ants that are known as one of social insects. The leader based approach has an issue that if the leader agent is lost the formation, the task cannot be completed. We propose a pheromone based ap-
proach that have no leader or no server by extending our clustering method. In the approach, we take advantage of a formation algorithms inspired by ant colony clustering (ACC), which is one of the clustering methods that model the behaviors of social insects such as ants. ACC simulates behaviors of food collection of ants, where ants interact each other using pheromones without any leader. In the clustering based on ACC, artificial ants gradually form several clusters without forming any shape, imitating real ants. We can apply the approach for collecting carts, which are intelligent carts with a computer and wheels rotated by motors, in a floor at the airport.

When we pass through terminals of the airport, we often see carts scattered in the walkway and laborers manually collecting them one by one. It is a laborious task and not a fascinating job. It would be much easier if carts were roughly gathered in any way before the laborers begin to collect them.

In our distributed ACC approach, some Ant agents, which is mobile software agents corresponding to ants, interactively traverse robots (intelligent carts). Furthermore, Pheromone agents, which is also mobile software agents corresponding to the pheromones, are created on robots included in a cluster, and diffuse to surrounding robots through migrations to attract other Ant agents to the cluster. The distributed ACC approach contributes to the followings:

1. dynamically reflecting circumstances, and
2. saving energy consumption because of move of Ant agents through migrations.

We also show that the number of Ant agents can be decreased without sacrificing efficiency of clustering in our experimental results.

Third, we extend the distributed ACC for formation, which we call pheromone based formation approach. In the pheromone based approach, each robot does not need any leader or server as well as the distributed ACC, and 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 shape and attracts the neighbor Ant to the positions with their driving robots using Pheromone agent, which we have proposed for the distributed ACC. As mentioned, 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 bi-directional attraction results in settling all the Ant agents with their driving robots down to the objective positions in the target shape without either knowledge of the whole shape. Since an attracted Ant agent repeatedly migrate 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.

Finally, we extend the pheromone based approach to consider time lags of pheromone diffusion. The pheromone based approach works well, but it has some rooms for improving in terms of efficiency. In this approach, we focus on time lag that is caused by the migrations of pheromone agents. As mentioned above, a pheromone agent attracts the specific ant agent to the neighbor location of the ant agent that generates the pheromone agent. The time lag occurs in the process where the pheromone agent moves from attractor to attractee through migrations. In the time lag, the attractor may move to another location. In this case, if the pheromone agent guides the attractee to the neighbor location based on the original position of the attractor, it is misguidance and decreases the efficiency of formation. We propose an extension of the pheromone based approach that the attractee of each pheromone agent predicts the future neighbor location of its attractor. We call the extended approach predictive approach. In the predictive approach, the prediction is achieved based on optional information of movement vector of the attractor and the generation time of the pheromone agent, which are included in the pheromone agent in addition to the pheromone information in the pheromone based approach. The attracted ant agents predict and modify the destination locations using the optional information. We show that this approach improves the duration time for convergence and the total mileage of all the robots.

1.2 Background and Related Work

Kambayashi and Takimoto have proposed a framework for controlling intelligent multiple robots using higher-order mobile agents [8, 9, 10]. The framework helps users to construct intelligent robot control software by migration of mobile agents. Since the migrating agents are higher-order, i.e. 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 them to make base control software relatively simple, and to add functionalities one by one as they know the working environment. Thus they 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.

Kambayashi and Takimoto have implemented a team of cooperative search robots in order to show the effectiveness of their framework. At the same time they have demonstrated that their framework contributes to energy saving of multiple robots [8, 1]. They have achieved significant energy saving for search robot applications. As well as these approaches, our formation approaches 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 [11, 12]. 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) [12]. Deneubourg et al. have originally formulated the biology inspired behavioral algorithm that simulates the ant corps gathering and brood sorting behaviors [13]. Wang and Zhang proposed an ant inspired approach along this line of research that sorts objects with multiple robots [14]. Lumer and Faiesta have improved Deneubourg’s model and proposed a new simulation model that is called Ant Colony Clustering [15]. Their method could cluster similar objects into a few groups.

Mizutani, Takimoto and Kambayashi 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, Lee, Takimoto and Kambayashi 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 paper 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 [16]. 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 [17]. 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 [18].

Some methods have a limited class of shapes that they can form. Unsal and Bay [19] proposed a formation method that forms shapes of rings and circles. In this approach, robots become beacons and instruct other robots to remain the
position and keep a certain distance. Mamei et al. [20] proposed similar method that form a crude polygons and it use message hop count instead of a proximity sensor.

Kondacs [21] 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. They use global and local compilation to automatically generate an agent. Stoy and Nagpal [22] present a related approach to 3D self-assembly using self-reconfigurable modular robot, where individual modules must remain connected. Although this system generates a wide variety of shapes, 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. Gordon et al. [23] proposed a method that form arbitrary shape sharing common coordinate system. However, the method involves significant central computation. Although Rubenstein and Shen [24] proposed a method that can self-repair, it can form only polar shapes.

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 form shapes 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 shape cannot control, it 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 form a shape instead of robots. The robots are only methods where mobile agents move to the target locations on the shape. 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, any solid shapes and even shapes with holes inside.

1.3 Thesis Organization

In this paper, we introduce four controlling methods using mobile agents, which achieve formation including clustering. Before giving the details of these methods, we give preliminaries for them in Chapter 2. In Chapter 3, we propose a leader based formation control approach that considers the distributions of swarm robots. In Chapter 4, we propose a pheromone based approach, which is based on a distributed formation control, extending mobile agent based clustering. In Chapter 5, we propose a predictive extension of the approach introduced in Chap-
ter 4. This algorithm considers the time lag caused by migrations of pheromone agents. Finally, we conclude remarks.
Chapter 2

Preliminary

2.1 Mobile Agent

2.1.1 Agent and Environment

Object Management Group (OMG), which gives a standard of distributed object techniques such as Common Object Request Broker Architecture (CORBA), and is composed of several hardware vendors such as IBM and Sun, has decided a plan about 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 an 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 [25].

The subject means a human, a robot or a 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 environment. That is, agents are independent and intelligent subjects that interact with the environment utilizing their own intelligence and actions, as shown in Fig. 2.1.

![Figure 2.1: The environment and agents.](image-url)
Here, what is the environment? Basically, when agents have capability of observing the outside of them and making decisions based on the observation, the environment indicates the part other than the agents. In the rest of the section, we describe some important features of the environment.

**Accessible and Inaccessible**

When agents can completely and accurately observe the details of the environment, the environment is accessible from the agents. Otherwise, the information from the environment is restricted or has delays, and therefore, the environment is inaccessible from the agents.

**Deterministic and Non-deterministic**

In some environments, when agents take certain actions, which cause the same changes of the environments, the environments are deterministic. In general, when multiple agents takes actions, the environment is non-deterministic.

**Episodic and Non-episodic**

When the experiences of an agent can be divided into units that are called episode, the environment of the agent is episodic. Episode is a sequence of observations and actions, which is used in the same meaning as trial. When an environment is episodic, the environment is simpler than a non-episodic environment because each episode is independent of the others.

**Static and Dynamic**

When an environment changes without agent’s actions, the environment is called dynamic. Conversely, when an environment changes only through agent’s actions, the environment is called static. In a static environment, because the environment does not change even if agents take a long time to determine actions, the time can be ignored, so that a system becomes extremely simple. However, when there are several agents in a static environment, the static environment may behave such as a dynamic environment in appearance, because some agents may change the environment.

**Discrete and Continuous**

When agent’s observations and actions can be distinguished in finite time, the environment is called discrete; otherwise, the environment is called continuous.
2.1.2 Definition of Mobile Agents

Recently, object oriented programming is a popular programming paradigm, where data representation and operations for it are abstracted, and only interfaces for the operations are given. Software agents are similar to the objects, except that they have some missions or tasks. Generally, the software agents have some features, all of which are not always satisfied, as follows:

- **Autonomy**
  They can independently take actions.

- **Re-activity**
  They can take actions along changes of the environment.

- **Pro-activity**
  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 agent to asynchronously process tasks and to suppress communication costs [26, 27].

2.1.3 Features of Mobile Agents

Fig. 2.2 shows communications of applications on computer 1 and computer 2 in server-client model. Server-client model have an advantage that an entire system can be implemented simply because a lot of clients are controlled by a few
servers. First, we describe differences of communication methods between traditional server-client applications and mobile agents as a figure. On the other hand, server-client model also have some disadvantages as follows:

- It is not easy for applications to be updated because they are installed on client side.
- Once some troubles occur on servers, clients also work correctly.
- Applications are not robust for communication troubles.
- The communication cost in applications is high.

Fig. 2.3 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
machine or accomplish load balancing through migrating to machines with less load. In dis-connectable communication environments such as wireless access, 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, communications often become costly because communications between a server and a client have to be always connected. In mobile agent model, the connections are restricted to migrations of the agents among computers. That is, the connections 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.

**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.

### 2.1.4 State of Mobile Agents

An agent defined by MASIF has two kinds of states as follows:

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.

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.

2.1.5 Environment of Mobile Agent

Some mobile agent systems use a server system to implement their mobility. For examples, Telescript that was developed by General Magic corporation, Aglets that was developed by IBM Japan, and picoPlangent that was developed by TOSHIBA are representative examples of server based systems. We use another server based system, AgentSpace (Fig. 2.4) and MobileSpace that was developed by Ichiro Sato. MobileSpace, which is an extension of AgentSpace, enables mobile agents to be composed hierarchically. We use these systems to implement our robot control system based on mobile agents.

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 agents to be efficient and independent of the others [28].

AgentSpace and MobileSpace transfer their attribute states to the destination
site with class files, but not execution states. Some other systems also transfer execution states, which require costly process for each migration.

### 2.2 ACO inspired behavior

Our mobile agents have behavior inspired by Ant Colony Optimization. In this section, we describe the details of the ACO inspired behavior.

#### 2.2.1 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. There are some kinds of pheromones, for examples, some pheromones attract other individuals that detect that pheromone, or some pheromones notify other individuals around them of danger. At this time, the properties of evaporation and diffusion that pheromones have 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.

#### 2.2.2 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 but not for itself, and occasionally they devote their life to defend their swarm.

Although each individual behaviors seem to be randomly decided based on their judgment depending on various distributed and local factors, the whole swarm seems to suitably behave from the macro-perspective.

Many ants have restricted capability of eyes, and in most cases, depend on the stimulus from tactile organ. Thus, their communication is done not through visual information but through chemical signals sensed by the tactile organs. This 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 control a swarm that is organized by them, 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 semi-optimal behaviors.

### 2.2.3 Food Collection of Ants

Food collection is one of the most important tasks for worker ants to maintain their colony. The food collection process consists of exploring and transporting foods.

Generally, the roles of worker ants are also distinguished to exploring ants and transporting ants. Once exploring ants find a food, they get 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. However, in seeing them as a swarm, the swarm shows a kind of intelligent behavior that seems to explore the shortest path between the nest and food. The observed intelligent behaviors for a swarm is called swarm intelligence. For example, consider a branch path as shown in Fig. 2.5. 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 disappears while pheromone 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 phenomenons is positive feedback of route selection, which is the mechanism of ant’s route selection. This mechanism achieves the optimization of detecting the shortest path without individual intelligence.

### 2.2.4 Ants as Multi-Agent 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 get 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. Fig. 2.6 shows the system of ants as multi-agents.

The generality and flexibility of ant colony are often applied to the design of artificial systems. The systems are called multi-agent systems, which are actively studied in fields such as artificial intelligence, artificial life, robots engineering
and other various computer science fields.

2.3 Ant Colony Optimization

A pheromone communication such an ant’s route exploration mentioned in the previous section can be modeled as a communication of multi-agent system, for which various methods are studied. One of the most practical multi-agent systems is Ant Colony Optimization (ACO). For example, it has been applied to solving a traveling salesman problem or network routing problem [29]. ACO is a kind of meta heuristics that is used to get approximate solutions independently of specific computational conditions. The meta heuristics try to resolve problems by introducing the following two notions:

One of the notions is probabilistic action selection of an artificial ant agent. The random factor enables an agent to be well controlled 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.

The other notion is pheromone evaporation. This imitates the behavior of physical pheromones. Pheromones emitted by each agent are diffused over time. This prevent old information, which is not any more useful because of changing the exploration territory, remaining for a long time. Thus, the notion of pheromone evaporation keeps information fresh in ACO algorithm.

Although ACO algorithms have some variations depending on target problems, we show traveling salesman problem as an example, because this kind of ACO have a lot of common properties to other problems. Details of this ACO and its latest research is published at Ant Colony Optimization Home Page [30].
Multiple artificial ant agents travel around cities, where distances between cities is represented by pheromone values. In traveling all cities based on ACO, cyclic paths are created as many as agents, which put pheromones on the cycle paths through which they passed. Pheromones on the paths are evaporated over time.

We define that the number of cities is $n$, the number of artificial ant agent is $m$, pheromone strength between city $i$, $j$ is $\tau(i, j)$, the initial pheromone strength is $\tau_0$, the number of iterations is $t$, of which the maximal value is $t_{\text{max}}$. The algorithm is represented by the following steps:

1. Initialize all pheromones to $\tau_0$ and set $t$ to 0,
2. Set all the artificial agents to the cities until its traces become cyclic,
3. Each ant agent repeat the selection of cities while circuit is completed.
4. Update pheromone information $\tau(i, j)$ on each ant’s cyclic path and its length
5. Increment $t$, and repeat from 2 if $t < t_{\text{max}}$
6. Output the cyclic path with the strongest pheromones of all the cyclic paths.

This ACO is also applied to clustering or sorting. In next section, we explain ant colony clustering.

### 2.4 Ant Colony Clustering

Stock-breeding of aphids and breeding larva are one of the behaviors of ant in their nest. Taking a look at ranch and nursery in the nest, live-stocks or larva are spatially organized and located by sizes and so on. This ecology is evolved to optimize the operation of feeding and so on.

This kind of behaviors are represented as following equations. In this equation, a probability that agent lift and down a object are $P_{\text{pick}}$ and $P_{\text{drop}}$.

\begin{align}
P_{\text{pick}} &= (1 - \chi) \cdot \left( \frac{k_{\text{pick}}}{k_{\text{pick}} + f(i)} \right)^2 \quad (2.1) \\
P_{\text{drop}} &= \chi \cdot \left( \frac{k_{\text{drop}}}{k_{\text{drop}} + f(i)} \right)^2 \quad (2.2)
\end{align}
Equation $f(i)$ is a density of peripheral analogous of $i$. More strictly, $f(i)$ is defined as an equation that decreases as amount of peripheral analogous increases.

$$f(i) = \begin{cases} 
\frac{\sum_{j \in N(i)} d(i,j)}{|N(i)|} & (N(i) \neq \phi) \\
1 & (N(i) = \phi)
\end{cases}$$  \hspace{1cm} (2.3)

Equation $d(i, j)$ is a similarity between $i$ and $j$ that is a distance in feature space and $N(i)$ is a set of closure of $i$. Furthermore, $d(i, j)$ is normalized to fulfill $0 \leq d(i, j) \leq 1$. When two objects are completely overlapped, $d(i, j) = 0$ is satisfied.

Equation $k_{\text{pick}}$ and $k_{\text{drop}}$ are parameters that gives thresholds to take actions respectively. And $\chi$ is reactive coefficient that is determined by the number $n$ of objects in ant’s Moore neighborhood that are adjacent 8 cells of an ant as follows:

$$\chi = \frac{n^2}{n^2 + k_{\text{crowd}}^2}$$  \hspace{1cm} (2.4)

In this equation, $k_{\text{crowd}}$ is a threshold and when $n = k_{\text{crowd}}$ holds, $\chi$ becomes $1/2$. The more objects are exist around an ant, the more increase a probability for ant to drop off the object approaching $\chi$ to 1.

Equation 2.1 and 2.2 become a model that ants that are acting randomly in the field are transporting objects according to environmental variables. Namely, a probability that an ant picks or drops the object is determined by a density of peripheral analogous around the ant. Based on this, an ant repeat picking up object in low density and dropping off in high density. As a result, an ant classify objects to clusters that have analogous features. This is called Ant Colony Clustering (ACC).
Chapter 3
Leader Based Formation

This chapter presents a control algorithm for composing formations of swarm robots using a leader agent. The algorithm is based on our mobile software agent based control model where robots acquire any control program through agent migrations on demand. The agent migrations have effectiveness for decreasing physical movements of robots, contributing to suppressing the total costs of any multi-robot systems. Our control approach takes advantage of the mobile agent model for composing formations. In the approach, we introduced a leader of mobile agents to collect locations of all the robots. The leader directs the target locations to the other agents based on collected locations. The control manner improves the time for formation task and the total movement of robots. We have implemented our algorithm in a simulator and conducted experiments to demonstrate the feasibility of the approach.

3.1 System Overview

Our system model consists of robots and two kinds of mobile software agents. We assume that the robots have simple capabilities of locomotion such as driving wheels, measuring distance and angle through an optical camera and a sensor as well as some communication devices with which they can communicate each other through a communication network such as a wireless LAN.

In our algorithm, all the controls for the mobile robots are achieved through the mobile agents. They are a guide agent (GA) and node agents (NAs). There is one GA in the system, which calculates the location of the conceptual barycenter of the formation and all the locations for the robots to occupy, based on the conceptual barycenter. NAs physically drive the robots to the locations to compose the formation. Here, consider that several formations are continuously performed. In that case, the distribution of robots in the field would be different every time.
In order to efficiently adapt the formation to each distribution, while suppressing the duration time for composing a formation and the total length of traces of robot movements, the GA determines the target locations of the robots based on the center coordinate, i.e. the conceptual barycenter, of each distribution.

To determine the center coordinate of the formation, we use the concept of the barycenter of all robots in the field. The GA migrates among robots in order to collect locations of the robots and calculate the barycenter of them as if they are connected into one object. Upon the completion of GA’s calculation, NAs drive the robots to the locations that the GA determines based on the conceptual barycenter. We assume that the objective shape is represented as a set of point coordinates. We also assume that the mobile agents on the robots do not know the absolute coordinates of robots but they can measure the relative coordinate of neighbor robots using sensors or cameras. In our algorithm, all mobile agents uses relative coordinate from the base robot that is selected by the GA. Each relative coordinate is represented as a vector value. When an mobile agent migrates from the base robot $R_1$ to the robot $R_n$ along the path $(R_1, \cdots, R_n)$, the relative coordinate $p_n$ of robot $R_n$ are calculated as follows:

$$p_n = \sum_{i=1}^{n-1} v_i$$  \hspace{1cm} (3.1)

$v_i$ is the vector value from robot $R_i$ to robot $R_{i+1}$. The $v_i$ can be measured by sensors or cameras.

### 3.2 Guide Agent

In this section, we describe the algorithm for the GA. The GA can migrate to robots through communication networks. First, the GA visits robots scattered in the field to find an idle robot. Once the GA finds an idle robot, the GA appoints that idle robot to be the base robot to calculate the relative coordinates. Then, the GA traverses all the reachable idle robots one by one in order to collect their locations.

When the GA visits all the reachable idle robots, the GA calculates the conceptual barycenter $g$ as follows:

$$g = \frac{1}{n} \sum_{i=1}^{n} p_i$$  \hspace{1cm} (3.2)

$n$ is the number of robots and $p_i$ is the relative coordinate of each robot from the base robot, which is calculated by (3.1). The information of the original formation
$F$ is represented as follows:

$$F = \{f_1, f_2, \cdots, f_i, \cdots, f_n\}$$

We assume that $f_i$ is the vector value from the conceptual barycenter $f_o$ of the original arrangement of nodes in formation $F$. Henceforth we call the conceptual barycenter as just the barycenter. The GA translates them to the vectors from the base robot. The GA uses the barycenter $g$ in the field instead of barycenter of the original formation. Thus, the vectors from $g$ are identical to the vectors from $f_o$. To translate the vectors from $g$ to the vectors from the base robot, the GA adds $g$ to each $f_i$ as follows:

$$p_{fi} = f_i + g, f_i \in F$$

Since each $f_i$ is identical to the vector value from $g$, each $p_{fi}$ becomes the target locations for robots to move. Upon completion of the calculation of all the vector values, the GA generates NAs that drive robots to $p_{fi}$. For example, consider that the objective shape is the square as shown in Fig. 3.1. The vector $f_i$ in this figure is a relative coordinate from the barycenter of the original formation that is represented as a small circle at the center in the figure. Figure 3.2 shows the initial state of the positions of robots and the pale color circle in the figure represents the base robot that becomes the base coordinate of the positions of robots. That is, the base robot has coordinate $(0, 0)$. The GA calculates the barycenter $g$ by (3.2), which is the vector (or the relative coordinate) from the base robot (Fig. 3.3). After that, the GA overlaps the barycenter of the original formation and the barycenter $g$ in the field as shown in Fig. 3.4, where $f_i$ is the relative coordinate from the barycenter of the original formation. To translate $f_i$ to the vector from the base robot ($p_{fi}$ in Fig. 3.4), the GA add $g$ to $f_i$ followed by (3.3), so that the relative coordinate $p_{fi}$ is calculated.
3.3 Node Agent

In this section, we describe the algorithm for NAs. The GA calculates the relative coordinates \((p_{f_1}, \ldots, p_{f_n})\) from the base robot. \(n\) is the number of robots of the formation. For each robot, the GA generates the corresponding NA, and gives each NA the relative coordinate \(p_{f_i}\) as the target location to which it should drive the robot. The NA calculates the movement vector \(m\) to the target location as follows:

\[
m = p_{f_i} - p
\]  

(3.4)

Coordinate \(p\) is the current coordinate of the NA from the base robot as shown in Fig. 3.5, and movement vector \(m\) is a relative coordinate from current robot to target location as shown in Fig. 3.6. If a NA finds some robots that are nearer to the target location than the current robot, the NA migrates to the nearer robot instead of driving the current robot in order to suppress the duration time and energy consumption for composing the formation. After each migration, the coordinate \(p\) of NA is updated as follows:

\[
p \leftarrow p + v
\]

Vector \(v\) is the vector from the current robot to the nearest robot that is measured by sensors or cameras.

3.4 Experimental Results

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

1. Robots are scattered in an 800 x 800 square field in the simulator.
2. The view range of each robot is 150 units, and the range of wireless network is wider than the view range.

3. Each robot can move 2 units in each step in the simulator.

4. The initial locations of robots are randomly set without overlapping.

5. Each robot is represented as a circle on the grid field.

First, we conducted experiments for formation of letter A in different initial arrangements of robots as shown in Fig. 3.7 and 3.8. In the figures, where small black circles represent robots, large circles represent robots with a NA, and the small pale color circle represents the barycenter, both formations are composed around the center of the area where robots scatter. The results show that GA calculates the barycenter and the NAs compose the formation around it correctly.

3.4.1 Migration

We improved the efficiency by suppressing the duration time, and the total cost by the total length of physical traces of robots for composing formations. In the algorithm, when all the NAs arrive at the target locations, the formation task is completed.

In order to show the preciseness in our algorithm, we measured gaps between the current locations and the objective ideal positions, and calculated the average of them. First, we calculated the barycenter for both actual and ideal points, and then calculated all the coordinate of robots relative to the barycenter. After that, we measured the distances between the relative coordinates of actual robots and
the ideal relative coordinates, and calculated the averages $D$ of them for various shapes with the difference numbers of robots as follows:

$$D = \frac{1}{n} \sum_{i=1}^{n} ||I_i - A_i||$$  \hspace{2cm} (3.5)

In the equation, $I_i$ is the ideal relative coordinate of a robot from the barycenter, $A_i$ is the actual relative coordinate of the robot from the barycenter and $n$ is the number of NAs composing the shape.

Migrations to other robots can contribute to suppressing the duration time and energy consumption for composing formations. To show this effect, we conducted an experiment under the condition where NAs continue to stay on initial robots without migrating to other robots. We compare the result of an experiment in which NAs migrate to the robots closest to their target locations, where the objective shape is a circle whose radius is 150 units. The number of nodes of the shape is 20, and the number of robots is 60. Therefore, only 20 out of 60 robots are used to compose the shape. Figure 3.9 shows the average distance of the experiments, where the horizontal axis is time and the vertical axis is the average distance calculated by (3.5). In the case with migrations, the whole task was completed at 150 time steps, where the GA completed its task at 120 time steps and then NAs moved to their target in 30 time steps.

Fig. 3.10 shows total length of traces. Horizontal axis shows the number of robots (40, 60, 80 and 100) and cases that do not use GA and migration. On the other hand, in the case without migration, it took 310 time steps to complete the whole task, where the total time of the task of NAs was 190 time steps that was 6 times as much as the case with migration. Also, as shown by the bar named “60” in Fig. 3.10, the total length of traces of robots for composing the formation was 600 for the case with migration, and 2700 for the case without migration. This result shows the traces with migration are 4.5 times as short as ones without migration in average. These results show the superiority of the migration.

Finally, in order to check the influences of the number of robots to the duration time and energy consumption, we conducted experiments with the cases of the number of robots: 40, 60, 80 and 100. Like the other experiments, the objective shape is a circle whose radius is 150 units, and the number of nodes of the shape is 20. Figure 3.11 shows the average distances of these experiments. Each line graph in the figure show the number of robots (40, 60, 80 and 100). As shown the figure, the GAs completed their tasks at time steps 80, 120, 160 and 200, respectively. This means that the cost of GAs is in proportion to the number of robots. On the other hand, NAs complete their tasks at 60, 30, 25 and 20 time steps, respectively. Also, as shown in Fig. 3.10, the total lengths of traces are 800, 570, 470 and 410, which is in inverse proportion to the number of robots. The result shows that
Figure 3.9: Effect of migration.

Figure 3.10: Total length of traces for different condition.

the migration is suppressing energy consumption more effectively as the number of robots increases. In our simulator, mobile agents are able to migrate to other robots in one time step and robots move 2 units in one time step. Hence, the faster network speed is, the more important the number of robots becomes.

Conversely, when network speed is slow, visiting robots for collecting their locations might become a bottleneck of the task in terms of total time. The bar named “No GA” in Fig. 3.10 shows the experiment where the GA does not collect any locations of robots. In this case, where the number of robots is 60 and the barycenter is not calculated, the length of traces becomes twice as long as the case with GA. These experiments show that the newly proposed approach can suppress energy consumption in proportion to network speed and the number of robots without sacrificing the total time.
3.5 Summary

We have proposed a control algorithm for composing formations of swarm robots based on their distributions using mobile software agents. In this approach, we introduced two kinds of mobile software agents, a guide agent (GA) and node agents (NAs). The GA traverses all the robots and calculates the conceptual barycenter of them, and then, calculates the suitable locations of the formation based on the barycenter. The GA generates NAs that drive robots to the calculated locations. Each NA migrates to the robot closest to the target location, and drives the robot to the location.

In order to show the effectiveness of our algorithm, we have implemented it on a simulator. On the simulator, we have conducted numerical experiments. We have shown that our algorithm suppresses the duration time for a given formation task and energy consumption. Furthermore, we have shown that it is effective for suppressing energy consumption to increase the number of robots.
Chapter 4

Pheromone Based Formation

The leader based approach have a risk of loss of the leader agent that results in a loss of the task. In this chapter, first, we propose a clustering method and next we extend the clustering method to form any shapes that both imitate pheromone communication of ants to mitigate the risk of a loss of the task.

4.1 Pheromone Based Clustering

This section presents a new approach for controlling multiple robots connected by communication networks. The control mechanism is based on a specific Ant Colony Clustering (ACC) algorithm. In traditional ACC, an ant conveys an object, but in our approach, the ant is implemented as a mobile software agent that controls the robot which is corresponding to an object, so that the object moves to the direction ordered by the ant agent. At this time, the process in which an ant searches an object corresponds to a sequence of migrations of the ant agent, which is much more efficient than the search by a mobile robot. In our approach, not only the ant but also the pheromone is implemented as a mobile software agent. The mobile software agents can migrate to surrounding robots, so that they can diffuse over robots within their scopes. In addition, since they have their strengths as vector values, they can represent mutual intensification as synthesis of vectors. We have been developing the elemental techniques for controlling multiple robots using mobile software agents, and showed effectiveness of applying them to the previous ACC approach which requires a host computer that centrally controls mobile robots. The new ACC approach decentralizes the mobile robot system, and makes the system free from special devices for checking locations.
4.1.1 The Ant Colony Clustering

The coordination of an ant colony is composed by the indirect communication through pheromones. In traditional ACO system, artificial ants leave pheromone signals so that other artificial ant can trace the same path [11, 12]. Kambayashi et al. have developed an ACC system based on pheromone signals [10]. Randomly walking artificial ants have high probability to pick up an object with weak pheromone, and to put the object where it senses strong pheromone. They are not supposed to walk long distance so that the artificial ants tend to pick up a scattered object and produce many small clusters of objects. When a few clusters are generated, they tend to grow.

Since the purpose of the traditional ACC is clustering or grouping objects into several different classes based on some properties; it is desirable that the generated chunks of clusters grow into one big cluster so that each group has distinct characteristic. In our system, however, we want to produce several roughly clustered groups of the same type, and make each robot have minimum movement. (We assume we have one kind of cart robots, and we do not want robots move long distance.)

In the implementation of our ACC algorithm, when the artificial ants are generated, they have randomly supplied initial positions and walking directions. While an artificial ant performs random walk, when it finds an isolated object, it picks up the object, and continues random walk. While the artificial ant performs random walk, when it senses strong pheromone, it put the conveying object. The artificial ants repeat this simple procedure until the terminate condition is satisfied. These behaviors in the ACC are achieved using mobile software agents below.

4.1.2 System Overview

Our system model consists of robots and two kinds of mobile software agents as shown by Fig. 4.1. The robots have simple capabilities of movement such as driving wheels, detecting objects through a camera, and checking obstacles through supersonic sensors, and they can communicate each other through a communication network such as a wireless LAN. Any other noble capability like determining absolute locations and directions through GPS, RFIDs, or other devices are not required.

All the controls for the mobile robots are achieved through the mobile agents. They are: 1) Ant agents (AA), and 2) Pheromone agents (PA). Some mobile agents (AA) traverse robots scattered in the field one by one to search isolated robots as shown by Fig. 4.2(a). In this traversal, migrating to a robot imitates a behavior of an ant that finds an object, and picks it up. Upon arriving on a robot, AA controls the robot and drives it. AA is the agent that drives a robot. Without AA,
robots are just sitting. But AA has no knowledge of direction it should lead the robot so that it just give the robot a random walk. PA is the agent that guides AA to which direction it should drive the robot. PA is created on one of the robots that form a cluster. Randomly walking robots happen to bump each other and to create a cluster by chance. Robots that form a certain size of cluster are locked so that they become a nucleus of a growing cluster. A robot situated in a center location of such a growing cluster creates PA. In order to imitate disseminating pheromone, PA migrates to other robots as shown by Fig. 4.2(b). Once PA reaches the robot where AA exists, the PA guides the AA to the locked robot where the PA originated.

In the following sections, we describe the details of Ant agents and Pheromone agents in our distributed ACC algorithm.

### 4.1.3 Ant Agents

AA has IP list of all the robots in order to traverse them one by one. If it has visited all the robots, it goes back to the home host for administration of the robot system to check the number of robots, and updates its IP list in the following cases:

1. some new robots have been added, and
2. some robots have been broken.

However, those cases are so rare that the home host almost never interferes.

In addition, AA can observe the states of robots as follows:

1. it is being used by a customer,
2. it is locked, and
3. it is free, i.e., unlocked and not used by any customer.

When AA visits a robot that is used by customers or that is locked, it immediately leaves it without doing anything. If it visits a free robot i.e. not used and isolated, it begins to control it. At this time, if there is PA on that robot, AA makes the robot move following the guidance of the PA as shown by Fig. 4.3, otherwise AA drives the robot randomly as shown by Fig. 4.4.

Once the robot controlled by AA is locked because of reaching a suitable cluster, AA has to leave the robot and start migration again to find another free robot. Before that, AA creates PA if there is no PA on the robot, as shown by Fig. 4.4. As a result, PA starts behaving as a pheromone as shown in the next section.

### 4.1.4 Pheromone Agents

Since the purpose of ACC is to nurture clusters, the number of objects in a cluster affects the probability of picking up and putting down an object. Such a property of a cluster is modeled by a pheromone attracting ants.

The pheromone intrinsically has the following properties:
1. the strength increases in proportional to the number of objects,
2. the strength decreases in proportional to the distance,
3. it has a scope and does not affect out of the scope, and
4. the strength decreases as time elapses.

The state of a PA depends on the number of objects, the distance between the objects, and elapsing time. Robots acquire those data through the camera and the timer on them. The data are represented as a vector value inside PA.

4.1.5 The Migration of PA

PA is created by AA on a locked robot as shown by Fig. 4.4. Once PA is created, it clones itself, and the newly created PA migrate to other robot as shown by Fig. 4.5. PA has a vector value in it. The length and the direction of the vector value means strength of attracting and guiding direction relative to the front of a robot respectively. In Fig. 4.6, thick arrows represent vector value, where the direction of vector value on robot $A$ is represented as the angle $\theta$.

Considering scope size $S$ and hold time $T$, the absolute value of vector $v$ is represented as follows, where distance is the distance that the PA moved, time is elapsing time since the PA was born, $K$ is the maximal value that $v$ can take, and $C$ is a suitable coefficient:

$$|v| = \min(C \ast (S - \text{distance}) \ast (T - \text{time}), K)$$

$C$ is just used to adjust the equation for each circumstance, and $K$ is used to prevent the synthesized value from being too big.

Vector value $v$ is initially computed by PA itself after the migration as follows:

1. PA migrates from robot $A$ to robot $B$,
2. PA on $B$ observes $A$ where PA was born, gets the distance between $A$ and $B$, and the direction ($\beta$ in Fig. 4.6) from $B$ to $A$, and

3. PA sets these values as an initial vector value.

On the other hand, the direction after some migrations is not so simple. If PA migrates from robot $A$ to robot $B$ as shown in Fig. 4.6, direction $\theta'$ relative to the front of robot $B$ is computed by the following formula:

$$\theta' = \pi - \alpha + \beta + \theta$$

### 4.1.6 The Fusion of PAs

The moving robot controlled by AA can receive several PAs. In such a case, AA needs the consistent guidance of the PAs. Several PAs are fused into one PA as shown by Fig. 4.5. Since the data with PAs are vector values, they can be easily synthesized as shown by Fig. 4.7.
4.1.7 Experimental Results

In order to demonstrate the effectiveness of our distributed ACC algorithm, we have built a simulator for clustering robots (intelligent carts) and conducted experiments on it. On the simulator, moving and rotating speed of robots, and lags required in agent migration and object recognition are based on real values in the previous experiments [5]. In the experiments, we employed three wheeled mobile robots, which are called PIONEER 3-DX (Fig. 4.8), as the platform for our prototype system. Each robot has two servo-motors with tires, one camera and sixteen sonic sensors. The power is supplied by rechargeable battery. A PIONEER 3-DX has one servo-motor and sensor controller board that sends/receives data to/from a host computer through a USB cable. The camera is directly operated by the host with the middle ware which is called ERSP. Each robot holds one notebook computer as its host computer. Our control agents migrate to these host computers by wireless LAN.

We have conducted several experiments with different number of robots and AAs, and compared their results. Fig. 4.9 shows the average size of created clusters (bar graph) and the average time taken till convergence (line graph). As shown by the figure, the size of cluster seem to be around 8.5 % of the number of robots for any number of AAs. Also, the time till convergence for 50 AAs is equal to the time for 30 AAs though it is less than the time for 10 AAs. Fig. 4.10 shows the total length of traces of all robots. As shown in the figure, the less the number of AAs is, the shorter the length of traces is. Notice that the shorter trace means less energy consumption. These results demonstrate the beneficial features of our ACC, in which the energy consumption can be decreased on some levels without sacrificing efficiency.
4.1.8 Summary

We have proposed a framework for controlling mobile multiple robots connected by communication networks. In this framework, scattered mobile multiple robots autonomously form into several clusters based on the ant colony clustering (ACC) algorithm. The ACC algorithm finds quasi-optimal positions for the mobile multiple robots to form the clusters.

In our distributed ACC algorithm, we introduced two kinds of mobile software agents; i.e. ant agents and pheromone agents. The ant agents represent the artificial ants. They see the mobile robots as objects and drive them to the quasi-optimal positions. The pheromone agents represent pheromone and diffuse the effects by migrations. In general, making mobile multiple robots perform the ant colony optimization is impossible due to enormous inefficiency. Our approach
does not need the ant-like robots and other special devices. The preliminary ex-
periments shows that our approach is efficient enough and it enables suppressing
energy consumption.

So far we are not aware of any multi-robot system that integrates pheromone
as a control means as Deneubourg envisaged in his monumental paper [13]. The
preliminary experiments suggest favorable results. We will show the feasibility
of our multi-robot system using Ant and Pheromone agent base ACC by further
numerical experiments.

4.2 Pheromone Based Formation

We proposed a clustering method that does not need any leader. Next, we extend
this approach and propose a decentralized control algorithm to form any kinds
of shapes. The swarm robots are expected to compose formations that represent
symbols. The robots are connected by communication networks and not neces-
sary intelligent; they initially do not have any control program to compose sym-
bols. 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 such as ants. We have
implemented the ant and the pheromone as mobile software agents. Each ant
agent has partial information about the formation, and drive a robot based on the
information to compose it. 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 diuse attracting informa-
tion. 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.

4.3 System Overview

Our system consists of robots and two kinds of mobile software agents. We as-
sume that the robots have simple capabilities of movement such as rolling wheels,
measuring distance and angle through an optical camera and other sensors. In ad-
dition 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 Fig. 4.11 without any central coordinate that directs the other robots to compose a shape. 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 a shape. Paled color arrows represent attracting force by pheromone. We assume that objective shapes (the target shapes) 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 of a shape and drives a robot to the position. We assume that all the AAs are sent from the user machine to one of the reachable robots and each AA does not know its own absolute position but has partial information of the shape. For example, when the objective shape is a circle and AA A’s neighbors are B and C as shown Fig. 4.12, A has only information of B and C represented as arrows in the figure ($P_{AB}$ and $P_{AC}$).

For simplicity, in this paper, 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. It is not an essential requirement, because our system can be generalized by considering each robot’s direction and adjusting them. 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 Fig. 4.13. As shown in the figure, $\theta_A$ is the angle to robot B from robot A and $\theta_B$ 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 $\theta_r$ of angles between the two robots as follows:

$$\theta_r = 180 - (\theta_B - \theta_A)$$  \hspace{1cm} (4.1)
In a word, the agent rotate its data vectors that are based on A’s coordinate by the difference angle $\theta_r$.

### 4.4 Ant Agent

In this section, we describe the algorithm for AAs (Alg. 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 a shape. 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 $V_i$ 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 $n$
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

PAs with vector $V_i$, vector $V_t$ held by the AA is computed as follows.

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

For example, the objective formation is a line with three robots, which are currently located as shown in Fig. 4.14. 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 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 $V_t$ with the vector $V_{\text{another-to-dst}}$ from the another robot to the destination, which is computed by using vector $V_{\text{another}}$ to the another robot as follows:

$$V_{\text{another-to-dst}} = V_t - V_{\text{another}}$$

If another robot is idle and $|V_t|/2 > |V_{\text{another-to-dst}}|$, the AA migrates to it instead of driving the current robot along $V_t$. 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 $|V_t| > |V_{\text{another-to-dst}}|$, 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. However, it is not an essential requirement as mentioned. Each AA has only an individual relative coordinate system. Therefore, PAs store angle values of coordinate system of the attractor.
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

AA that generated them and the attractee AA receiving the PAs adjusts its coordinate system by using angle values of them. Let $\theta$ be the angle value that the AA has, $n$ be the number of the PAs that the AA receives and $\theta_i$ be an angle value that $i$th PA has. Once an AA receives a series of PAs with those angle values, the AA updates its angle value as follows:

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

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.

4.5 Pheromone Agent

In this section, we describe the algorithm for PAs (Alg. 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 (Fig. 4.15):
1. PAs clone themselves on the current robots.

2. PAs migrate to all the other robots within the view range of the current robots.

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 ids 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 $V_t$ is updated using vector $V_n$ to the destination robot of the migration as follows:

$$V_t \leftarrow V_t - V_n$$  \hspace{1cm} (4.2)

Fig. 4.16 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 $V_t$ points to the location to occupy represented by the black star. Once it migrates to robot $B$, $V_t$ would point to the location represented by the dotted star. In order to make $V_t$ keep pointing to the black star, the PA has to adjust $V_t$ through the calculation: $V_t \leftarrow V_t - V_{n_B}$, where $V_{n_B}$ 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 $D$. 

---

46
4.6 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.

1. Robots are scattered in an 800 × 800 square field in the simulator.
2. The reachable range of WiFi network for each robot is 150 units.
3. Each robot can move 2 units in each step in the simulator.
4. Interval time of generating PAs is 1 step.
5. Duration time of PAs is 10 steps.
6. The initial locations and angles of robots are randomly decided without overlapping.
7. Each robot is represented as a circle on the grid field.
8. Robots that have an AA are emphasized by large circle.

We confirmed that robots converge to the target formation as shown in Fig. 4.17 where the location and angle of the formation is determined by an initial condition. Fig. 4.18 shows the result of the formation that represents the letter A in different initial conditions from Fig. 4.17. As shown in the figures, the initial conditions determine the location and the angle of the resulted formation.

4.6.1 Migration

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 shape, we calculated the average distance $D$ as defined in (4.3). 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 $A_i$ and $I_i$, calculating the average distance $D$ by the following equation:

$$D = \frac{1}{n} \sum_{i=1}^{n} \|I_i - A_i\|$$

$I_i$ is the ideal relative coordinate of the $i$th robot from the barycenter and $A_i$ is the actual relative coordinate of $i$th robot from the barycenter. In the equation, $n$ is the number of AAs that compose a given shape. Figure 4.19 and 4.20 shows 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 Fig. 4.19, 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 Fig. 4.20, 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 another aspect, we fixed the number of robots to 40. Under the condition, we measured the duration time for convergence and length of movements and compared with those with no migration. As shown in Fig. 4.21 and 4.22, the time for convergence and total length of movements were reduced by the migration.
4.6.2 The Number of AAs Composing a Shape

We have examined the relation between the duration time for convergence and the number of AAs composing the shape, where we fixed the objective shape 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 shape 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 Fig. 4.19. 

Figure 4.19: Time for convergence for the different number of robots.

Figure 4.20: Total length of movements for different number of robots.

Figure 4.21: Time for convergence compared to migration off.
shows that the duration time for convergence increases as the 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 shape, the whole formation is affected. The change of the location of an AA is observed by neighbor AAs and then the neighbor 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 objective shape becomes larger, the number of time steps for propagating the changes of locations increases, so that the duration time for convergence also increases.

4.6.3 Comparison to Leader Based Approach

Figure 4.24 shows the average distances of this algorithm and the leader based approach for convergence time, where the horizontal axis is time and the vertical axis is the average distance calculated by (3.5). In the experiment, the objective formation is a circle with radius 150 units, the number of nodes of the formation is 20,
and the number of robots that are scattered in the field is 40. We assume that NAs that compose a formation are generated after the GA calculates the barycenter and the target locations based on it, and hence, the line chart for the leader based approach, which represents the change of average distances, starts from 100 time steps after calculation of the barycenter. As shown in the figure, the experimental result shows leader based approach is more efficient than the pheromone based approach even if it includes the time for calculating the barycenter and the target locations. This is because each NA knows its own target location to move, so that NAs can go straight to the locations.

On the other hand, the pheromone based approach determines the target locations dynamically. It continuously adjusts the current locations once they arrive around the targets. Indeed, in most cases, all NAs completed their task by the 140 steps, while the pheromone based approach took approximately 600 time steps to settle down. Also, for the same reason, the total length of traces of the leader based proposed approach becomes much shorter than the pheromone based approach. The trace of the leader based approach is 30 times as short as the pheromone based approach. The pheromone based approach takes 25000 units, while leader based approach takes 800 units.

4.6.4 Accuracy of Movement and Sensors

We have conducted experiments in the simulator assuming the ideal conditions. In such a condition, robots can move correctly with no error, and sensors or cameras can measure precise distances and directions to other robots. In the real world, some movement errors and sensor errors would be caused by imperfect hardware. We simulate sensor errors, movement errors and angle errors.

Figure 4.25 shows the influences of errors for the leader based approach and the pheromone based approach in the same error condition, where the objective shape is a circle whose radius is 150 units. The number of nodes of the shape is
20, and the number of robots is 60. Sensor and movement errors are 20 percent of sensed distance and movement length respectively in uniform distribution. Angle errors is added within [-10, 10] degrees in uniform distribution. In the leader based approach, the duration time of the task was not influenced by the errors. Since robots could not measure the precise locations, however, the eventual formation was distorted. The average distance of robots from actual locations to the ideal locations was 20 units.

On the other hand, although the pheromone based approach was also affected by errors, the eventual formation is less distorted than the leader based approach. This arises from the difference of the distance between attractor and attractee. In the leader based approach, NAs that are attractors tend to move to the target locations that are attractees in a long way repeating migrations, so that errors are accumulated. On the other hand, in the pheromone based approach, each agent just attracts other agents to neighbor locations. Thus, the leader based approach would be suitable for composing formations with not so many nodes for not so big shape.

4.6.5 Feature of Pheromone Based Approach

Furthermore, to examine the effect that the shape of a formation gives to the convergence time, we conducted extra experiments for letter G as shown in Fig. 4.26. The contour of the letter is similar to a circle, but unlike the circle, the outer line of letter G is not closed. Thus, the change in G is propagated in the single direction, while the change in the circle is propagated in two directions. This means that the convergence time of G is longer than a circle. Actually, as shown in Fig. 4.27, letter G took twice as much time as circle for convergence, where the width and height of the G are both 400, the diameter of the circle is 400 and the number of AAs is 31.
4.6.6 Extending the Number of Neighbor Robots

Although the pheromone based approach is able to work without the risk of blocking completion of a task, it takes longer time than the leader based approach. Some situations may require shorter duration time for convergence at a cost of decentralization. We make our algorithm flexible to modify the degree of distributed computation. Each AA has relative location information of AAs to come as neighbors within the area around it. Extending the area, our algorithm can decrease the time for convergence due to more information of the objective shape. We used the setting in which the number of neighbors to 2, 4, 10 and 19 for a circle with 150
Figure 4.28: Time for convergence for different number of neighbor AAs.

units radius and composed by 20 AAs. The case for 19 neighbors means that each AA has whole information of the objective shape. Figure 4.28 shows that the more neighbor AAs are, the less time it takes for convergence. As shown in the figure, the case for 19 neighbors showed a great improvement for the convergence time, it suppress the time about 5 times compared with case 4 neighbors. Notice that the increase of the number of neighbor AAs does not mean that central control is required. We only make each AA have extra neighbor AAs’ information and send PA to the neighbor AAs.

4.7 Summary

We have proposed a decentralized algorithm for the formation control for swarm robots using mobile agents. In this algorithm, we introduced two kinds of mobile software agents; i.e. 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 effectiveness of our algorithm, we have implemented a simulator on which we have conducted some experiments. We have shown that the algorithm can achieve not only the distributed formation control but also suppressing energy consumption through the experiments.

On the other hand, in some case where the target formation has a long part like letter G, the time for convergence increases compared with other kind of shapes. In order to solve this problem, we may need to make each ant agent has more global information about the formation, even though that means it partially
sacrifices the decentralization.

Also we conducted an experiment compared with the leader based approach. In this experiment, although the pheromone based approach takes more time, the leader based approach have a risk where the leader agent is lost and therefore, the formation task cannot be completed. If necessary, pheromone based approach gives each Ant agent the locations of all the nodes as neighbor locations to reduce time instead of the locations of partial nodes. Moreover, pheromone based approach have high tolerance to errors such as sensor errors in real world.
Chapter 5

Predictive Formation

This chapter presents an extended distributed control algorithm for composing formations of swarm robots, which considers the time lag that migrations of pheromone agents cause. When an ant agent receives a pheromone agent, the ant agent that generated the pheromone agent may have moved because of the time lag. Thus the pheromone agent may misguide other ant agents in attracting. In this extended approach, the attracted ant agent predicts the location of attracting ant agent, and migrates or drives a robot to the predicted location. We have implemented a simulator based on our extended algorithm, and conducted numerical experiments to demonstrate the feasibility of our extended approach. We have shown that our new approach saves total mileages of moving robots in over 10 percent compared with our pheromone based formation.

5.1 Predictive Method

In this section, we describe the extended part of our predictive formation method. In predictive formation, we consider the time lag that arises from the migrations of PAs from attractor AAs to attractee AAs. For example, AA \( B \) attracts AA \( A \) to the neighbor location of \( B \) to which \( V_t \) points as in Fig. 5.1 (a) and AA \( B \) is moving with the velocity \( V_m \). In this figure, the circles represent the robots with the AAs and the star represents the location for AA \( A \) to occupy. If AA \( B \) generates a PA with AA \( A \) as an attractee at time \( t \), the PA has the location for \( A \) to occupy that is represented as the star in Fig. 5.1 (a). When AA \( A \) receives the PA at time \( t_n \), it is considered that AA \( B \) advances about \((t_n - t)V_m\) as in Fig. 5.1 (b) because AA \( B \) was moving with speed \( V_m \) at time \( t \) and the duration time \((t_n - t)\) is elapsed. In this case, it is considered that AA \( A \) is desirable to move toward the star in Fig. 5.1 (b) instead of the dotted star in Fig. 5.1 (b). In the predictive approach, attractee
AAs predict the locations to occupy and update $V_t$ as follows:

$$V_t \leftarrow V_t + (t_n - t)V_m$$

If the robot that generates the PA is close enough to the location to occupy, the prediction is not applied because the prediction might hinder the fine adjustments of their locations. Notice here that whenever PAs migrate to another robot, $V_m$ has to be updated as well as the vector value of PAs in (4.2) as follows:

$$V_m \leftarrow V_m - V_n$$  \hspace{1cm} (5.1)

$V_n$ is the vector from the source robot to the destination robot. The differences of angles of coordinate systems are also transferred as (4.1).

To enable the prediction, AAs generate PAs and store the information of the movement vector of the robot with the attractor and the current time into the PAs. They are $V_m$ and $t$ respectively. After that, the PAs transport the information as translating the vectors through (4.1), (4.2) and (5.1).

### 5.2 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:

1. Robots are scattered in an 800 x 800 square field in the simulator.
2. The reachable range of wireless network for each robot is 300 units.
3. Each robot can move 2 units in each step.
4. Interval time of generating PAs is basically 2 steps.
5. Duration time of PAs is 100 steps.

6. The initial locations and angles of robots are randomly determined without overlapping.

Fig. 5.2–5.5 show outputs of the simulator for different settings, where each robot is represented as a black circle on the field and a robot that has an AA is represented as a colored circle.

To demonstrate that the pheromone based formation and the predictive formation makes the robot formations converge to various objective shape, we have conducted the experiments that compose letter A formation, letter G formation, line formation and lattice formation. Letter A is composed of 20 robots and letter G is 31, lattice is composed of 25 robots in 100 units intervals and line is 20 in 30. In every experiment, used robots include 20 redundant robots other than ones used for composing a formation. In our previous paper, we showed that the redundant robots that were not directly used for composing the formation gave flexibility to movements of ant agents through their migrations, and contributed to improving efficiency of our algorithm. For example, the experiment of formation of letter A had 40 robots in total because letter A used 20 robots for the formation and 20 redundant robots were added. Each figure shown in Fig. 5.2–5.5 display the result corresponding to each shape.

In order to measure the degree of correspondence with the objective ideal shape quantitatively, we calculated the average distance $D$ as follows. 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 $A_i$ and $I_i$, calculating the average distance $D$ by the following equation:

$$D = \frac{1}{n} \sum_{i=1}^{n} \|I_i - A_i\|$$

(5.2)
Figure 5.4: Formation of line.  Figure 5.5: Formation of lattice.  

![image](image1.png)  ![image](image2.png)  

Figure 5.6: Comparison in letter A. 

\( I_i \) is an ideal relative coordinate of each robot from the barycenter and \( A_i \) is an actual relative coordinate of each robot from the barycenter. In the equation, \( n \) is the number of AAs that compose a given shape.

In our previous work, we showed that performance of the pheromone based formation was closely related to the distribution and the connection of neighbor robots. Thus, in these experiments, we have selected letter A, letter G, line and lattice shape to compare with the pheromone based formation because their distributions of neighbor robots are contrasting.

We observed the degree of correspondence with the objective shapes in the four shapes using (5.2). Fig. 5.6, 5.7, 5.8 and 5.9 are the degree of correspondence with the objective shapes for letter A formation, letter G formation, line formation and lattice formation respectively. The vertical axes show average distances to the objective formations that are calculated by (5.2). These axes mean the degree of convergence, where the lower the degree is, the more convergent the formation is. The horizontal axes show duration times on the simulator. As shown in the figures, for both algorithms of the pheromone based formation and the predictive formation, the convergence of all the formations was completed, where the predictive formation was more efficient than the pheromone based formation for all the formations.

59
Table 5.1 shows the total moving mileages of robots in formations letter A, letter G, line and lattice in the cases where the simulator runs in 2000 steps for letter A, line and lattice and 4000 steps for letter G till their convergence. Notice that the formation of letter G takes longer time to finish the tasks than the others. These results show that the predictive formation saves over 10 percent of the total moving mileage of robots for the pheromone based formation.

In the experiments mentioned above, the interval time of generating PAs is 2 steps. To observe the effects of the interval time on both approaches, we have conducted the experiments in which we have changed the interval time within 2, 4, 8 and 16 for the formation of letter A. Fig. 5.6, 5.10, 5.11 and 5.12 show the results of the duration time for interval time 2, 4, 8 and 16 respectively. Also, Table 5.2

<table>
<thead>
<tr>
<th></th>
<th>Letter A</th>
<th>Letter G</th>
<th>Line</th>
<th>Lattice</th>
</tr>
</thead>
<tbody>
<tr>
<td>previous(units)</td>
<td>7193</td>
<td>14812</td>
<td>8597</td>
<td>9603</td>
</tr>
<tr>
<td>new(units)</td>
<td>6205</td>
<td>13237</td>
<td>7483</td>
<td>8408</td>
</tr>
<tr>
<td>saving rates(%)</td>
<td>13.735</td>
<td>10.633</td>
<td>12.958</td>
<td>12.444</td>
</tr>
</tbody>
</table>
Figure 5.9: Comparison in lattice formation.

Figure 5.10: Comparison of letter A with interval 4.

shows the saving rates of total moving mileages of robots for these intervals. As shown in the figures and table, the predictive formation approach is still efficient than the pheromone based formation for all the intervals, but the improvement rates become less in proportion to the interval times. In the case of interval 16, Table 5.2 shows that there is not so much difference of the total moving mileages of robots between the predictive formation and the pheromone based formation. This suggests that the longer interval time is, the more the difference between the real location and the predicted location becomes. Thus, to obtain maximal improvement by the predictive formation approach, it would have to run in the same interval time as the pheromone based formation one.

Table 5.2: Comparison of letter A with various interval.

<table>
<thead>
<tr>
<th>Interval</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>previous(units)</td>
<td>7193</td>
<td>7167</td>
<td>7571</td>
<td>7950</td>
</tr>
<tr>
<td>new(units)</td>
<td>6205</td>
<td>6732</td>
<td>7314</td>
<td>8056</td>
</tr>
<tr>
<td>saving rates(%)</td>
<td>13.735</td>
<td>6.069</td>
<td>3.394</td>
<td>-1.33</td>
</tr>
</tbody>
</table>
5.3 Summary

We have proposed an effective extension of our previously proposed decentralized control algorithm for composing specific formations of swarm robots. In this extension, we consider the time lag of the diffusion of pheromone agents and conduct predictions to compensate the lag.

In the algorithms, we have introduced two kinds of mobile software agents; i.e. ant agents and pheromone agents. In the previous approach, the ant agents generate the pheromone agents that have local information about the formation to guide other ant agents to the appropriate locations to form the target formation. In our new approach, we extend the pheromone agents so that they have movement vector of the robot and birth time to predict the future locations of the robots in order to decrease the influences of the time lag.

Once pheromone agents are generated, the pheromone agents clone themselves and make them migrate to other robots to find the attractee ant agents. When ant agents receive the pheromone agents, which have information to guide the ant agents, the ant agents can predict the locations that the pheromone agents point to and then move to the locations. Eventually the whole robots compose the
target formation.

In order to show the effectiveness of our algorithm, we have implemented a simulator and conducted numerical experiments on it. We have shown that our new algorithm is not only capable of achieving distributed formation control but also more effective in suppressing the time for convergence and energy consumption. We have achieved to save the total moving mileages of robots over 10 percent in four shapes. When interval times of pheromone agents become longer, the improvements of our new approach become lesser although our new approach has advantages over our previous approach. This is because in the case of longer interval the predictions become inaccurate. As the future works, we are planning to invent more accurate prediction algorithms than current one.
Chapter 6

Conclusions

In this chapter, we give some discussions for approaches proposed in this paper, and conclude remarks.

6.1 Discussions

In this paper, we have proposed several mobile agent based formation approaches. Our approaches can accurately arrange mobile robots to the places that some robots should occupy in a shape for formation. At this time, each robot is not reserved for a specific position in the shape. Instead of that, 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.

Our approaches can roughly be categorized into two kinds of strategies: leader based strategy and pheromone based strategy. In the leader based strategy, a leader mobile agent traverses all the robots to collect their location information, and then, order other mobile agents to drive robots to specific locations for formation relatively to the barycenter of the robots. On the other hand, in pheromone based strategy, there are Ant agents as many as positions in formation, which have the information of the locations for neighbor Ant agents to occupy, and attract the neighbor Ant agents with robots to the locations using Pheromone agents, which we proposed for clustering.

As shown by the experimental results in the paper, the both strategies work well, but they have trade-off. In the leader based strategy, the locations for robots to occupy are directedly decided based on the location information collected by a leader agent, while the pheromone based strategy needs duration time to decide
them. Therefore, the leader based strategy is more efficient than the pheromone
based strategy. On the other hand, in terms of robustness, the leader based strategy
is weaker than the pheromone based strategy. The both strategies continuously
work even if some robots have a trouble for their wheels or motors to rotate them,
because each agent can migrate to other robot as soon as it knows that the robot
it drives does not move. However, if a computer executing an agent is broken
down, the agent itself would be lost. In the same situation, once a leader agent is
lost, the leader based strategy cannot compose any formation. On the other hand,
in the pheromone based strategy, as long as each Ant agent has information of
extra neighbor Ant agents, the pheromone based strategy can compose formation
without any distortion although the formation lacks some robots on the shape.

In order to mitigate the trade-off, we have proposed an extension of the naive
pheromone based approach. The extended approach decreases convergence time
for formation by predicting target position aberration caused by time lag. We
have shown that the extension contributes to efficiency of the convergence time,
but there is still great difference in efficiency between the extended approach and
the leader based approach.

Through the observations, we can say that we have to decide which strategy
should be used depending on the level of emergency of each mission.

6.2 Conclusions

We propose a formation control method that have following features: it does not
have central computation, it can effectively use robot resources, it can work with
sensor errors or movement errors, it can construct arbitrary shapes that allows
various density inside the shapes, and it can continue the task when some robots
break down except for computers and communication devices because mobile
agent can migrate other unbroken robots.

6.3 Future Work

As a future work, we intend to extend our method to three dimensional formation
control that can be used in reconfigurable robots, outer space or drone formation.

Our approach can continue tasks when robots can move because of some trou-
bles by migrating other robots. However, once a leader agent or some Ant agents
are lost, our current approaches cannot form a complete shape. In the future, we
want to give a method to address this problem. For example, since the pheromone
based approach can form a shape with some lacks of nodes, Ant agents corre-
sponding to the lost nodes 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 nodes.
Acknowledgment

I would like to express the deepest appreciation to Dr. Takimoto for his kind guidance. For a long time, he patiently provided direction when I got stuck at my work and gave me proper advises. He checked my work despite it being a holiday when I was late my dead line. I never go through this work without his great dedication. And then, I would like to thank Dr. Kambayashi from the Nippon Institute of Technology for his penetrating insight. He also corrected my poor English at editing my papers of international conference and give me polite advice. Finally, I sincerely thank to my friends and members of Takimoto laboratory for my productive life in the laboratory and my family for moral support for a long time in my university life.
Bibliography


[27] 大須賀昭彦. エージェントの研究動向. 人間主体の知的情報技術に関する調査研究. (財) 日本情報処理開発協会 先端情報技術研究所.
