Adaptive Task Allocation in Multi-Agent Systems Based on Swarm Intelligence

Wonki Lee

The Graduate School

Yonsei University

School of Electrical and Electronic Engineering

Adaptive Task Allocation in Multi-Agent Systems Based on Swarm Intelligence

A Dissertation Thesis
Submitted to the School of Electrical and Electronic Engineering
and the Graduate School of Yonsei University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Wonki Lee

February 2018

This certifies that the dissertation of
Wonki Lee is approved.

Dissertation Supervisor: DaeEun Kim

Jin Bae Park

Euntai Kim

Seong-Lyun Kim

Dongseok Sun

The Graduate School

Yonsei University

February 2018

Acknowledgements

This thesis work would not have been possible without the support of many people. It is a pleasure to thank to all of those who supported me in all the respects during the completion of my thesis project. I am heartily thankful to my supervisor, Prof. DaeEun Kim, whose encouragement, guidance, and support from the initial to the final level enabled me to develop an understanding of the subject. I like to show my gratitude to Prof. Jin Bae Park, Prof. Euntai Kim, Prof. Seong-Lynu Kim and Dr. Dongseok Sun for critical reading and comments on my thesis. All the group members at the Biological Cybernetics Lab. have helped and inspired me. Finally, I thank my parents and my family for supporting me throughout all my studies. This thesis would not have been possible without their love and care.

Contents

List of Figures

List of Tables

ABSTRACT

Adaptive Task Allocation in Multi-Agent Systems Based on Swarm Intelligence

Wonki Lee

School of Electrical and Electronic Engineering
The Graduate School
Yonsei University

In nature, it is well known that social insects, such as honeybees, termites, and ants, use task-partitioning strategies for their survival. Many interactions between many different individuals can be assumed as a complex network but lead to an efficient colony-level performance. They create an effective division of labor using just a few basic behavioral rules for individual agents with limited abilities. Each agent performs a single task among multiple possible tasks depending on the current needs of the swarm system and different tasks are performed simultaneously by specialized individuals at different locations and with little interruption. This tendency to specialize is effective managing the division of labor in multi-agent systems, especially when there is a coat associated with switching tasks.

In this dissertation, the concept of swarm intelligence inspired by the division of labor in several social insect species has been applied to solve scheduling problems in multi-agent systems. These tasks may handle dynamic environments and require not only convergence

to the desired level of performance but also coverage, robustness and fault tolerance. In particular, we suggest a task allocation algorithm to regulate the fraction of agents relative to the fraction of task demands and re-assignment in multi-agent systems. For one possible solution, we present a biologically inspired approach based on the response threshold model, in which the tendency of an individual to select a single task among candidate tasks is determined depending on task need and the corresponding response threshold value. Obtaining a suitable division of labor in the response threshold model is mainly related to the suitable thresholds.

Here, we introduce various task allocation methods that each agent updates its response threshold using information obtained from the surrounding environment, such as what objects have been observed recently and what tasks have been done by neighboring agents observed in the local surrounding area. It relies neither on a centralized mechanism nor on communication aids between robots and can therefore be employed easily in a swarm of robots. Repeated threshold regulations create an agent specialization for particular tasks by lowering the threshold values of tasks, and this tendency induces division of labor within the group. Task changes of individual agents lead to the swarm-level emergence toward a desired division of labor. We also analyzed the mathematical convergence of task distribution among a swarm of agents.

Various experimental results using demonstrate the method's ability to adapt the swarm to environmental changes such as changes in swarm membership or task demands. As demonstrated in the experiments, the system approximately converges to the desired task distribution, and this provides insight into how swarm intelligence self-organizes even when using local information. The method's tendency to specialize also minimizes the number of task changes needed to produce the desired task distribution over agents. In addition, we analyze the effect of task allocation strategies as a self-organized method for decomposing a task into sequential subtasks. Using data from various experimental cases, we show that if there is a transfer bottleneck at a central location, task partitioning can sometimes be an effective strategy for reducing the traffic jam and improving the overall performance of the group. Our results support the hypothesis that social insects use various task allocation strategies to increase their probability for the colony survival.

Key words : task allocation, multi-agent system, swarm intelligence, response threshold model, mathematical convergence, specialized tendency

Chapter 1

Introduction

In recent years, there has been increased interest in developing a method to perform various tasks with multi-agent systems, which is generally composed of a set of agents that share the same set of behavioral rules and perform tasks in different places. Such systems have many related applications, including cooperation in handling of parts or transport, the exploration of unknown areas, search and rescue, and surveillance operations. These tasks may need to be performed in dynamic environments and require not only convergence to the desired level of performance but also coverage, robustness, and fault tolerance.

Among many tasks performed with multi-agent systems, distributed task assignment problems have received significant attention. It is an interesting and important process used when performing complex tasks. The basic problem is that a group of agents should select a specific task to ensure an optimal distribution among multiple tasks. Many methods for solving these problems have been studied, including game theory-based negotiation algorithms [11, 46], dynamic target assignment problems [37, 202], quantized consensus over a network [105, 61], or stochastic policies for task allocation [19, 153]. We present a biologically inspired approach to dynamic task allocation and re-assignment in multi-agent systems. One of the advantages of bio-inspired systems is their ability to respond to an external stimulation in a simple and immediate manner. Due to the specialized characteristic to perform a specific task among tasks, this system can overcome the limitation of a single agent because multiple agents carry out various tasks simultaneously at several locations. The method uses the response threshold model to determine the tendency of an individual

to select a single task among multiple candidate tasks depending on the response threshold.

1.1 Why bio-inspired task allocation

Colonies can be characterized by cooperation, communication between individuals and the ability to solve complex problems. One interesting characteristic often found in insect colonies is division of labor. Colonies adapt themselves to dynamically changing environments. Individual workers perform different types of jobs at different places. However, they need to meet new demands and/or respond to changes in demand levels [88]. Colony-level flexibility in responding to external and internal perturbations is a crucial characteristic for a colony's survival [186, 103, 194]. It is well known that social insects can coordinate their behaviors to accomplish desired tasks beyond the capability of a single individual. There are many examples of division of labor based on task partitioning: ants can collectively carry large prey, termites can build large and complex mounds, and bees can build delicate honeycombs [186, 187, 197, 183, 64, 211, 191, 182].

Many social insects show an effective division of labor using just a few basic behavioral rules for individual agents with limited abilities. Each agent performs a single task among multiple tasks depending on the current needs of the swarm system and different tasks are performed simultaneously specialized individuals who perform their own tasks at different locations and with little interruption. This specialized tendency is effective for managing the division of labor, because there is often a cost associated with switching task [28, 119, 130]. The tendency toward division of labor in several insect societies is found in colonies adaptable to dynamically changing environments. Many social insects use various task partitioning strategies to deal with tasks required for colony survival such as foraging and garbage disposal. Foraging and nest defense in ants [51] and foraging and nursing in honeybees [218, 29] are all examples of labor division.

1.2 Motivation and objective

In this dissertation, the concept of swarm intelligence inspired by the division of labor in several social insect species, has been applied to solve scheduling problems in multi-agent systems [200, 160, 77, 144, 234, 109, 130]. The behavior of an individual agent may have little effect in a given environment and the same behavior might not have the same effect

when it is performed at a different place or time. However, repetitive and continuous task selections can lead to the desired level of performance in the overall system level. It also has the strength of being free from the risk of an individual failure. Sometimes, this tendency leads to much simpler and adaptive systems than the other theoretical control algorithms. Hence studying the behavioral mechanisms of animals and applying algorithms derived from this behavior computationally is a potentially future line of inquiry.

Among various interesting tasks, we consider a task allocation algorithm for regulating the proportion of agents performing tasks relative to the proportion of task demands. This relies neither on a centralized mechanism nor on communication aids between robots. For various task allocation methods, each agent updates its response threshold using information obtained from the surrounding environment such as the associated task demand and the task states of neighboring agents. Thus, the task transition rate from one task to another is regulated adaptively depending on the environment and can ultimately produce convergence to a stable task distribution. The result provides a hint of how swarm intelligence self-organizes even using local information.

We also analyzed the mathematical convergence of task distribution among a swarm of agents. Interaction rules are automatically generated by maximizing the performance evaluation function of the overall system. We used the foraging task. Tasks involving multiple robots can be observed in many real-world scenarios, including rescue, mining, and exploration [36, 199, 30, 85]. The foraging task is effective for demonstrating the performances in multi-agent systems. From the various results with swarm robotics, we demonstrate that the method is effective and robust in a dynamically changing environment. There are many examples of such foraging tasks using multi-robot systems, and various task allocation methods using an adaptive process have been proposed [100, 198, 84, 178, 3].

1.3 Organization of dissertation

The organization of this dissertation is as follows. This chapter outlines our rationale for choosing a bio-inspired model for task allocation. Chapter 2 describes the background for this research, including various task allocations that occur in nature and their applications to multi-agent systems. Many kinds of bio-inspired tasks have been tested using swarm robotics and task partitioning has proven to be one of the most challenging.

Chapter 3 explains the basic concept of task allocation based on a response threshold

model. The individual decision to switch among multiple tasks depends on the response threshold, which makes different response to the same task demand. In this model, each agent responds by performing a given task if the demand of that task exceeds its own corresponding response threshold. Different responses to the same task demand are generated by the different response thresholds, determining the preference of an individual assigned to a given task whether to perform that task. Obtaining a suitable division of labor in the response threshold model is mainly achieved by setting suitable thresholds, and two kinds of approaches using fixed and variable thresholds are tested. The thresholds are either constant (in the fixed threshold model) or change during task performance (in the variable threshold model).

We handle dynamic task allocation in the object foraging task, in which multiple robots are tasked with collecting various objects in parallel during a given time span. All agents can perform several tasks, and individual agents determine whether to perform a specific task depending on the environmental stimulus and on each individual's response threshold for each task. We tested the swarm robots using the foraging task in order to find an efficient strategy as well as to understand the foraging behavior of social insects.

In chapter 4, we examine an improved task allocation algorithm for parallel multi-purposed agents. Here, we introduce a task selection mechanism in which the response threshold is updated based on what objects have been observed recently and what tasks have been performed by an individual's neighboring agents observed in the local surrounding area. The robot's individual perception is regulated by the relative difference between the proportion of each task and the proportion of robots performing the corresponding task. We also analyzed the convergence of task distribution. Task distribution can be described in terms of task transition rate. Swarm-level task distribution can be controlled by changing the tasks of individual agents. Such local task changes lead to the swarm-level emergence of an efficient division of labor. Especially for our foraging task, the system will approximately converge to the desired task distribution as demonstrated in the experiments. We suggest a task allocation algorithm for regulating the fraction of agents proportionally to the fraction of task demands.

Our proposed algorithm shows its capability to adapt the swarm to environmental changes such as changes in swarm membership or task demands. It also produces the desired task distribution across agents and minimizes the number of task changes due to the tendency to specialize. The tendency to perform a specific task is computed using the task demand and the corresponding response threshold. A lower task threshold or a higher task demand

leads to a higher tendency to perform the task and this tendency leads to the specialization of a given task. Repeated threshold regulations create an agent specialization for particular tasks by lowering the threshold values of tasks, and this tendency induces division of labor within a group.

For application to the factory-domain problem, an agent-based schedule algorithm as applied to a truck-painting job assignment in a vehicle factory is studied. In a real manufacturing system, many parallel machines can process several tasks, but it is beneficial to minimize machine state changes because each change incurs additional time and material costs. Thus, the objective of scheduling algorithms is to minimize the total number of task changes in the system while maintaining the desired performance level (as measured by, for instance, throughput). Every task is assigned through a bidding process among agents, and the repeated task assignment patterns induce each agent to tend to specialize in a specific task. It affects to reduce the number of task changes. Various experimental results show that our approach is comparable with that of other conventional methods.

In chapter 5, we present a self-organized method for decomposing a task into sequentially interdependent subtasks and allocating the individuals in a swarm to perform subtasks in parallel, using a response threshold model in which the robot's individual perception is regulated by the relative difference between the number of tasks not completed and the number of robots performing the corresponding subtask. Instant information is used to decide the task. We use a foraging task in which a group of agents are tasked with collecting food pellets and bringing them to the central nest. Harvesting agents transport pellets from the resource area to a common storage location called the cache area (transfer area), and storing agents transport them from the common storage to the central nest. This approach has the effect of reducing interference among the individual agents because they become more segregated and improved transport efficiency allows for better overall swarm performance. We also show the method's capability to adapt to dynamic changes in the environment and to converge to the desired task distribution even in case of multiple stage sequential tasks.

In addition, we analyze the effect of the various task transfer strategies using data from various experimental cases by changing the size of the nest entrance, the number of food sources, and the number of foraging robots. Our specific hypotheses are as follows: what is the important factors that influence foraging performance?; and which task partitioning strategies will be helpful depends on the environmental conditions? From the experimental results, we show that if there is a transfer bottleneck at a central location, task transfer can sometimes be an effective strategy for reducing the traffic jam and improving the overall

foraging performance of the group. In particular, task transfer sequenced from the slowest agents to the fastest agents can especially improve performance in an environment with multiple food resources, several routes converging to the nest, and a narrow nest entrance.

Finally, Chapter 6 presents conclusions about the proposed models. In this chapter, the whole dissertation is also summarized, and potential future lines of inquiry are discussed.

Chapter 2

Background

Multi-agent systems can be used to perform various dynamic tasks. This area of research generally has focused on a large group of relatively simple agents that can solve a complex problem. Each member of a group should determine the current task that is most commensurate with its current surrounding labor states. Generally, there are two issues involved in performing tasks: the cooperation and the division of labor among agents in a group. Cooperation involves performing a complex task as a group of agents instead of improving the ability of a single individual, while the division of labor involves efficiently managing tasks that are costly and time intensive. One possible approach for solving these types of problems is to establish a dynamic task allocation mechanism using adaptive processing that independently adapts all agents in a group to a dynamically changing environment. The basic requirements for achieving this are to maintain the specialized individuals and to process tasks in parallel.

In recent studies, agent-based approaches using swarm intelligence inspired by several social insect species, have garnered attention as a potential solution to the multi-agent systems problem. This concept is useful for designing a task allocation model in a multi-agent system. A single individual has limited hardware capacity and only a few behavioral rules. However, the overall system can effectively perform more difficult tasks because this model has the advantage of concurrency and fault tolerance. It enables both the simultaneous performance of tasks in several locations and flexibility with regard to individual failure in a large-scale system.

In this chapter, an overview of task partitioning in biology and its application to multi-agent systems, particularly swarm robotics is given in detail.

2.1 Swarm intelligence

Swarm intelligence refers to a subset of artificial intelligence that studies self-organizing, decentralized collective behaviors exhibited by multi-agent systems composed of several locally interacting agents. It is the direct result of self-organization, in which the interactions of low-level components create a global-level dynamic structure that may be regarded as intelligence [22]. These lower-level interactions are guided by a simple set of rules that individuals of the colony follow without any knowledge of its global effects, and without the help of a leader or external control. Individuals in the colony only have local-level information about their environment. Using direct and/or indirect methods of communication, local-level interactions affect the global organization of the colony.

Generally, it is believed that self-organization is induced by four elements: positive feedback, negative feedback, randomness, and interaction. Positive feedback is defined as the first rule of self-organization. It is basically a set of simple rules that help to generate the complex group structure. Recruitment of honey bees to a promising flower patch is one example. Positive feedback can help the overall system converge to one of its possible stable states, although it can also potentially destabilize the system. The second element is negative feedback, which reduces the effects of positive feedback and helps to create a counterbalancing mechanism by biasing the system toward stable states and dampening some fluctuations. The number of limited foragers is an example of negative feedback. Randomness is the third element in self-organization. It adds an uncertainty factor to the system and enables the colonies to discover new solutions to their most challenging problems. Lastly, there are multiple interactions between individuals. There should be a minimum number of individuals who are capable of interacting with each other to turn their independent local-level activities into one interconnected living organism. As a result of combinations of these elements, a decentralized structure is created. In this structure there is no central control, even though there might seem to be one. This creates dynamic and efficient structures that help the colony to survive despite many challenges.

The main inspiration for multi-agent systems comes from the observation of social animals. The behaviors of ants, bees, birds, and fish provide examples of how simple individ-

uals can become successful when they act as groups. The interest in social animals stems from the fact that they exhibit a sort of swarm intelligence. The behaviors of groups of social animals are robust, flexible, and scalable. By taking inspiration from social animals, various systematic applications of scientific and technical knowledge can be studied.

2.1.1 Swarm intelligence in nature

The most striking cases of swarm intelligence are exhibiting complex and collective behaviors without any centralized decision-making process [31]. For example, many species of ants live in organized in societies. Some agents of the colony are assigned to explore the environment: they look for food and collectively select the best resource to bring to their nest [15]. Other agents are capable of collectively building complex nests, creating tunnels and chambers for food and dumping garbage. Other examples of complex behaviors in social insects are exhibited by honey bees. For instance, they are capable (as are ants) of regulating the temperature of their hive by altering their own individual temperature [206]. These decisions are made using purely distributed approaches, without centralized coordination or explicit negotiation. All individuals select a suitable task locally, with no clear picture of what is going on at the level of the colony.

Many other species exhibit another type of collective behaviors: coordinated motion. Examples of coordinated motion can be observed in birds and in fish. They aggregate and move together in large groups that behave as if they were single organisms. They can also change shape by expanding or contracting [112]. This coordinated motion can be also observed in other animals. Mammals such as sheep perform coordinated motion by a few individuals with ability to recognize food resources [175, 136]; and crowds of human pedestrians have been shown to exhibit global patterns such as lane formations [164]. In these behaviors, there is no external guidance nor leader agent, although some individuals might have better information and transfer this information to the rest of the group in direct or indirect ways [45].

2.1.2 Applications of swarm intelligence

Inspired by various collective behaviors found in nature, artificially engineered systems have been proposed for solving real-world problems. The performance of swarm intelligence is not as impressive as that shown by computational intelligence methods. However,

the applications are growing rapidly. Some potential applications of swarm intelligence are discussed below. Optimization involves maximizing or minimizing a real function. Systematically choosing input values within an allowed range and computing the value of the function be achieved solved by searching with multiple software agents in parallel. Examples of such systems include ant colony optimization [52] and particle swarm optimization [106]. Both algorithms are inspired by naturally evolved swarm intelligence systems, specifically ant colonies and bird flocks. Another example is swarm robotics systems, presented in more details in Section 2.2.

Routing is a distributed algorithm in telecommunication networks. In routing, each node decides to which node each packet should be sent. Ant Net [52] was inspired by the behavior of ants and showed improved performance compared to other algorithms. Another application area is vehicle routing. Routing of hundreds of vehicles has been implemented using technology inspired by swarm intelligence [71, 148].

Clustering is a problem related to data analysis that consists in grouping data in order to have similar to each other items together in the same cluster and different from each other items in different clusters. Ant-based clustering algorithms [82], also inspired by the behavior of social ants.

The military is one of the most promising areas for the application of swarm intelligence, allowing for various missions to be carried out at low cost and with robustness by, for instance, developing a swarm of small unmanned aerial vehicles. Surveillance unmanned aerial vehicles (UAVs) could keep watch over a convoy, landing for refueling on one of the trucks. Working together as one team, they could perform surveillance around the convoy. Another application is implementing behavior like cooperative hunting. A swarm of UAVs could search for targets while trying to avoid detection. By optimizing flying patterns by combining shared sensing information, they are capable of searching larger area than an uncooperative group. Swarm intelligence based control algorithms can introduce fault tolerance to improve coverage of unfamiliar and difficult terrain.

Scheduling is the optimizing jobs in a manufacturing process. It is used to allocate plant and machinery resources, plan human resources, plan production processes and purchase materials. In some applications, swarm intelligence has been shown improved performance over conventional methods. The application of swarm intelligence based scheduling for truck painting in a factory context is analyzed in more details in Section 3.4 of Chapter 3.

2.1.3 Characteristics of swarm intelligence

Swarm intelligence systems have characteristics that make them preferable to centralized systems. Scalability is the potential of a swarm to show the expected the level of performance for increasing swarm sizes. It is promoted by the intrinsic tendency of swarm intelligence systems to be generally based on local interactions among the components of the system. The number of interactions per system does not change relative to swarm size, and no consideration of variation in swarm size is needed.

Parallelism refers to the capability to perform multiple tasks in parallel at the same place or in different places. For instance, colonies of leaf-cutting ants [9] perform different activities in parallel: cutting leaves, collecting leaf pieces, and then storing them. Multiple individuals perform different tasks at the same time, improving the overall performance [42]. More examples found in nature are explained in Section 2.3.1.

Robustness is the ability of the swarm intelligence system to perform tasks well even in the presence of environmental disturbances. Systems that display this tendency tend to share numerous traits in common, including decentralization due to the absence of central decision mechanisms; multiplicity of sensing, which induces an increase in the overall signal to-noise ratio; and redundancy, in which many individuals repeat the same behaviors continuously for the same functionality. Incapacitation of one or a few systems is minimized, enabling tasks to be continued.

Flexibility is the ability of the system to respond to changes in the environment. For instance, ant colonies change the path to a food source after finding one path that is the shorter than the one used previously [87].

Easy implementation is the ability easily communicating to a broad audience of potential users. In easy implementation, there is no need for a heavy math or statistical background. The tuning parameters are few and easy to understand and adjust. In some cases, this can be a part of the optimization options of a larger project. In addition, low maintenance costs due to built-in adaptability in response to changing operating conditions result in low total-cost-of-ownership. To obtain these properties in artificial systems, the design of the interactions among system components is important.

Swarm intelligence systems have many attractive characteristics, but there are some algorithmic drawbacks. The first drawback is that these systems usually suffer from premature convergence when problems with multiple optima are being optimized. There is no guarantee that the solution found will be a global optimum. The second drawback is that the performance of these systems is very sensitive to parameter settings. Tuning the proper

inertia is not an easy task and is problem-dependent. Beyond these specific technical issues, swarm intelligence systems also lack a solid mathematical foundation for analysis (especially for realistic algorithm convergence conditions) and a generic methodology for parameter tuning. In addition, there are a few potentially serious issues, such as the nature of predictability in distributed bottom-up approaches, the efficiency of the emergent behavior, and the dissipative nature of self-organization.

2.2 Swarm robotics

Swarm robotics is the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment [189]. Swarm robotics involves a large number of autonomous robots. Each individual robot lacks global information, and there is no assistance via centralized control. Individual rely on locally obtained sensing and communication among robots, and the desired collective behavior is obtained through self-organization based on robot-to-robot and robot-environment interactions. Figure 2.1 shows two general taxonomies according to what collective behaviors are studied and according to what design or analysis methods are used for swarm intelligence systems. Some collective behaviors studied in swarm robotics are reviewed in the following sections.

2.2.1 Description of collective behaviors

Collective behaviors are studied using swarm robotics to solve many complex problems. They can be divided into three main categories [26]; spatially-organizing behaviors, which organize and distribute robots and objects in space; navigation behaviors, which organize and coordinate the movement of the swarm; and collective decision-making and task allocation, which allow the swarm to agree on a common decision or divide the swarm into two or more groups based on their individual decisions. A few representative examples for each category aggregation, coordinated motion, pattern formation, collective exploration, and collective decision-making/task allocation are reviewed below.

Aggregation is a behavior inspired by behavior models of animal (such as bees, fish, cockroaches, penguins, even in bacteria [31]) that allows all robots to meet at the same location. It is the simplest collective behavior and a necessary precondition for other col-

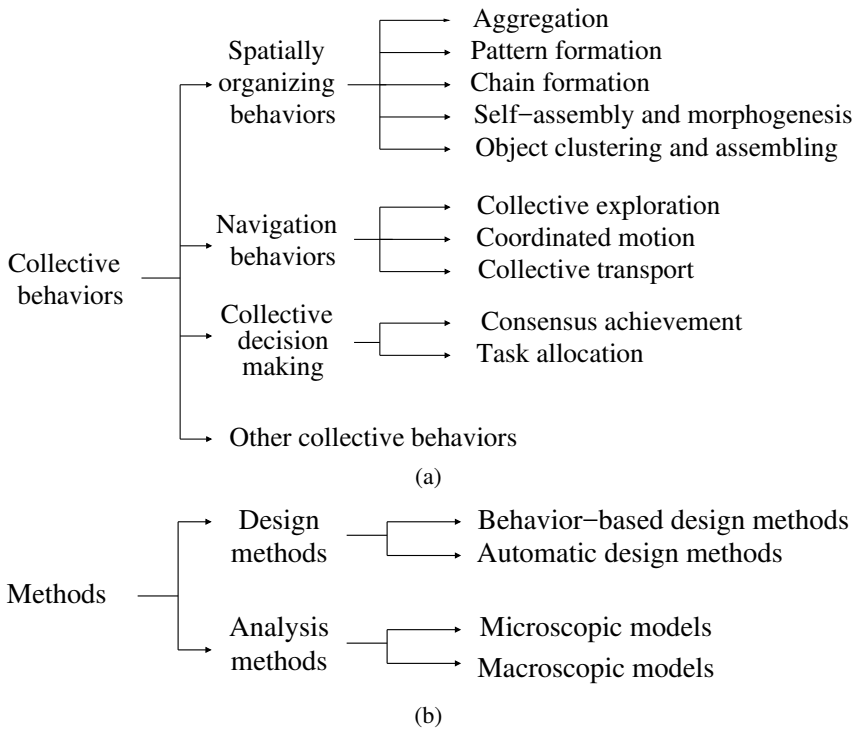


Figure 2.1: General two taxonomies for swarm intelligence: (a) type of collective behaviors; and (b) design and analysis methods (Modified from [26]).

lective behaviors such as coordinated motion and pattern formation. This is mainly implemented by means of probabilistic finite-state machines. The most challenging problem is aggregation at a random location when no information, cues, or landmarks are available. In these cases, artificial evolution [168] or probabilistic finite-state machines [158] are used to decide where to aggregate.

Coordinated motion can also be observed in nature. Groups of up to thousands of animals such as shoals of fish [86] or flocks of birds [167] move together like a single creature [44]. Basically, a swarm of underwater or flying robots moving together can increase energy efficiency. Additionally, coordinating the movement of multiple robots can lead to improved performance compared to a single robot. Doing so could increase the sensing range and improve performance in data acquisition or search and rescue task. This could be a critical factor in determining success or failure in dangerous applications such as military missions.

Pattern formation is used to organize robots in a regular and repetitive pattern. Pattern formation is inspired by biological processes such as crystal formation [117] or the formation of chromatic patterns in animals [156]. This behavior is necessary for coordinated motion, and it commonly uses a virtual physics-based design.

Collective exploration, inspired by the behavior of social insects such as bees and ants, is used to explore an environment efficiently and to find resources, Robots form an interconnected static or dynamic network covering areas of interest in an environment, and this is implemented by means of communication methods, and finite-state machines [57].

Collective decision making and task allocation concerns the ability of robots to make choices. The choices of individuals influence the choices of other individuals, and two types of decision making are available: consensus to the same decision, as inspired by cockroaches [4]; and task allocation to two or multiple tasks, inspired by social insects such as bees and ants [31]. Probabilistic finite-state machines [67] and statistical-physics [161] are used for consensus decision-making [33], while the probabilistic finite-state machines [142, 28], are mostly used for task allocation. Task allocation is the main collective behavior considered in this dissertation and more details are explained in Section 2.3.

2.2.2 Design methods

The implementation of collective behaviors in swarm robotics still suffers from a lacks of rigorous design methods. After designing a desired macroscopic objective, the derivation of microscopic behaviors for individual robots and of interaction rules among robots is not obvious. As categorized in [26], two main design methods for swarm robotics are summarized: behavior-based design methods and automatic design methods.

The behavior-based design method is an iterative process. Microscopic behavior is implemented and modified until the desired goal is obtained. One behavior-based design method uses finite-state machines [158], which can be probabilistic or deterministic. For behavioral models of social insects such as ants or cockroaches, the state transitions are controlled by probabilities. In other cases, such as morphogenesis [170], the state transitions are not controlled by probabilities. Virtual-physics based design is another behavior-based design method. Each robot is assumed to be a virtual particle governed by the virtual influential forces of other robots or the environment. In swarm robotics, virtual forces can be used to coordinate robots into desired formations [204] or to guide them robots toward a target destination [107, 184, 127].

In automatic design methods, microscopic behaviors and interaction rules are automatically generated by maximizing the performance evaluation function of the overall system. Two design approaches, reinforcement learning [96] and evolutionary computation algorithms [73] are mainly used. To apply reinforcement learning, robots are required to modify their behaviors from positive and negative feedbacks through trial-and-error interactions. However, performance is typically measured by each robot's individual sensors, and it is hard to evaluate the overall performance of each interaction. Evaluating performance at the swarm level is important to obtain desired levels of performance [26], this limits the direct applicability of reinforcement learning method to swarm robotics. Only a few studies have been performed on swarm robotics [152, 173]. In evolutionary robotics, an evolutionary computation algorithm [73] is used to optimize the fitness function. Both swarm performance and individually evaluated performance can be used in evolutionary robotics [220], and more studies have been done on this method [13, 165, 6, 205]. However, evolutionary robotics typically uses artificial neural networks [17], and these are difficult to understand and reverse engineer.

2.2.3 Analysis methods

Modeling is useful to predict whether the designed methods will be effective for implementing the desired collective behaviors. However, a unique theoretical method for modeling swarm robotics does not yet exist. Two different levels of modeling is available: microscopic and macroscopic. On the one hand, microscopic modeling focuses on the individual robot and on interactions among individual robots. It is difficult to apply mathematical methods, so microscopic modeling is performed using numerical simulations.

On the other hand, macroscopic modeling focuses on the swarm as a single system. Deterministic and stochastic modeling methods can also be used. Deterministic modeling is typically based on control theory, which assumes that each robot is connected via either static or dynamic graph topology based on interactions. It is used to prove convergence, i.e., whether the swarm will eventually (i.e., asymptotically) reach the desired macroscopic state, such as aggregation [68], foraging [143, 126], and task allocation[89, 134]. Control rules to achieve the desired properties can be also derived with some assumptions [89]. Other deterministic model is based on ordinary differential and difference equations [161, 172] and on partial differential equations [19]. Stochastic modelings are also based on stochastic difference or differential equations. One of the stochastic models used to define a

macroscopic model in swarm robotics is based on rate and master equations [150]. Similar approaches have been applied to many behaviors, including foraging [137], clustering [150], and stick-pulling [149].

2.3 Task allocation problem

Social insects perform a multitude of tasks such as feeding the brood, nest maintenance, defense, and foraging, and divide this labor among hundreds, thousands, or even millions of workers. Similarly, engineers envision swarms of autonomous agents jointly performing complex and multiple tasks. In order to accomplish this, the following questions must be answered: What factors enable a multi-agent system to display division of labor? What task allocation mechanisms do social insects apply, and what mechanism should robotic swarms use? Similarly, coordination of numerous robots operating in noisy and varying environments usually rules out centralized control algorithms. For these reasons, decentralized and self-organized task allocation methods in biological and engineered societies have received increased attention in the recent years. This dissertation studies design methods for task allocation problems concerning individual task selection in multi-agent systems, focusing on the following three questions:

1. What kind of decision rules does an individual agent follow in performing tasks? One way is to find phenomenological rules that describe the behavior patterns of individuals.

2. How does the individual obtain external information about task needs? Each agent should obtain information from local stimulus within its sensing range, through interactions with other workers, or through both. Task selection of an individual agent is a response to the estimated environmental information.

3. What internal mechanisms control the behavioral rules for task selection? Both genetic factors and the effects of interactions with other workers should be considered as important factors.

Individual agents can change their preference tasks while performing a task depending on how much they have specialized in one or a few tasks and on changes in task needs. Repeated task selection changes colony-level patterns, including the sizes of task groups and distribution of task allocation. The methods we want to discuss attempt to explain task allocation patterns from individual workers to the colony level.

2.3.1 Task allocation in nature

Task allocation is one of challenging subjects in multi-agent systems. This subject has been studied in both biology and swarm robotics. In nature, an individual agent generally has very limited perception and knowledge of the environment, and also there is no centralized control that guides the cooperation behavior among a swarm of agents. Despite the limited abilities, a swarm of agents in nature show an effective task allocation by interacting with each other and sometimes perform a high level of tasks beyond the capability of a single agent. This concept can achieve effective task allocations based on the specialization tendency, i.e., different tasks are simultaneously performed at different places by specialized individuals [187]. Examples of labor division include nest defense and foraging in ants [51] and nursing [29] and nectar and pollen collection [218] in honeybees.

It is well known that several social insects use task partitioning for a colony's survival [186, 197, 64, 191]. A single individual performs one of multiple tasks. While this behavior might have a very limited effect on the colony-level, interactions among individuals in a group can lead to an efficient self-organized system for regulation of work [197, 191]. The colony-level flexibility responding to the dynamically changing environment is an essential feature and task partitioning as an adaptive strategy plays a great role for the colony.

A typical example of task partitioning strategies can be found in the honeybee *Apis mellifera*. For the foraging task, the honeybee is partitioned into two groups; one group works on a resource-collecting task and the other is assigned to a resource-storing task. The former group of honeybees moves around to find nectar, picks it up, returns to their nest and passes it to the latter group. Then the latter group of honeybees stores the nectar in a cell of their hive. It is also known that honeybees use this kind of task partitioning strategy when they collect other materials such as propolis, water or pollen [197, 64, 191]. Another example is found in the termite *Hodotermes mossambicus*. There are two groups of termites cooperating to perform foraging effectively. The one group cuts grass and leaves into pieces, and drops them to the ground. Then, the other group collects the fallen leaves and takes them to their nest [139, 7]. Recently, it is reported that simple regulatory mechanisms based on a common stomach can lead to the optimized regulation of protein forging and protein allocation in honeybee colonies with responses to environmental changes in colony-level [193].

Usually, in insect societies, task partitioning is often involved with two genetic factors; age-dependent (temporal polyethism) and different body shape (worker polymorphism) [225, 208, 91]. For worker polymorphism, workers within a colony have morphological

differences. For example, there is a large amount of body size variation in bumble bee workers. The largest workers may be ten times than the smallest workers and the size is correlated with task types; larger workers tend to forage, while smaller workers tend to perform brood care and nest thermoregulation [43]. Temporal polyethism can be observed in honey bee colonies. There is a correlation between the age of workers and tasks they perform. Younger workers perform brood care and nest maintenance inside the hive and older workers forage for food and perform defense tasks outside the hive [223, 225].

Individuals can be specialized in two types, a strong or soft manner. Genetic factors certainly play a role in some types of strong specialization, such as age-dependent specialization and different body shapes. Strongly specialized individual performs only one or few activities. However, unpredictable real-time events of the environment and the variability within the colony require additional mechanisms for ensuring dynamic task allocation. So, the softly specialized individual performs several activities, but tends to perform the activity that is most needed by the group at every instant.

However, this tendency can be changed flexibly by the distribution of age in colony members. For example, in a colony that the proportion of young honey bees is high, the age in which a bee starts a foraging task is lower than in a normal colony and the presence of older bees delays or inhibits the development of physiological age of other younger bees in the colony. It is shown that worker interactions among bees drive mechanisms of hormonal regulation resulting in a social inhibition [91]. This fact is used in other studies of task allocation based on social inhibition [20, 21].

Task allocation can also be considered as labor division, task assignment, or task partitioning. The work involves decomposing a task into a sequence of several subtasks to allow a group of agents to perform each of the subtasks in parallel at different locations [186]. Depending on the needs of the swarm system, this subject involves with decomposing a task into sequentially interdependent subtasks and allocating a group of agents to perform different subtasks in parallel. The posterior subtasks should be processed after completing the prior subtasks in order to achieve the overall task. To increase the overall performance in a swarm level, a task partitioning method is needed for balancing the task demands of subtasks by regulating adaptively the number of agents assigned to each subtask.

Such subtasks are common in natural and artificial systems. Many kinds of sequential subtasks can be observed in social insects and there are many examples of division of labor based on task partitioning [187, 183, 182, 62, 3, 230]. The tendency for division of labor is found in the colonies adaptable to dynamically changing environments. Various task parti-

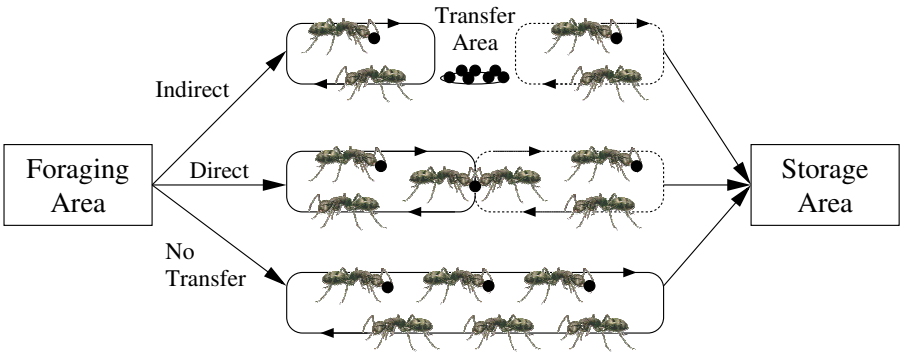


Figure 2.2: Various ways of task partitioning for performing a foraging task with a group; indirect and direct object transfer methods are shown with no task partitioning method. Top shows the two subtasks with indirect transfer using a cache site. Middle shows the two subtasks that object are transferred directly between different groups. Bottom shows that tasks are performed without task partitioning.

tioning strategies are employed to handle their tasks required for a colony's survival. The tasks such as garbage disposal and forage are performed based on just a few basic behavioral rules. Depending on the current need in the colony-level, the individual worker performs one specific task among multiple candidate tasks. Tasks are simultaneously performed at different locations. This tendency usually has an advantage to manage the division of labor effectively. Their self-organization and self-regulation strategies may provide a hint of solving task allocation in multi-agent systems.

Many social insects handle complex tasks consisting of sequential subtasks. Various forms of task partitioning strategies observed in nature are shown in Fig. 2.2. In a direct transfer method, an item is transferred directly between workers for transferring liquid such as nectar and water in a colony of social insects. *Atta cephalotes* (for leaves), *Messor* (for seeds), and *Polybia* (for wood pulp to build their nest) all use the direct transfer to pass their materials to other workers [183, 185, 145, 7, 188]. It can be also used for the exchange of regurgitated food between adults and larvae.

The direct transfer method is also used for exchange of regurgitated food between adults and larvae and for transferring liquid such as nectar and water. A typical example of direct transfer method can be found in the honeybee, *Apis mellifera*, that is partitioned into two groups for the foraging task. One group performs collecting resource and the other performs storing food. The former group of honeybees moves around until they find nectar and

returns to their nest and passes it to the latter group of agents. Then the latter group of honeybees stores the nectar into their hive. This kind of direct transfer method among two groups is also observed when they collect other materials such as propolis, water or pollen [197, 64, 191, 3]. This kind of task partitioning that the material is exchanged from the collectors to transporters is bucket brigade (BBs).

In contrast, task partitioning using an indirect transfer method can also be observed in nature. An indirect transfer in foraging can also be observed in nature. There is no physical contact between individuals. Each worker delivers goods to or takes goods from a fixed or temporally changing transfer location. There is no physical contact between workers. Workers deliver goods to or take goods from a cache location. The cache is a temporally changing location or fixed chamber with a pile of materials such as food or garbage which should be transferred into the nest or out of the nest. The location of a cache can be either fixed or unfixed, and the unfixed cache can be produced somewhere on the way between the source and destination of the materials.

An example of a transfer area is a chamber with a pile of materials such as food or garbage which should be transferred into the nest or out of the nest. This transfer location can be located somewhere on the way between the source of materials and the nest. An example with an indirect transfer method can be found in the termite, *Hodotermes mossambicus*. One group cuts grass and leaves into pieces, and drops them to the ground. Then the other group collects the fallen leaves and takes them to their nest [139, 7]. This strategy can also be found in waste disposal processing. They make a garbage heap in their nest and this area is used as a cache for transfer. Some workers bring a piece of garbage and drop it near the garbage heap. Then another workers in the heap area throw the garbage out of the nest [83].

There have been many studies to explore the indirect transfer method [100, 81, 101, 3]. Using the common stomach as an information center, *Metapolybia* wasps regulate their partitioned work for building the nest [104, 101]. The colonies maintain the interactions among workers to ensure the steady construction of the nest. With the common stomach, highly requirable tasks are performed and those in low demand are ignored [102].

Atta cephalotes, known as leaf-cutting ants, use various task partitioning strategy to handle their materials such as food and garbage [83]. When an ant finds a food source, it selects one of two strategies. The first strategy is to just return to the nest, carrying food by itself; this means that the ant has no task partitioning strategy. In the second strategy, the forager ant directly passes the food he is carrying to another or drops it in the middle of

the pheromone trail so that other moving ants can bring it to the nest. These strategies are also applied to waste disposal processing. They make a garbage heap in their nest and this area is used as a cache for transfer. Workers belonging to a part of the fungus farm bring a piece of garbage and drop it near the garbage heap. Then, another worker in the heap area throws the garbage out of the nest. This cache prevents the spreading of parasites and disease from the garbage and segregates contaminated areas from clean areas. Direct and indirect transfer can actually happen in the same species is modeled [192]. They explained task partitioning in ant that individual ants can adjust their foraging behavior by engaging in or by abandoning from stinging or transporting due to a common stomach system.

The above described process of task partitioning can be divided into task partitioning with a single transfer or with multiple transfers. The typical example of single transfer task partitioning is a nectar transfer with honeybees. Collector bees take honey from the nectar source and hand it over directly to the resource-storing honeybees. No more additional transfers are needed in this process. Task partitioning with multiple stages is more complex than a single transfer. An example of multiple transfers is found in the ant *Atta sexdens* for leaf fragment transfers [66]. There are three types of tasks among ants: arboreal cutters, cache exploiters, and carriers. Arboreal cutters move up a tree to cut leaves and drop all of the leaf fragments to the ground. The pile of fallen leaves acts as a cache for task partitioning. Cache exploiters then find the pile caches, cut the leaves into pieces and take them to the foraging trail. Finally, carrier ants take the pieces and transport them to the nest. Similarly, the foraging behavior of *Atta cephalotes* is another good example of multiple-stage task partitioning. They use both indirect and direct transfer with multiple stages [66].

2.3.2 Mechanisms of task allocation

The overall pattern of the division of labor often consists of multiple components, each stemming from a different source. Consequently, the observed level of specialization is caused by a mix of environmental, genetic, learning, and social factors. We shall now describe them shortly, together with support from empirical evidence from biology and engineering. The factors known to affect the decision of an individual agent to perform a given task are summarized in Figure. 2.3. The external factors outside the individual are the stimulus based on task needs and the interactions among agents within a limited range. The internal factors within the individual are the response threshold, which determines the response to perform a specific task and is basically based on genetic factors and the results

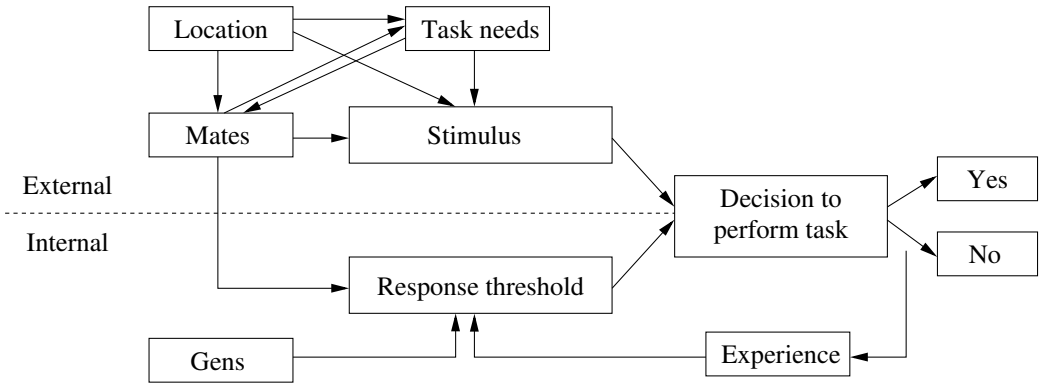


Figure 2.3: Components known to affect the decision to perform a task.

of experience. Performance of a task increases an individual's intrinsic probability of performing that task again. Interactions among agents may also affect an individual's internal state. External factors can affect internal factors. Performance of a task affects not only the stimulus perceived by other agents but also the individual's preference with regard to that task. Task allocation is ultimately the result of both the internal state of the individual and the external state as determined by interactions with the environment.

Many different kinds of adaptive behaviors in nature can been observed, and it is known that several insect societies show an adaptive task allocation based on specialization characteristics; each agent has performing tendencies for all tasks and each task is performed by specialized individuals that have higher tendency than others and perform task priorly and frequently [196, 75, 65, 215, 203].

To explain the concept of specialization, one study performed biological experiments about the relation between caste ratios and division of labor in the ant [224]. Generally, workers are divided into two factions. The small minors fulfill most of the quotidian tasks such as cleaning or brood care, and the larger majors are responsible for seed milling, abdominal food storage, and defense. By reducing the proportion of majors to minors, majors started to participate in the tasks usually performed by minors.

This dynamic task allocation can be explained based on stimulus-threshold relation by assuming that specialists of Type A have low thresholds for the stimulus associated with Task A and high thresholds for Task B, whereas specialists of Type B have low thresholds for the stimulus associated with Task B and high thresholds for Task A. As soon as the number of specialists of Type B decreases, the stimulus associated with that task will increase until

it exceeds the corresponding threshold for Type A specialists. Consequently, some Type A specialists will perform Task B until the corresponding stimulus falls below their threshold. This tendency allows their colonies to adapt to a dynamically changing environment for colony survival. In most cases, cooperating among specialized agents can obtain better performance in completing a task than a single agent and adaptive division of labors for global behaviors can be obtained by simple rules.

Individuals can have a specialized tendency as a strong or soft type. The member with the strong specialization performs only one or a few activities among all of the tasks required to be completed for the colony survival. However, unpredictable changes in the environment and the fluctuating proportion of members result in a supplementary mechanism within a group for maintaining an adaptive division of labor task allocation. Therefore, the members with the soft specialization are required to perform several activities. They usually perform the urgent task required by the group at each moment. For example, if the number of specialists of Task A decreases, the demand for Task A increases, and accordingly, some specialists of Task B perform Task A, which they would normally not do [224].

In order to explain mechanisms of division of labor in social insects, several models have been proposed [214, 24, 55]. Among many models, we focus on the response threshold model [23, 212]. The response threshold model used in the insect-inspired model approach is composed of four components, stimulus of task, response threshold to task, task probability function, and threshold updating algorithm. The probability for performing each task is decided according to stimulus of task and threshold value for task, and the threshold values are regulated based on the current performing task to have a specialized tendency. Task demand decreases as the individual performs the task and individuals with higher thresholds are unlikely to perform the task because individuals with lower thresholds most probably handle the demand. This model is based on observations of the collective behavior in a colony of insects, particularly the work performed by bees and ants, and a simple mathematical model using response thresholds for the regulation of division of labor was proposed [224, 25, 212].

An individual agent responds to perform a task based on the task intensity and the corresponding response threshold value that determines the tendency of an individual to respond to perform task. Different responses to the same task intensity are generated according to the different response threshold and this characteristic determines the tendency of an individual agent to perform tasks or not. Due to their apparent simplicity, response threshold models have been extensively used by engineers as task allocation algorithms for groups of artificial

agents. This model can explain various task allocation phenomena in colonies; foraging and nest defense in ants [51], foraging and nursing [29], and nectar and pollen collection in honeybees [218]. These characteristics have been a source of inspiration for many researchers in application areas and various works are done base on this model [147, 38, 110, 98].

Task allocations based on the response thresholds have been extensively studied since their divisions of labor are realized by simple rules. In this model, obtaining an appropriate distribution of thresholds are important to obtain a desired division of labor. Each individual responds a task only if the external stimuli denoting the need for its performance exceeds its internal threshold for the task. With this respect there are, however, two issues. First, there is a question of how to find appropriate values of the thresholds. Usually, the values of thresholds are fixed and differ between the workers, thus reflecting the genetic, morphological or age variation [212]. However, the randomized thresholds may impair the overall performance [130].

To overcome this problem, an improved version that dynamically adapts thresholds has been studied [41], where a computational model of how wasp colonies coordinate individual activities and allocates tasks is studied. This model regulates thresholds based on the self-reinforce-learning model [210] to obtain optimal thresholds. The response thresholds change due to individual experience. After performance, threshold is updated by decreasing threshold of performing task and increasing threshold of other tasks not performed. Based on these repeated behaviors, each agent comes to have a specialization tendency for a specific task and this tendency induces more effective division of labor in a group. The response threshold model is optimized with learning algorithms [115], or by means of evolutionary simulations [34] and artificial neural networks [141]. This model has been successfully applied to explain colony reaction to perturbations [219] and division of labor patterns observed in honeybees [16], and ants [23]. The diagram of task allocation explained with the fixed and variable thresholds values are shown in Figure 2.4. The lower the threshold, the higher will be the response of an individual for performing task for a given same task intensity.

The second problem is that response threshold models originally assumed that tasks stimuli are commonly available to all members in a group, which is possible only when the needs of the society may be efficiently communicated and aggregated by the members. This is the case of multi-agent systems with adequate communication capabilities, like teams of robots with a central unit overseeing their behavior [114, 19]. Also, response thresholds have been used to solve and load balancing [35, 190] problems in computational clusters

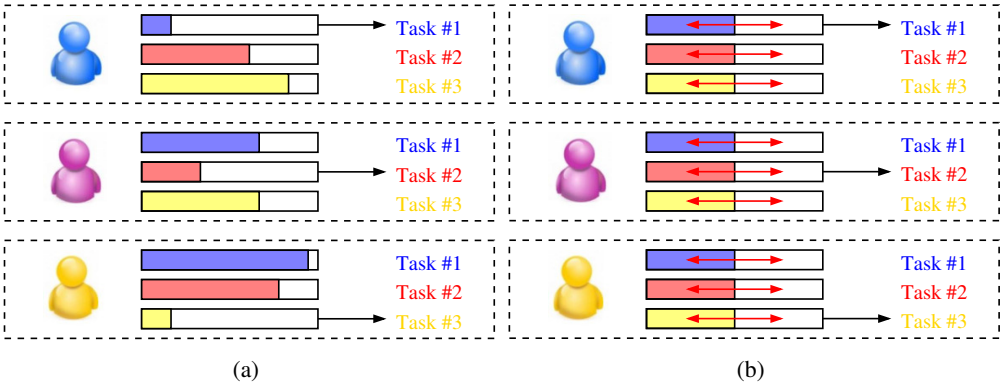


Figure 2.4: Diagram of task allocation explained with response threshold model. The control of task allocation can be explained with (a) fixed threshold values or (b) variable threshold values.

where communication between computer processes is relatively easy. However, global task needs may sometimes be estimated from locally accessible information [119, 130, 2, 151].

To address the issue of global stimuli, message passing between neighboring agents is an often used technique in task allocation in robotic systems. The most well-known methods are market-based algorithms [69, 116, 216], where agents compare their preference to perform a task and the most eager agent performs it. The market-based algorithms are similar to the ones using response thresholds, in the sense that with the former approach agents compare their will to perform the task with each other, whereas in latter approach they compare it with their internal thresholds.

2.3.3 Applications to task allocation

Swarm intelligence and collective robotics have been inspired by adaptive behaviors observed in nature [22, 18, 232, 166, 49, 155]. Swarm robotic system is primarily inspired by social insects. It is composed of homogeneous or heterogeneous agents and various tasks are performed at different places at the same time. In many robotic systems, no communication or only local communication is available and the knowledge of the environmental information is also limited. There is no centralized mechanism and the individual robot has an equal capability for completing each task. In the swarm robotics applications, there has been an issue of how the agents in a swarm are allocated to several subtasks in order to

maximize the overall system performance.

A lot of works have focused on determining whether an agent performs a given task or not and maximizing the colony-level performance [179, 28]. Several effective and adaptive behaviors have been identified in recent studies: task allocation [135, 221, 94], searching [36, 90, 60, 131] and foraging [114, 137, 180]. Traditionally, task allocation has used a centralized mechanism [27, 174, 53, 169]. However, depending on tasks, it may not be possible that a controller reads all the needed information, such as positions of agents and the current tasks assigned to all agents, and then allocates its task to each agent. Yet it requires not only obtaining the desired performance but also the robustness and fault tolerance. Hence, the distributed task assignment approach with no centralized control has received attention recently.

In swarm robotics, many studies show how the individuals in a swarm are allocated to the subtasks to maximize the overall system performance. Agents can perform different tasks and specialization characteristics increase the performance in division of labor. Task partitioning can be used as a means of reducing the physical interference in a group of robots [195, 171, 177, 48, 178]. In addition, self-organized task allocation using the response threshold model [24] has been employed in foraging [140] and object clustering tasks [2]. There have been studies of task partitioning in swarm robotics to handle the transfer of objects among homogeneous robots [178, 181, 28, 63].

The foraging task has been used as a test in a multi-robot system. Foraging task is one of popular subjects to demonstrate task partitioning in multi-agent systems. A group of robots are assigned to search for desired items and deliver them from a resource area to specific locations. In a simple forging task, a group of robots must collect objects of a single type, usually for energy balancing or a similar objective. The impact of simple communication on the performance of a society of robots in a single-prey foraging task was studied [10]. They used the minimal knowledge of the behavioral state of the fellow agents. A behavior-based algorithm using reinforcement learning was proposed to induce the robots to learn how to balance each of the behaviors. The objective was to optimize the performance in a foraging task [151], using an energy level to maintain the energy stocked in their nest by collecting a food item [113]. A similar task allocation problem motivated by energy gain has also been previously studied [115]. The foraging taxonomies [231] and multi-robot coordination problems and solutions [226, 127] have been studied.

In typical foraging tasks, an individual robot or robots in a group collect objects from an environment and immediately remove the objects from that spot [92] or transfer them to

another common location such as a nest or transfer cache [72, 93]. There are many examples of such foraging tasks using multi-robot systems and various task allocation methods have been proposed to maximize the performance [28, 38, 229]. Several variables could be considered for the evaluation of the performance in a foraging task. The performances can be evaluated based on the total time required to complete a given task, energy consumption, actual amount of task allocations, number of foraged objects and number of task changes. Moreover, if costs are incurred due to the switching of the current task to another, the minimization of the task switching is beneficial to the overall system performance. Simulation-based experiments were used to demonstrate the robustness and adaptability of the proposed approach to environmental variations, and the division of labor is demonstrated with minimum task switching to obtain the specialization for specific tasks. In this paper, a dynamic task allocation problem similar to that in Lerman et al. (2006) , which handles the foraging problem with two types of objects according to their colors is considered.

Most aligned with this work is Jones and Mataric's work (2003) , where the division of labor is achieved in a distributed manner using mathematical modeling. They suggest a ratio model for the division of labor. In this model, two types of objects are used (e.g., pucks), and each robot is equally capable of foraging for both puck types, but can only be allocated to forage for a single type at a time. According to the control policy, a robot may switch the foraged puck type from red to green depending on whether the ratio of red puck entries in the view history is smaller than the ratio of robot entries foraging for red pucks, and similarly, a robot may switch the foraged puck type from green to red depending on whether the ratio of green puck entries in the view history is smaller than the ratio of robot entries foraging for green pucks. The ratio of robot entries and that of puck entries that the robot has sensed are used to determine the foraging state of the robot.

In a self-organized task allocation in swarm intelligence, stochastic decision based on the response threshold model assigns given tasks to the individual robots in a swarm. While some studies use fixed thresholds for responding differently to the same task demand [114], most works employ methods using adaptively changing thresholds to allow flexible task change of each agent according to the dynamic environment. Krieger et al. (2000) implemented a simple and decentralized task allocation mechanism based on the individual activation-thresholds. Each robot is assigned with threshold values and the robot will decide to travel and collect food items if the energy level of the nest is less than the threshold. Labella et al. (2006) presented a task allocation algorithm in a group of robots involved in the object-retrieval task. Each robot leaves the nest with some probability. If its foraging

task is successful, that is, a robot is successful in retrieving a prey, then it increases its probability, If it is not successful, it decreases its probability. Division of labor in a group level is autonomously progressed with this algorithm.

Yang et al. (2009) proposes a classical foraging scenario in order to test the response threshold model. In that scenario, robots decide whether they go foraging with a certain probability in order to adjust the number of working robots depending on the amount of food left at the nest, which will lead to the division of labor. Castello et al. (2013) proposed an adaptive response threshold model in the task allocation algorithm for a robotic swarm. They adjusted the number of working robots efficiently according to the amount of food left at the nest. They decreased the response threshold values if a relatively small amount of food is placed within the home zone and increased the response threshold values if more food is observed in the nest zone. Once threshold is calculated, robots decide whether to go foraging or stay at home based on the probability.

Kalra and Martinoli (2006) compare an auction-based approach with the threshold-based approach. If the information about the task is available to the individuals in a group, the auction-based approach performs better than the threshold-based method. If the information is not accurate, the threshold-based approach could perform as well as the auction-based approach.

Other interesting studies have focused on handling task transition rates for task allocation in a swarm of robots [79, 89, 19], and show that an appropriate selection of task transition rates can guide the desired task distribution. In addition, the macroscopic analytical model that describes the dynamics of the task transition was studied [153].

A few works tackle the problem of task allocation for sequentially interdependent subtasks. There have been traditional task partitioning approaches focusing on reducing interference. Pini et al. (2013) proposes an autonomous task partitioning in a swarm of foraging robots by using a cost function, conceptually similar to the problem presented in this article. The robots autonomously calculate the traveled distance and deposits the objects at an appropriate site. Then the robots performing random walk can find objects and transfer them to the nest. The object transportation is executed as a sequence of subtasks performed by multiple robots and task partitioning improves the performance.

Brutschy et al. (2014) presents a self-organized method for allocating the individuals of a robot swarm to tasks that are sequentially interdependent. Objects are transferred directly between robots and the task change in the method is based on the interface delay (waiting time) experienced by the robots in one subtask relative to the interface delays experienced

in the other subtask. Elsayed and Al-Wahedi (2015) investigates task allocation for partially sequential tasks in a two-robot environment. Two robots are non-identical. The first has an ability to decide the optimal order of performing subtasks, while the other just follows a set of sample behaviors. Zahadat and Schmickl (2016) presents an adaptable partitioning of autonomous underwater robots based on social inhibition. An agent selects its own task based on the relative demand for a set of multiple tasks and the response threshold values regulated by local interactions.

In real manufacturing systems, the machine environments are changed dynamically and unpredictably due to various real-time events, such as machine breakdowns, changes in the scheduled product plans, or the arrival of high-priority tasks. Even under such circumstances, the overall systems that are composed of multiple machines should adapt to their dynamic environment and maintain their expected performance. Therefore, dynamic scheduling algorithm [209] handling the problem of scheduling in the presence of real-time events, is important in many real-world applications, and many various methods have been studied to solve these kind of dynamic scheduling problems [12, 5, 228, 207, 233, 1, 201, 47].

Bio-inspired approaches based on the response threshold model have also been applied to others applications. Among various tasks, truck painting problem is the basic and simple but appropriate task that the specialized tendency is most needed for improving performance due to the extra costs for task changes. The objective of the dynamic scheduling algorithm is to minimize the number of task changes of the overall system while it maintains the desired performance level such as throughput; the number of tasks completed during a given time span. Achieving optimal performance is a non-trivial problem because of environmental unknowns such as the colors of future trucks as well as other uncertain events such as paint booth failures.

A simple bidding mechanism based on market-based approach was first proposed: each paint booth submits a bid for trucks depending on its current queue length and the color of the last truck in its queue [162, 163]. The bids of all agents are compared, and the task is assigned to the highest bidder. This simple multi-agent bidding system is 10% more efficient than the previously used centralized scheduler in terms of the total number of color changes in all the booths. An improved method inspired by the ant behavior model was proposed [34]. A truck painting booth is represented as an agent that autonomously competes to paint a truck. Each booth has a threshold value for each color and the probability that booth with a specific color will get a truck is obtained by the stimulus of truck and the threshold of

agent for that color. Obtaining a proper threshold is directly related to the overall system performance in the response threshold model, and the threshold values are changed depending on the assignment of tasks. Method inspired by the wasp behavior model also studied [41]. The threshold values are updated at each time depending on the currently performing tasks. Repeated task selecting behaviors result in agent specialization for particular tasks by lower the threshold values of tasks, and this tendency induces the less number of task changes than market-based approach.

2.4 Summary of Chapter 2

In this chapter, we introduce different background ideas. Our basic interest is based on the behavior of animals that show adaptive task allocation for improving colony survival in a dynamic environment and there are many detail analysis of this behavior in literature. These behaviors can be explained using stimulus-threshold relations. Various applications with multi-agent systems are also introduced.

Chapter 3

Basic for task allocation

We want to present an agent-based task allocation method using response threshold model inspired by insect societies. This model has the effect of reducing the task switches by inducing the division of labor due to its specialized tendency and it has been studied for the purpose of task allocation in a group of robots. First, a fixed response threshold model using a randomly-generated threshold value was used for a foraging mission [229]. Later, the adaptive response threshold model in which the threshold value is adaptively changed depending on the environmental circumstances has been suggested [38]. Robots decided whether they would go foraging or remain in a standby mode near the base area was performed.

The foraging task used for our tests involves exploration for desired objects, and the task switching of the robots seems inevitable in this type of task. We applied for improving the performance of a foraging task. We suggest a method that includes the use of task history to estimate the global task demand. Each robot estimates the desired task demand using the moving average of the observed tasks in the course of time. The robots are characterized by their own response threshold values. The threshold values are uniformly distributed among a set of robots to allow for the diversity of the response characteristics.

The division of labor generally uses a global task demand as an important variable. Based on the various experiments, an appropriate choice that uses the task history of the recently-observed information can be used as a guide in a multi-robot system. Interestingly, local information regarding the surrounding environment can be used to estimate the global

task demand well. Each robot becomes to have a specialization tendency, and the overall system thus has the advantage of greatly reducing the occurrence of task changes. If there is an extra cost for task switching, the system based on the response threshold model may greatly improve the energy efficiency by reducing the task switching frequencies.

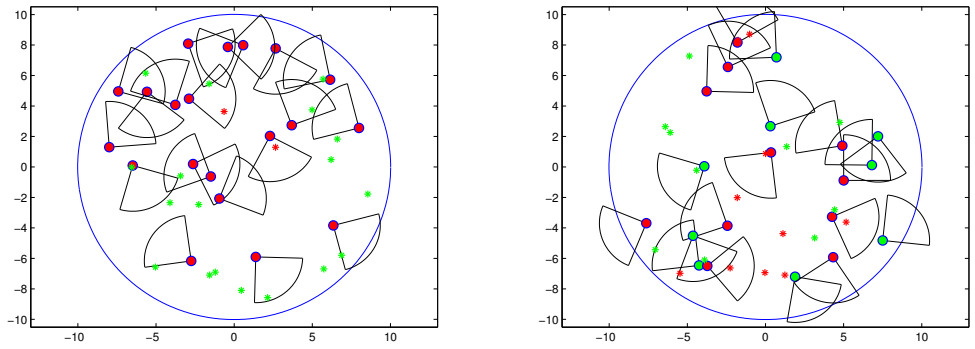
This chapter is published in conference and journal [125, 130].

3.1 Description of foraging task

We consider the problem of dynamically adjusting the population of robots performing specific tasks in proportion to the amount of the tasks. The purpose of this task allocation is to manage division of labor efficiently to reduce the overall task completion time. If all the tasks need the same amount of time to perform, one possible solution is to assign a group of agents, large or small, to each task depending on the demand of the task. In recent works, self-organized task allocation has been studied to handle tasks that are both distributed and sequential [74]. In a multi-foraging task, one or multiple objects are collected in complex environments. A group of robots is used to identify an efficient multi-foraging task, whereby efficiency is defined as a function of the energy wasted while performing tasks [32] or the energy level in a nest [229]. A multi-foraging task was used to demonstrate the use of a mathematical model for a general dynamic task allocation mechanism [138].

In a foraging task, two types of objects are used (e.g., pucks), and each robot can forage for both the puck types: red and green pucks; however, each robot can be simultaneously assigned to only one type of task, i.e., collecting either red or green pucks. Each robot has two response threshold values for the two given tasks, i.e., the foraging red and green pucks, respectively, and each robot performs the task of foraging for a specific puck according to its control policy.

Most aligned with this work is Jones and Mataric's work (2003) . They suggest a ratio model for the division of labor. A robot may switch the foraged puck type from red to green depending on whether the ratio of red puck entries in the view history is smaller than the ratio of robot entries foraging for red pucks, and similarly, a robot may switch the foraged puck type from green to red depending on whether the ratio of green puck entries in the view history is smaller than the ratio of robot entries foraging for green pucks. The ratio of robot entries and that of puck entries that the robot has sensed are used to determine the foraging



(a) (b)

Figure 3.1: Snapshot of simulation environment: (a) snapshot of an initial state that all robots are assigned to clear red-colored objects; and (b) snapshot of a desired state that the proper number of robots are assigned to each task according to its proportion. The large red-colored and green-colored circles are robots and small red-colored and green-colored circles are objects to be collected by robots. The vision range of a robot is fan-shaped as shown above.

state of the robot. According to the multi-robot task allocation taxonomies proposed by Gerkey and Mataric [69], the categorization of this paper is ST-MR-IA: single-task robots (ST), multi-robot tasks (MR), and instantaneous assignment (IA) problem.

3.1.1 Simulation environment

The foraging task is performed in a circular arena as shown in Figure 3.1. The working area of the arena is 315 m^2, and the radius is approximately 10 m. Robots explore the arena to forage objects. The large red and green circles are robots, and the small red and green dots are pucks. In the beginning, the robots and pucks are randomly distributed in the arena. The robots move to forage for the pucks during a given time span. If a robot approaches one puck, the puck is immediately removed from the environment. After a puck is removed, a new puck of the same type or the same color is placed in an arbitrary location in order to maintain a constant number of pucks in the arena, and the robot continues to forage for the remaining pucks. This type of foraging task is considered to be similar to feeding behaviors in nature, if the pucks and robots are assumed to be the prey and predator, respectively.

In the foraging mission, robots may switch their current task according to an individ-

ual control policy when it is appropriate to control the balance or improve the overall performance of the system. Here, a task allocation algorithm for regulating the population of agents in proportion to the number of given tasks is considered. That is, the dynamic task allocation maintains the proportion of robots with the tasks of foraging for red and green pucks, respectively, such that they are equal to the proportion of red and green pucks in the arena. Therefore, if 30% of pucks in the foraging arena are red pucks, 30% of the robots should forage for red pucks. The system performance can be determined from the number of foraged pucks, time spent or energy consumed during the completion of a given task. If some costs are incurred for switching a task, it is also recommended that such task switching be minimized while maintaining the desired division of labor.

The simulation experiments are implemented using MATLAB program based on a realistic model of the ActivMedia Pioneer 3 DX mobile robot (Adept MobileRobots, Amherst, NH, USA). The sensors data for collision avoidance or motor actions for movement are considered with the inclusion of noisy signals to imitate real systems in which noise is experienced. However, the observation and gripping behaviors in real robots may be slightly different from those in simulated robots. They never fail to grip a puck and obtain exact information using their own sensors. In addition, the delivery of the puck is not considered in the simulation. Therefore, the performance of real swarm robots may be slightly different from that of simulated robots in that aspect. However, these limitation are common in simulation and the results may be almost identical with a commonly used simulators, such as that in Gazebo [217] or Swarmanoid [176].

3.1.2 Robot behaviors

During performing task, each robot repeats a set of the same behaviors. The robot behaviors include vision sensing using a camera, avoiding obstacles, wandering, clearing and task switching. Each robot observes the surrounding environment through its own sensors and grips the closest puck within a constant distance. The robot simultaneously detects and avoids obstacles in order to avoid collisions. In addition, the robot changes the foraging target based on the individual control policy. A flowchart of the robot behaviors is shown in Figure 3.2, and the details of the robot behaviors are explained in the following subsections.

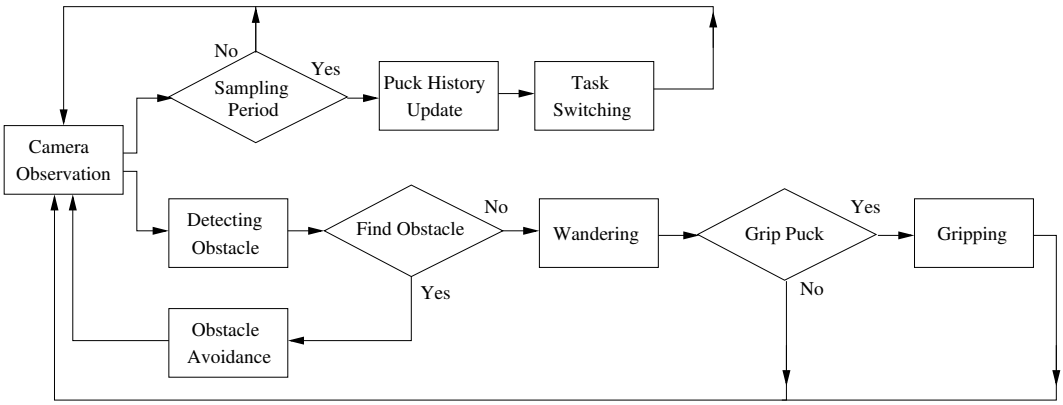


Figure 3.2: Flowchart of robot behaviors.

Vision sensing

Each robot has two types of tasks; collecting red pucks or green pucks, and it can switch between tasks depending on the task demand. The robot is equipped with a camera and captures the scene in front of it at $\pm 90^{\circ}$ within a 2 m range. Using this visual information, the robot observes nearby pucks, and it updates its view history of the pucks.

All of the robots maintain a limited, constant-sized history that stores the most recently observed puck types. It does not contain the identity number or location of a detected puck. Only the last N observed pucks are stored in the puck history and used to estimate the desired global task demand. If there is enough space in the history, all of the information is stored. Otherwise, the oldest data are replaced by the new data. The puck history may be biased based on where the robot is located, and the robot improves the estimation of the task demands by wandering in an arena.

As an observation behavior, the robot stores the puck observation for every 2 m movement. This is an important thing because continuous reading of visual information in a short time can lead to the same information about the environment. This may cause duplicated results in the puck history, which would reduce the calculation accuracy of the global task demand. An exact division of labor may then not be successfully obtained. Thus, robots have periodic shots of vision image for every wandering step. Robots have the maximum forward speed of 0.25 m/s; therefore, the robot camera captures an image for every eight wandering steps.

Obstacle avoidance

Each robot uses eight infrared (IR) sensors to perform the obstacle avoidance behavior. Using IR sensors is effective in the area of autonomous robotics [70, 56, 157]. The wall of a circular arena and other robots are treated as obstacles. The agent tries to avoid obstacles, but small pucks are ignored with no interference with the agent's moving. Each sensor is equipped on the front side of the robot at a uniform distance from each other to cover 180°, and their detecting range is approximately 0.5 m. The robot changes its current moving direction when the sensors detect obstacles.

If the right sensors detect obstacles, the robot turns to the left; conversely, it turns to the right when the left sensors detect obstacles. The robot turns away from the obstacle at an angle of 45° in either case. If the obstacles are simultaneously detected on both parts, the robot changes its moving direction by turning 180° in the counterclockwise direction. However, at times, these constant angle changes may lead to similar patterns in the movement direction and cause the occurrence of a congested group of robots if the robots are gathered in a specific area. To avoid this congestion, a Gaussian random variable is included to obtain a variance in the obstacle avoidance angle.

Obstacle avoidance behavior is a basic behavior that is critical to the safety of the robot. Therefore, the occurrence of collisions with other robots and arena boundaries should be eliminated. However, the puck is not considered as an obstacle because the robot may end up spending considerable time in obstacle avoiding behaviors.

Wandering

The size of the foraging robot is 0.3 m in diameter, and there is no communication between the robots. A robot can move forward and backward at the maximum velocity of 0.25 m/s. Apart from saving information in the puck history, the robot forages for a puck at each time step. The initial movement direction of each robot is randomly set from $-\pi$ to π, and the robot maintains the moving direction until a desired puck is detected. If a robot finds the pucks of the same color as that of the current robot task, the robot then turns toward the closet puck location and moves ahead to grip it. The robot does not perform wandering behavior if the obstacle avoidance behavior or puck gripping is performed. If the robot senses a puck of a type that is different from its own interest type, it ignores the puck.

Clearing

Robots find the closest object within its vicinity among target objects of interest, then move towards the closest object. If the distance between a robot and the object is shorter than $0.3m$ and there is no obstacle near the robot, the object can be immediately cleared at the current location, and object of the same color is placed at random position as a replacement. Robots keep the current moving direction if there is no detected object, and turn towards the puck of interest.

Task switching

Each robot should decide the puck color that is to be foraged for in order to regulate the division of labor in a multi-robot system. The individual robot can estimate the approximate global task demand using puck information stored in the history and by measuring the ratio of the observed puck colors. After a robot updates its puck history in an observation behavior, it re-evaluates the task switching function and decides whether it should change its current foraging task or not. There is no communication between robots, and robots only use the information in their own puck history to estimate the task demand. Extra time is required to change the task, and the robots pause at their current location to change their task. Details regarding the proposed task switching function are provided in the next subsection.

3.2 Task allocation with fixed response threshold

3.2.1 Task selection method

The desired task distribution can be obtained by appropriately selecting the individual transition rate. Based on the response threshold model, the following task switching function is defined.

$$P_{ij}(t) = S_{ij}(t) - \theta_{ij} \tag{3.1}$$

where $S_{ij}(t)$ is the estimated global task demand for the j-th task of the i-th agent at time step t, and θ_{ij} is the response threshold value for the j-th task of the i-th agent that determines the tendency to perform the corresponding task. For each agent, the score $P_{ij}(t)$ is calculated using the difference between the task demands and the threshold values for all

of the tasks, and the agent chooses the task with the maximum score. If the task demand is increased and the response threshold value is decreased, the calculated value of the score $P_{ij}(t)$ is high. A robot with a lower threshold starts to perform a task earlier than one with a higher threshold for the same task. This mechanism is represented in the task switching algorithm.

The score value $P_{ij}(t)$ for the i-th agent to work on the j-th task at the time step t is obtained, and each robot switches the current performing task depending on the result obtained from Equation (3.1). In order to estimate the global task demand, the robot calculates the proportion of each type of puck by counting the number of red and green pucks in the puck history and estimates the task demand $S_{ij}(t)$ as follows:

$$S_{ij}(t) = \frac{1}{L}\sum_{l=1}^{L} color_{ij}^{l}(t) \tag{3.2}$$

where $color_{ij}^{l}(t)$ is the color j in the l-th puck history of robot i at time t. If the l-th puck color is color j, then $color_{ij}^{l} = 1$, otherwise, it is zero. L is the length of the puck history and is set to $L = 20$.

The task demand is estimated using the moving average of the color scores in the puck history. The moving average is commonly applied to the time series data to smooth out short-term fluctuations and read long-term trends. Conceptually, if the length of the puck history increases, the changes in the task demand, as well as the frequency of task switching may decrease. Therefore, an improved approach is proposed as follows:

$$S_{ij}^{'}(t) = \frac{1}{L}\sum_{k=0}^{L-1} S_{ij}(t-k) = \frac{1}{L}\frac{1}{L}\sum_{k=0}^{L-1}\sum_{l=1}^{L} color_{ij}^{l}(t-k) \tag{3.3}$$

The first method in Equation (3.2) (called History1) uses the moving average of the puck history; the second method in Equation (3.3) (called History2) uses the moving average of the estimated task demand obtained from the first method. This method enlarges the length of the task history to $2L$.

Each robot has an equal number of response threshold values for the given tasks. In the foraging task, each robot has two types of thresholds for foraging red and green pucks, respectively. In the response threshold model, the division of labor can be regulated depending on the distribution of the response threshold values in the group. In previous works, the effects of randomly selected threshold values [119] and a single threshold value for se-

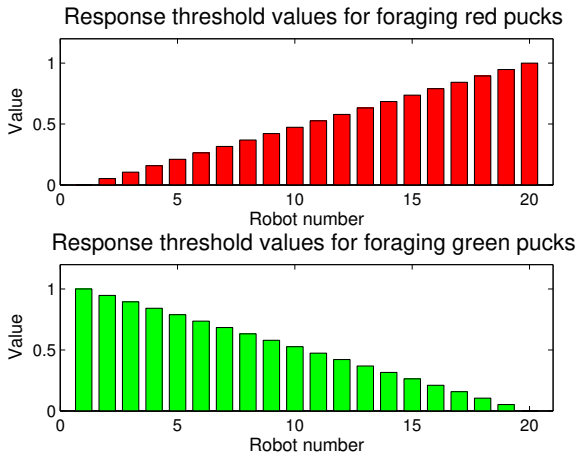


Figure 3.3: Response threshold values of robots for two tasks.

quentially ordered tasks [125] were studied. In the basic concept of the response threshold model, a single threshold value is required for a corresponding task. To apply a single threshold for two tasks, the task should be ordered sequentially, proportional to its task demand, and we would require a different type of task selection function. In the case of two tasks, the performance of a single threshold can be the same as those of the multiple thresholds. However, for more than two tasks, the performance is very different because the task can be only changed to the before or next task. In this paper, an individual robot has a constant response threshold value, which can be presented as follows:

$$\theta_{i,red} = \frac{1}{M-1} \times (i-1), \quad i = 1, ..., M \tag{3.4}$$

$$\theta_{i,green} = 1 - \frac{1}{M-1} \times (i-1), \quad i = 1, ..., M \tag{3.5}$$

where M is the total number of robots. The assigned response threshold values for each task can be between zero and one as shown in Figure 3.3. The sum of the response threshold values for each robot is one. Here, we require an individual agent to have response threshold values at evenly-spaced intervals within the range from the minimum to the maximum value. Subsequently, we shall compare the proposed distribution of the response threshold values with the randomly selected values [119].

Based on the response threshold model, the tendency for a specific task is changed de-

pending on the estimated task demand $S_{ij}(t)$ and the response threshold value θ_{ij}. The individual response for the same task demand varies according to the individual response threshold value for each robot; this method used for selecting a specific task is a specialization for a specific task The agents with low threshold values tend to specialize in the corresponding task. The smaller the threshold value is for a given specific task, the greater is the activation achieved for that task. Therefore, the task specialization for a division of labor is well demonstrated using this model. This specialization tendency can also reduce the number of task switches. If we assume that there is a cost incurred for task switching, time consumption, and the minimization of such task changes would be advantageous for group behavior.

3.2.2 Simulation results

To analyze the proposed algorithm in the foraging task, not only the total count of foraged pucks, but also the total occurrences of task switching for all of the robots are measured. Fifty pucks were randomly distributed in an arena, and twenty robots continuously moved from one place to another to forage for pucks. At the beginning, half of the robots were tasked with foraging for a red puck, while the others were tasked with foraging for a green puck. For each experiment, 20 independent runs were averaged, and the results with History1and History2 were compared to a ratio model (called Ratio) [95].

Result with changes in task demands

First, the basic situation in which the ratio of pucks was changed in the time course was used as a test condition. During the first 1,000 simulation time steps, the number of red pucks was maintained at 30%. That is, there were 35 green pucks and 15 red pucks in a given arena. After the first 1,000 time steps, the number of red pucks was switched to 80% and 50% for the next two consecutive sets of 1,000 simulation time steps, respectively. The results of the simulations during the aforementioned 3,000 simulation time steps are shown in Figure 3.4. The red dashed-dotted line represents the performance of the ratio model; the other lines represent the results obtained on using the response threshold model. The blue dotted line represents History1; the green solid line represents History2. The error bar shows the standard deviation over 20 runs.

In all of the methods used, each type of puck, red and green—was steadily collected as

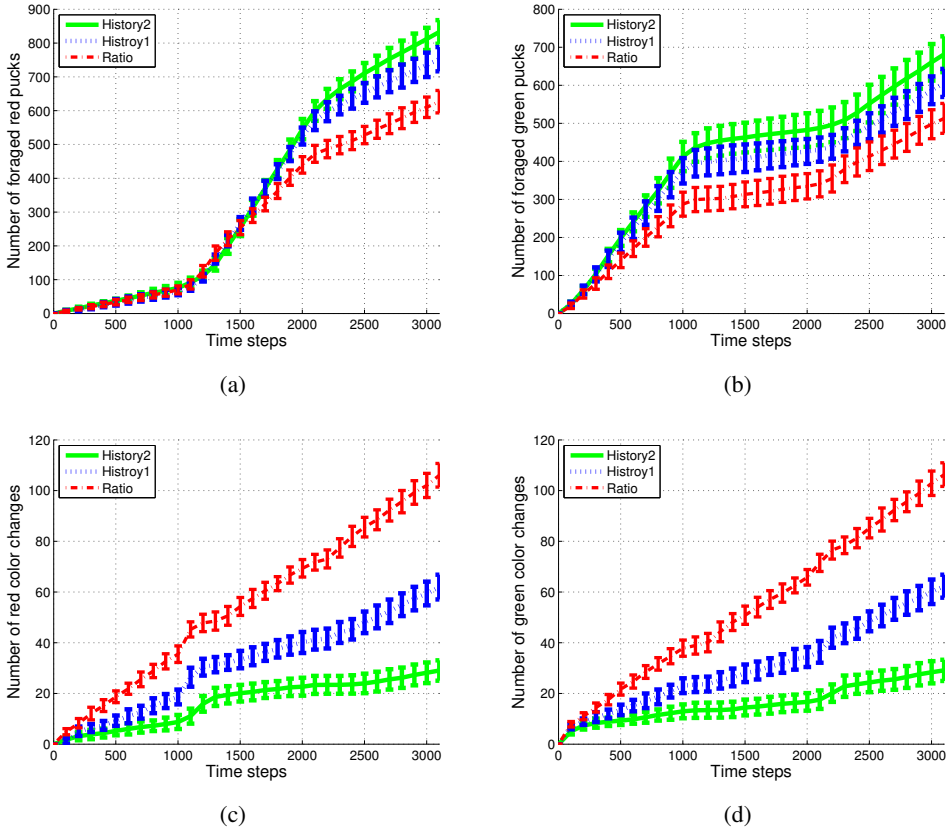


Figure 3.4: Number of foraged: (a) red and (b) green pucks and occurrences of task switching in foraging for (c) red and (d) green pucks.

shown in Figure 3.4(a),(b). However, the greatest number of pucks was foraged in History2. This means that robots could spend more time in collecting pucks, which would reduce the time wasted. As shown in Figure 3.4(c),(d), in terms of reducing the task switching, the response threshold model, especially History2, showed a greater improvement than the result of ratio model. The frequent task changes lead to the poor foraging performance of the ratio model.

These results are obtained owing to the specialization tendency of each robot. If some robots start to intensively forage for one specific puck, those robots had a tendency to forage for pucks of the same color. This tendency is demonstrated well in Figure 3.5, which displays the number of pucks foraged by an individual robot. The first row presents the results of History2. Each row shows the foraged red pucks, foraged green pucks, task changes for

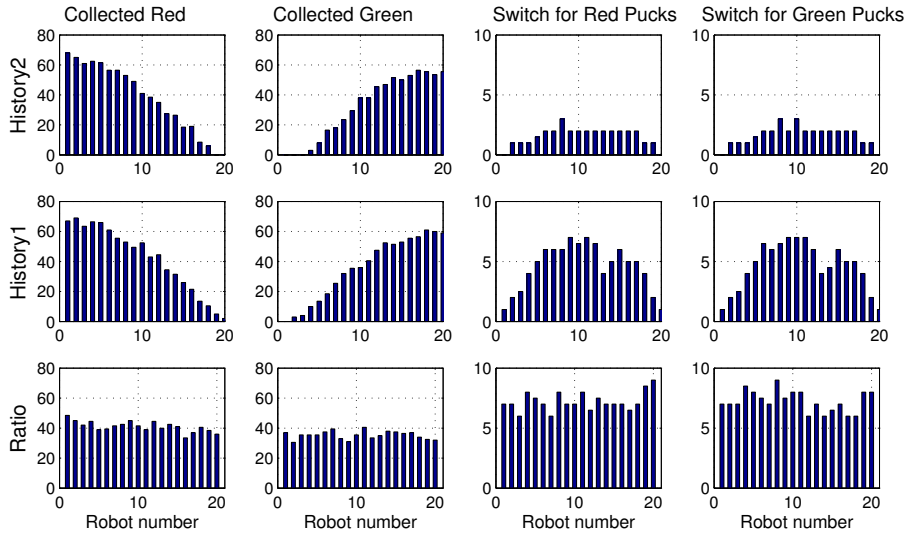
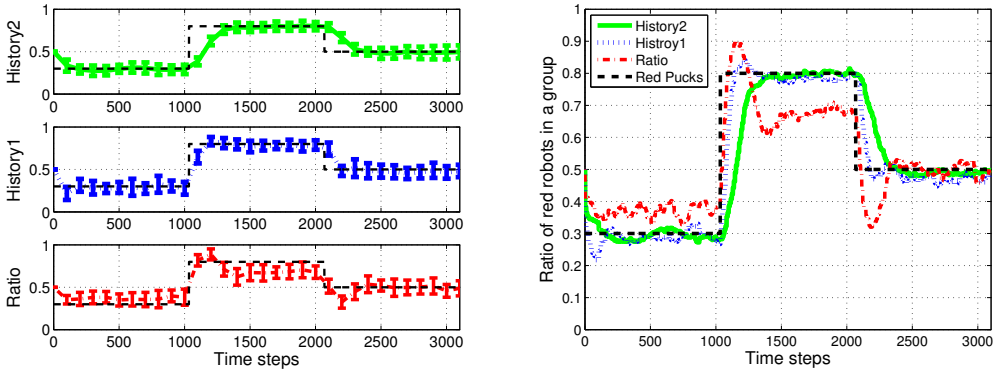


Figure 3.5: Number of foraged pucks and task changes of an individual robot.



(a) (b)

Figure 3.6: Ratio of robots foraging for red pucks in a group: (a) individual trend of each algorithm; and (b) overlapping trends of all algorithms.

the red pucks and task changes for the green pucks in the left-to-right direction.

In the ratio model, all of the robots foraged for red and green pucks in a similar manner. However, in the response threshold model, the pucks were selectively foraged for by each robot. Some robots were strongly specialized in a specific task. From the results, it can be observed that Robots 1 and 2 only foraged for red pucks and Robots 19 and 20 preferentially foraged for green pucks. Accordingly, there were few task changes for another type of puck

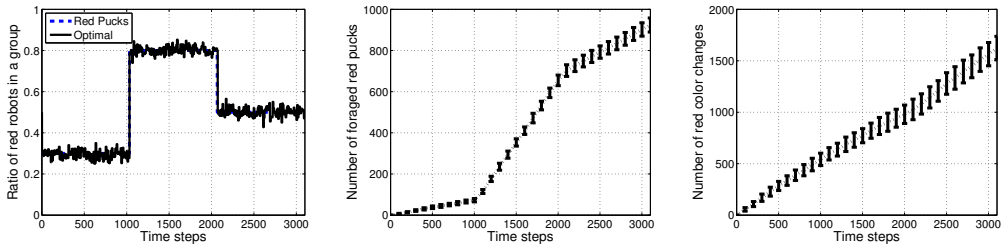
in these robots. However, other robots were softly specialized depending on a slight gap between the response threshold values of the two tasks, and they foraged for both types of pucks. Therefore, the majority of the task changes occurred for these robots. We often observe that time and energy is required for task changes in real application problems. Minimizing the task changes would reduce this time consumption, and thus, History2 may be a preferable solution.

Figure 3.6 shows the changes in the ratio of robots foraging for red pucks in each algorithm. Both History1 and History2 show the same ratio of robots foraging for red pucks in a group as the ratio of red pucks. $History1$ shows a faster convergence tendency; however, it shows greater overshooting than $History2$. In general, the shorter puck history resulted in a quicker convergence to the desired state; however, more frequent task changes are required for robots in a group than for individual robots.

The ratio model showed the quickest reaction to the task changes for red pucks. Nevertheless, it showed the worst performance in terms of overshooting and some errors in the ratio of robots foraging for red pucks. In the ratio model, a robot focused on balancing the estimated task demand and the ratio of the foraging task performed by neighboring robots. In the case in which the proportion of one specific task was given, a few extra robots were assigned to forage for the minor-color pucks. This feature produced a better result in the foraged pucks with a minor proportion in the whole population of pucks. In the two cases in which the ratios of the red pucks were 30% and 80%, respectively, there were more robots foraging for minor pucks than the ratio of minor pucks, and this tendency caused some gap between the global task demand (portion of two types of pucks) and the assigned robots in the ratio model.

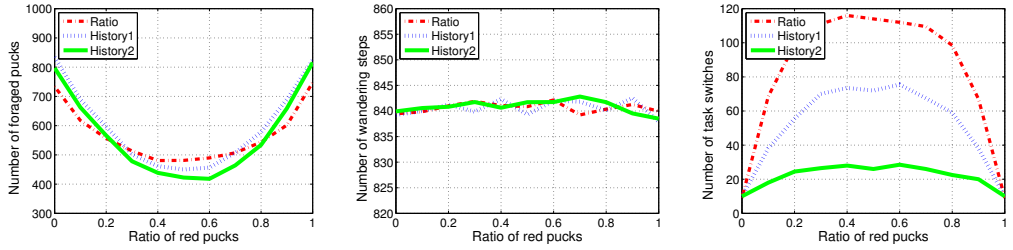
Figure 3.7 shows the results of an optimal method with the assumption that all of the agents knew the exact ratio of each puck. In this case, an individual robot could easily select its task using a stochastic strategy and select a task probabilistically in proportion to the ratio of tasks. Further improved performances can then be obtained in the aspects of accuracy in the ratio of red robots and the number of foraged red pucks as shown in Figure 3.7(a),(b). However, there is a requirement for more than tens of times the task switches as shown in Figure 3.7(c).

An individual robot has to exert energy to forage for pucks and to grip the pucks. In addition to these costs, some extra costs were incurred due to the task switching. To observe the effect of task switching, the three methods, History1, History2 and ratio, were evaluated for varying ratios of red pucks for 1,000 simulation time steps with no additional time for



(a) (b) (c)

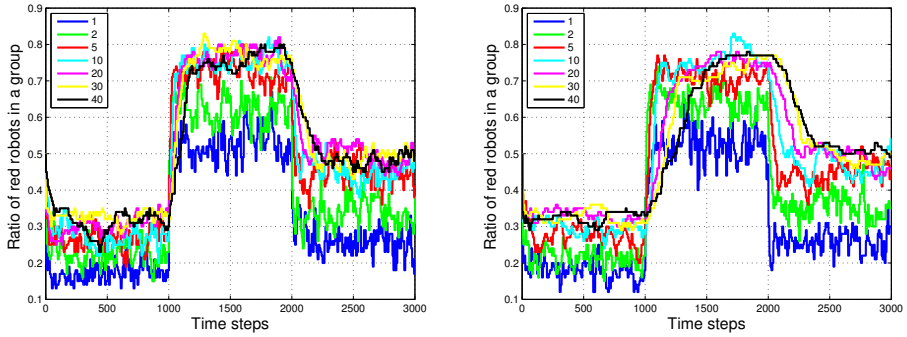
Figure 3.7: Result of an optimal method using a stochastic approach: (a) ratio of red robots in a group; (b) number of foraged red pucks; and (c) number of task changes in foraging for red pucks.



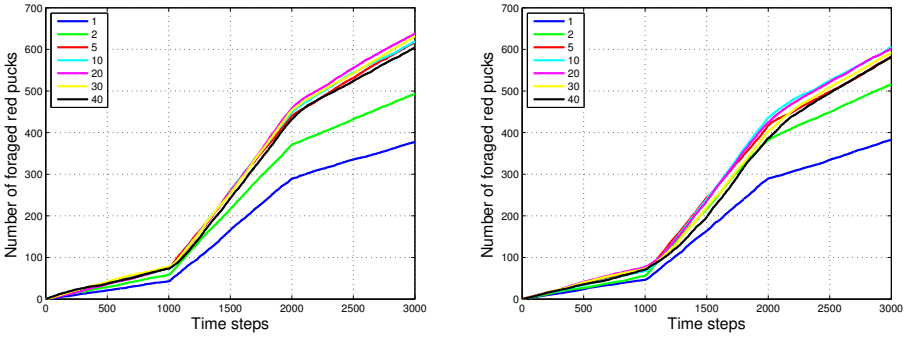
(a) (b) (c)

Figure 3.8: Performance evaluation with varying ratios of red pucks: (a) number of all foraged pucks; and (b) number of wandering steps; and (c) number of task changes.

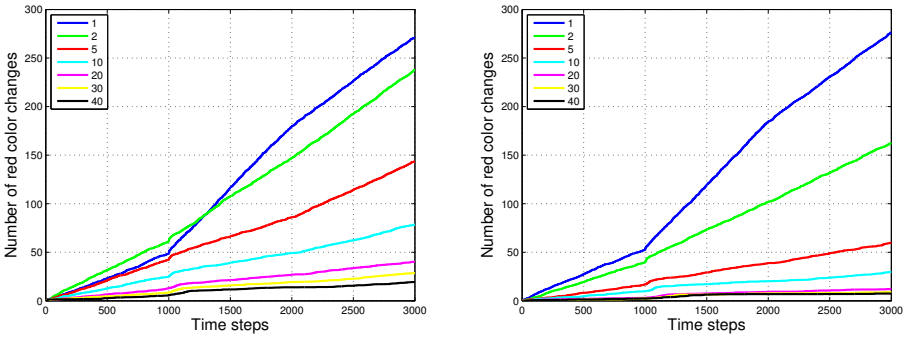
the task change, but only counting the number of task changes. The performance in the collection of pucks changed depending on the ratio of foraging tasks. The total number of foraged pucks and wandering steps was almost the same for the three algorithms, as shown in Figure 3.8(a),(b). Therefore, the energy required for wandering and gripping did not differ across these cases; moreover, the total consumed energy mainly depended on the number of task switches for each experiment. History2 showed an improved performance in the aspect of the task changes as shown in Figure 3.8(c). The results for the two methods, History1 and History2 compared with those for the ratio model in terms of change in demand were shown. The response threshold models used a fixed constant-sized history for storing the most recently observed puck types. History2 exhibits a better performance than History1 in terms of task switching. This induced an improved performance in the number of foraged pucks, while it probably guessed the global task demand (the current proportion of pucks in the environment).



(a)

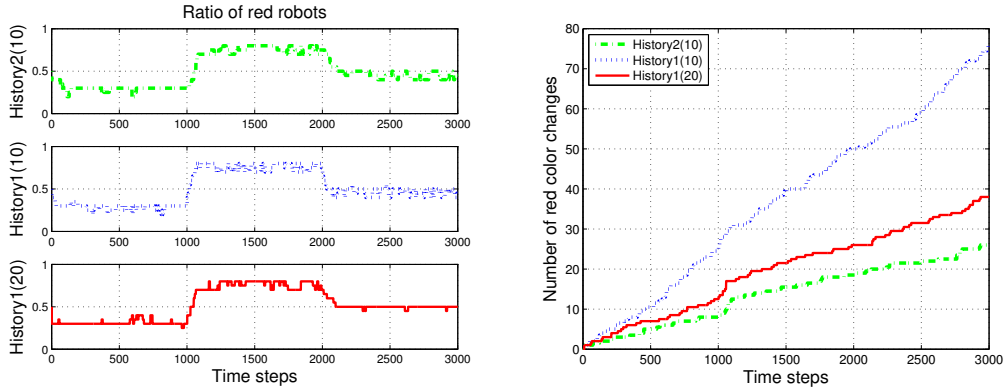


(b)



(c)

Figure 3.9: Performance of the History1 and History2 methods with various lengths of the puck history. The various colors indicate varying sizes of the puck history from (L=1) to (L=40): (a) ratio of red robots in a group in History1 (left) and History2 (right); (b) number of foraged red pucks in History1 (left) and History2 (right); and (c) number of task changes for red pucks in History1 (left) and History2 (right).



(a) (b)

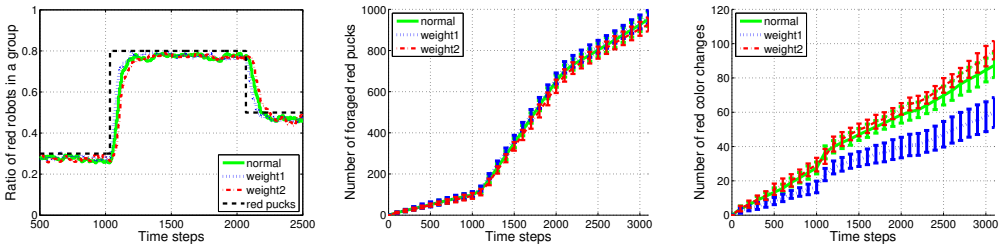
Figure 3.10: Comparison results of History1 (L=10), History1 (L=20) and History2 (L=10): (a) ratio of red robots in a group; (b) number of red color changes. The numbers in parentheses refer to the length of the puck history.

Therefore, the difference in the performances between the ratio model and the response threshold model was evident when there were costs related to task switching. In a constraint condition with a cost for task switching, if the cost of task switching increased, the total energy wasted in the foraging for pucks increased rapidly. Task switching occurred infrequently in the response threshold model as compared to the ratio model. Thus, the response threshold model is more suitable if the task switching cost is high.

Results with changes in size of history

Figure 3.9 shows the experimental results of the two methods, History1 and History2 for varying sizes of the puck history. For the two methods, as the length of the puck history increases, the accuracy of the proportion of robots foraging for red pucks in a group increases. The number of robots collecting red pucks converged to the desired results; however, the total number of task changes decreases as the length of the task queue increases. We also observed that the time required to converge to the ratio of robots to forage for red pucks increases with the History2 method. This effect is the result of the specialized tendency of History2.

Generally, the total number of task switches decreases when the size of the task queue increases. When the two methods with the same size of task queue were compared, His-



(a) (b) (c)

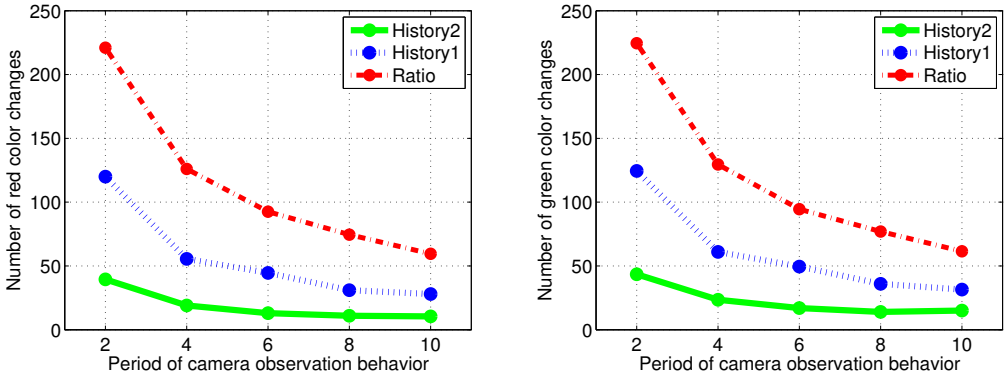
Figure 3.11: Results of weighted History1 (L=10): (a) ratio of red robots in a group; (b) number of foraged red pucks; and (c) number of red color changes.

tory2 was found to be more effective than History1 (see History1 with L=10 and L=20 and History2 with L=10 in Figure 3.10). As the size of the task queue with History1 increases to twice the original value (from L=10 to L=20), the ratio of red robots becomes more accurate, i.e., even better than that in History2 (L=10). Actually, History2 uses double the size of the task queue used in History1 for L=10. However, the number of task switches for History2 (L=10) is still less than that for History1 (L=20). The moving average of the puck history has a positive effect on the task changes.

Figure 3.11 shows the results of the weighted History1 (L=10); the green solid line represents the original History1, the blue dotted line represents the weighted History1 where the weight is set to $[1^2, 2^2, ..., (L-1)^2, L^2]$ to emphasize the newest history; and the red dashed-dotted line represents the reversely weighted History2. Owing to the weight of the newest history (weight1), a slightly faster response to the change in task demand and improved performance in terms of smaller task changes were observed with the appropriate proportion of pucks to the task demand than the original History1.

Results with changes in vision sampling period

To avoid recording the same puck repetitively in the puck history, the robots sense the environment once every eight time steps using visual information. Figure 3.12 shows that increasingly frequent vision sensing leads to increased task changes. In addition, the performance of the response threshold model was more robust for the change in the vision sampling period than in the ratio model.



(a) (b)

Figure 3.12: Comparison of the results of the number of task changes with a variation in the vision sampling period from two to ten time steps: (a) task changes for red pucks; and (b) task changes for green pucks.

Results with fixed number of tasks

The case in which the foraged pucks are not reproduced was considered, and Figure 3.13 shows the results. There were 500 pucks in the same arena; 20 robots wandered in order to forage for the pucks. The proportion of red pucks was set to 30% at the starting of the simulation. The foraged pucks were not reproduced in the arena. Therefore, the total number of pucks decreased as time passed. In this foraging task, the division of labor results differed minimally from those of the ratio model.

The red and green pucks were steadily foraged in both the models, as shown in Figure 3.13(a). The number of task switches was smaller in the response threshold model, as shown in Figure 3.13(b). This is similar to the results obtained in the previous experiments. However, several robots switched their tasks to forage for green pucks at an earlier period. The green pucks were preferentially foraged because the ratio of green pucks was higher than that of red pucks. After some simulation time steps, the robots began to switch their states to forage for red pucks according to the increasing task demand for foraging for red pucks. Therefore, the ratio of robots foraging for red pucks and foraged red pucks changed, as shown in Figure 3.13(c),(d). These features are largely shown in the response threshold model, especially in History2, which had a strong specialization characteristic.

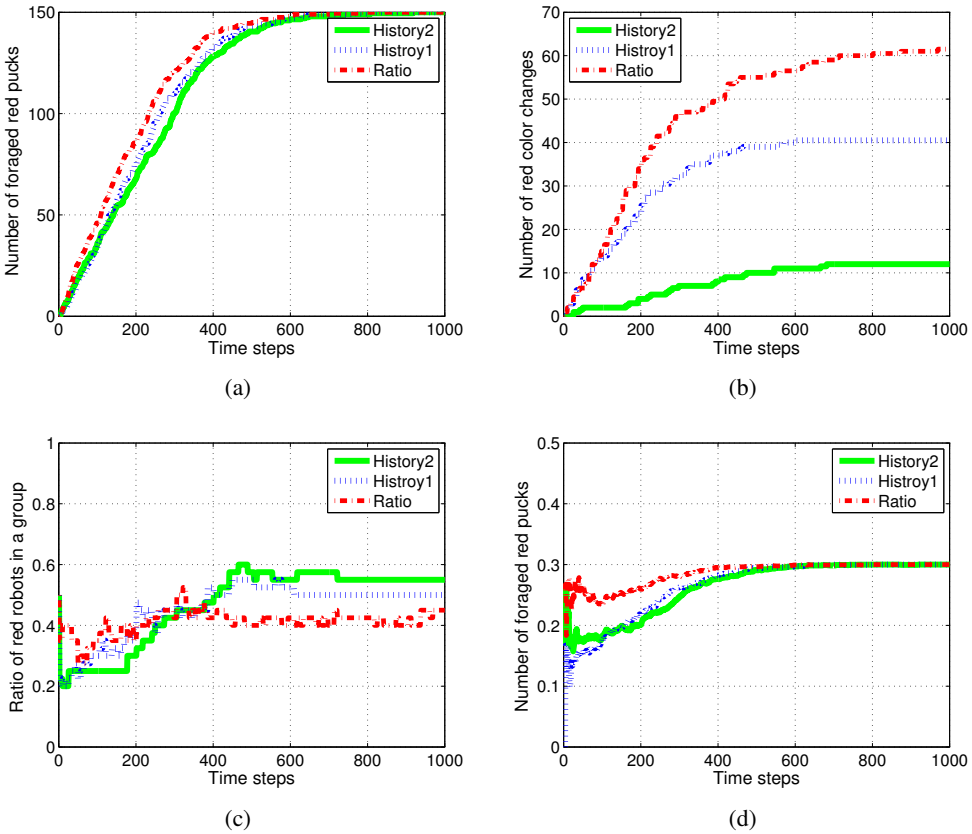


Figure 3.13: Results of foraging tasks when the foraged pucks were not reproduced: (a) number of foraged red pucks; (b) number of task switches in foraging for red pucks; (c) ratio of robots foraging for red pucks; and (d) ratio of foraged red pucks.

Results with changes of threshold distribution

It is necessary to determine the appropriate response threshold value to improve the system performance in the response threshold model. In the proposed approach, the response threshold values in the group of robots were at evenly-spaced intervals within the range from the minimum to the maximum value. If randomly assigned values were used, as shown in Figure 3.14(a), a slightly decreased performance in terms of the ratio of the division of labor in the group was obtained, as shown in Figure 3.14(b). However, more improved performance in the rate of foraged pucks as compared to the ratio model was still obtained. In addition, the number of task switches was markedly improved in comparison to the ratio model, as shown in Figure 3.14(c),(d).

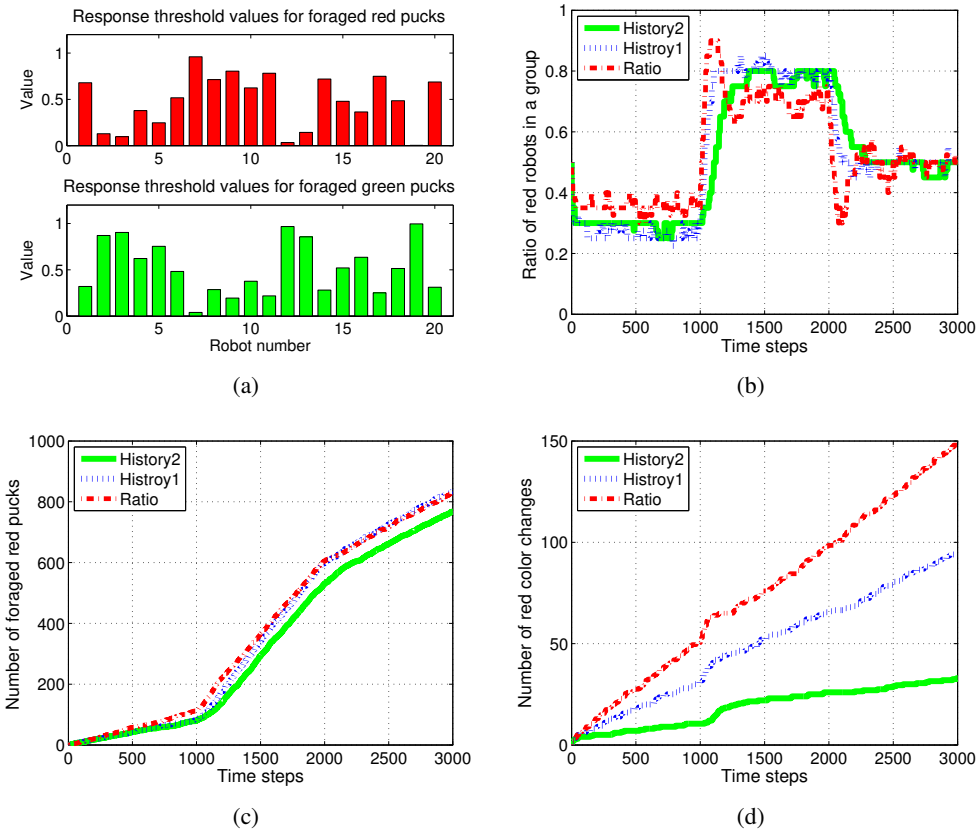


Figure 3.14: Randomly distributed pattern for response threshold values for two foraging tasks: (a) response threshold value in a random pattern; (b) ratio of robots for red pucks in a group; (c) number of foraged red pucks; and (d) number of task switches for red pucks.

3.2.3 Drawback of specialization in foraging task

Despite the previous results, the proposed method based on the response threshold model can be effective in all aspects. If the proportion of foraged pucks is considered as the index of the system performance, the response threshold model showed a worse result than the ratio model. A robot in a response threshold model has a specialized tendency to perform one specific task. Thus, specialized robots should move a much greater distance to forage for specific color pucks without task switching. However, if the proportion of some pucks is much smaller than those of other colors, the probability of obtaining the specific puck is reduced, and the minor pucks have a decreased chance of being foraged by the robots than in the ratio model.

In fact, in a circular arena, the probability that the robots will forage for specific pucks is related to the square of the ratio of the task. It the portion of pucks is 30% and 70% for red and green pucks, respectively, the chances that the robots with the task of foraging for red pucks will find the red pucks may be $0.3^2/(0.3^2 + 0.7^2) * 100\%$. The probability of obtaining a foraged red puck may be 15.5%. Figure 3.15(a) to 3.15(f) show the results with some changes in the experimental settings when the proportion of red pucks was fixed to 30% during 1,000 simulation time steps; (a) and (b) show the results with the vision camera angle 60° from the front; (c) and (d) ratio show the results with the vision camera angle 20° from the front; and (e) and (f) show the results with the vision camera angle 20° from the front; the number of pucks increased three times. Figure 3.15(a),(b) shows the ratio of robots foraging for red pucks and the foraged red pucks. Although the ratio of robots in the group matched the ratio of tasks, the ratio of foraged pucks was lower for the response threshold model, as expected. Therefore, if the ratio of foraged pucks is regarded as the system performance measure, the response threshold model showed a weakness because there is a trade-off between the accuracy and the number of foraged minor pucks.

If the robots capture images of the front with a narrow angle, the exact estimation of the global task demand will be difficult, and the results of the division of labor will decrease in accuracy and will require more time to become stable, as shown in Figure 3.15(c),(d). However, when the density of the distributed pucks increased by three times that of the original experiment, despite the poor camera detection ability, the performance of the division of labor increased in accuracy, as shown in Figure 3.15(e),(f). In a foraging task in a circular arena, the results obtained with various systems are related with not only the ability of the robots, but also the surrounding environments. Additionally, the obstacle avoiding behavior and distribution of pucks in a circular arena make it difficult to search for specific pucks. Here, pucks are not considered as obstacles, and robots can pass by the pucks easily. If the pucks are considered as obstacles, the movements of the robots in a fixed area become restricted. The inaccuracy in the estimation of task demand and frequent obstacle avoidance may limit the specialization tendency in the response threshold model.

We tested another experiment in which robots treated the pucks as obstacles. Figure 3.16 shows the ratio of robots foraging for red pucks in a group, and the number of task switches for red pucks when the pucks are considered as obstacles. These performances are evaluated for the same foraging task as used in Figures 3.4 and 3.6; however, pucks are considered as obstacles. We found a slight mismatch between the global task demand and the ratio of foraged robots in both History1 and History2, as shown in Figure 3.16(a), and

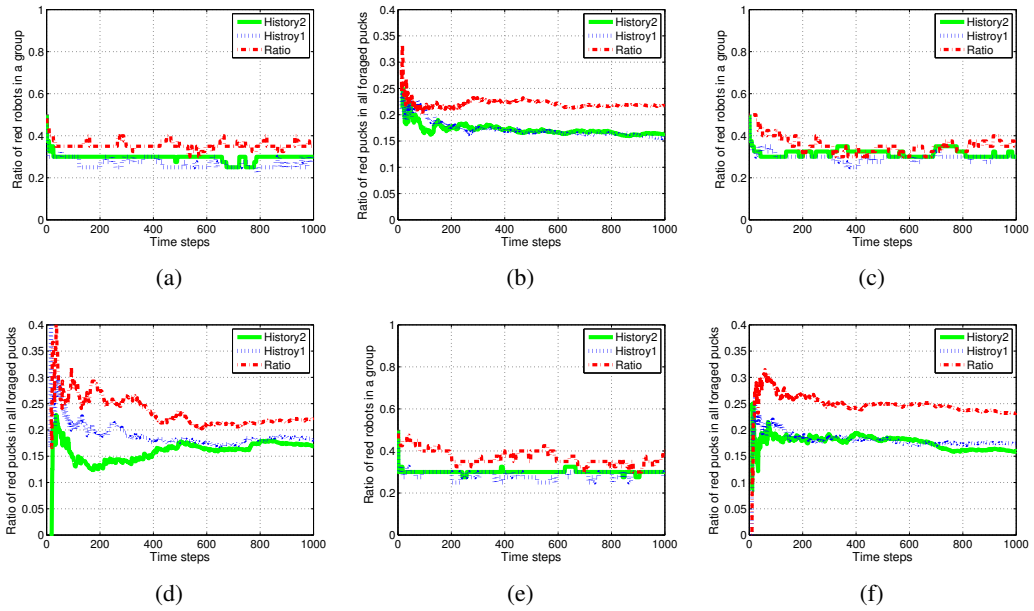
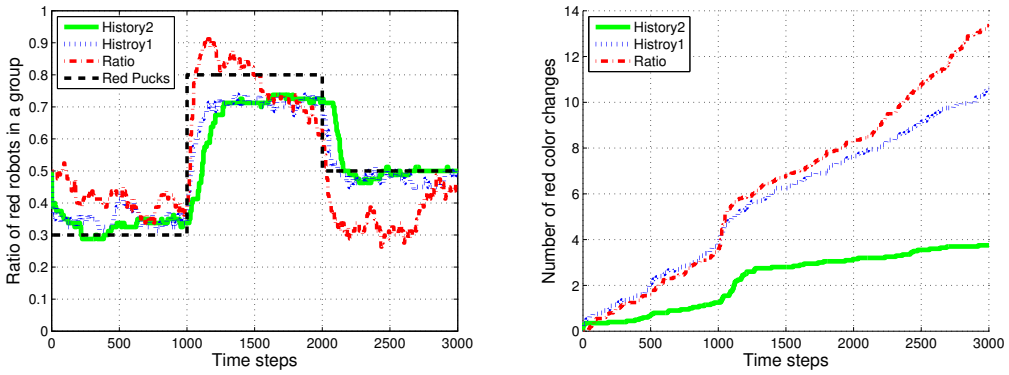


Figure 3.15: Ratio of robots foraging for red pucks in a group and the ratio of red pucks to all foraged pucks: (a) robots foraging for red pucks (front 60°); (b) ratio of red pucks (front 60°); (c) robots foraging for red pucks (front 20°); (d) ratio of red pucks (front 20°); (e) robots for red pucks (front 20°, triple pucks); and (f) ratio of red pucks (front 20°, triple pucks).

the number of task changes for red pucks was similar to the result obtained with ratio, as shown in Figure 3.16(b). However, History2 still showed the best performance in terms of reducing the task changes among the three algorithms.

3.3 Task allocation with variable response threshold

In the previous section, uniformly distributed threshold values is given in advance for each agent in a group and these values are not changed during performing task. We showed that the distribution of thresholds determines the accuracy of division of labor and its performance is much improved than the previous method, especially in the aspect of task changes. However, the response threshold values should be calculated manually and it is difficult to assign a proper value in case of multiple tasks. In addition, this approach does not satisfy the characteristics of a swarm robotics; robust, scalable, and flexible. We should apply the



(a) (b)

Figure 3.16: Ratio of robots foraging for red pucks in a group, and the number of task switches for red pucks: (a) robots for red pucks in a group; and (b) number of task switches for red pucks.

variable threshold method to obtain more adaptive and improved performance.

3.3.1 Task selection method

There is also a number of tasks with their associated demand. We assume that the tasks are ordered in a sequence and an appropriate level of response threshold for each segment can be defined as shown in Figure 3.17. If an agent has its response threshold within the range of a specific task, it is assigned to the task. This concept is the same with task assignment tendency in honey bee colonies. The threshold represents the physiological age and different tasks are performed during its life-time in a process of behavioral development.

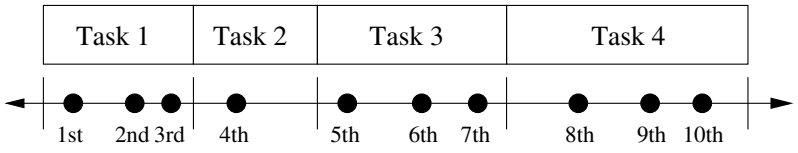


Figure 3.17: Example of diagram used for task allocation. Four tasks are ordered in a sequence proportionally to its demands and the threshold values of ten agents are spaced. Agents belonging to range that is split into segments relative the task demands is assigned to the corresponding task.

Threshold regulation

We are interested in how the patterns of all the thresholds are spread uniformly over the range, from θ_{min} to θ_{max}, in the whole swarm for an accuracy task allocation [123, 122]. For this, we design a simple algorithm via social interaction inspired by jamming avoidance response. This behavior is performed by some species of weakly electric fish [222]. It occurs when two electric fish with very similar discharge frequencies meet. Each fish shifts its discharge frequency to increase the difference between the two fish's discharge frequencies. By doing this, both fish prevent jamming of their sense. The jamming avoidance behavior escapes jamming of close frequencies.

Inspired by this behavior, we design a simple model for regulation of threshold values between a pair of individuals within a limited sensing range. If agent i meets agent k, the threshold value θ_i of agent i and θ_k of agent k are updated following the rules:

Rule 1. If $\theta_i > \theta_k$ and $|\theta_i$ - $\theta_k| < \alpha$, then θ_i = θ_i + δ and θ_k = θ_k - δ.
Rule 2. If $\theta_i < \theta_k$ and $|\theta_i$ - $\theta_k| < \alpha$, then θ_i = θ_i - δ and θ_k = θ_k + δ.
Rule 3. If θ_i = θ_j, then θ_i=$\theta_i \pm X$ and θ_k=$\theta_k \pm X$.

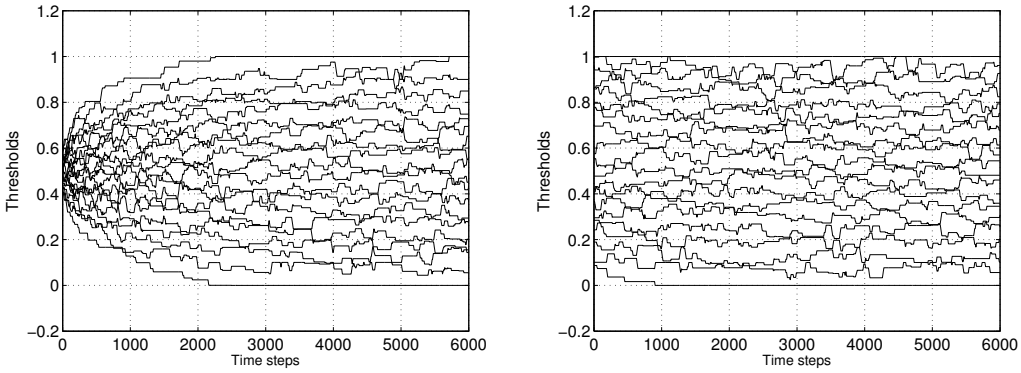
where α is a minimum difference of thresholds between individuals, δ is a constant parameter, and $X \in (0, 1)$ is a random value. θ_i is restricted to a range of (θ_{min}, θ_{max}). Each agent updates its threshold for every interaction with another agent. Repeated interactions among members can lead to an almost uniform distribution of thresholds, regardless of initial thresholds.

3.3.2 Simulation results

Figure 3.18 shows distributed thresholds when the initial thresholds of all agents are almost the same and randomly distributed over the range. We can see that the proposed method can spread the response thresholds of the whole swarm agents. Using these variable thresholds, three different experiments are performed to test the performance.

Results with changes in task demands

In the first experiment, we investigate the adaptability of the method responding to changes in task demands. There are two tasks, foraging red pucks (task 1) and green pucks (task 2).



(a) (b)

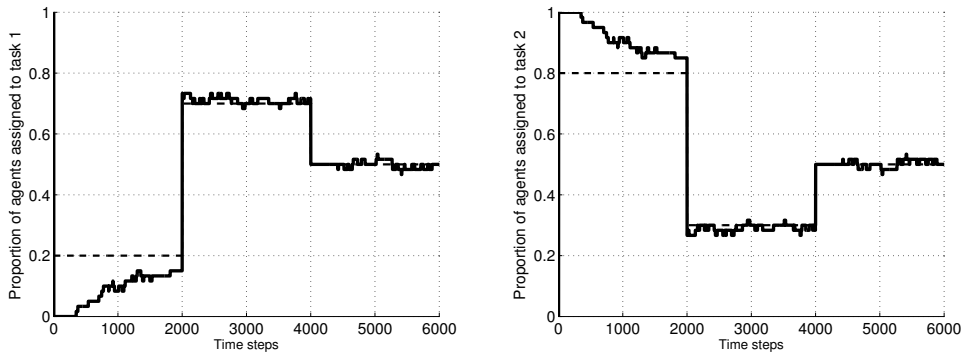
Figure 3.18: Changes of thresholds with $\alpha = 0.1$ and $\delta = 0.001$ from the different initial distribution: (a) initial thresholds are almost equal; and (b) initial threshold are randomly distributed over the range.

At the beginning, the proportion of task 1 and task 2 is set to 20% and 80%, respectively. At the time step 3,000, it is changed to 70% and 30%, and at the time step 6,000, it is changed to 50% and 50% in sequence. Figure 3.19 represents the results. Since all the robots start with the same task, they all start with task 1. As time passes, the swarm is split into two groups with the same proportion of task demands. By changing the task demands, the proper proportion of agents is re-assigned to the changes in task demands.

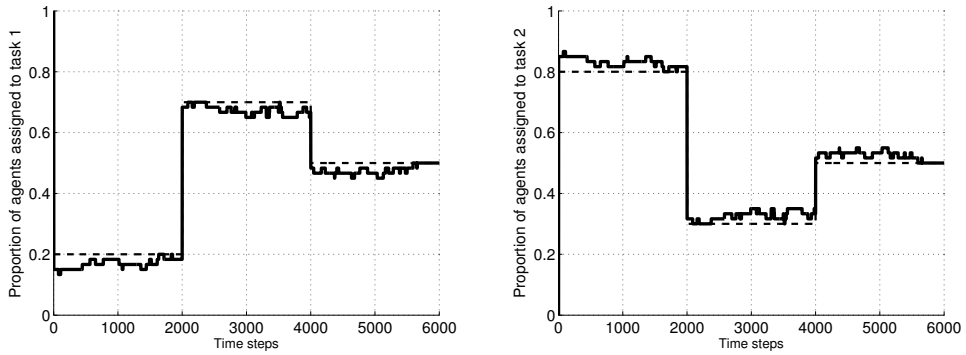
The accuracy of division of labor may be a little changed because the distribution of thresholds are varied depending α and δ. But the overall performance still converged to the desired state. The result shows that task allocation can also be obtained using variable threshold values.

Results with changes in number of agents

Figure 3.20 shows the robustness and flexibility of the swarm when the number of agents is changed. During the first 1,500 simulation time steps, about 50% agents are assigned to task 1 because the task demand is fixed to 50%. After that time, all agents assigned to task 1 are removed from the arena. At that moment, we see that the proportion of agents performing task 1 is dropped to 0%. However, about 50% agents of the remaining agents are re-assigned to task 1. Figure 3.20(a) represents the behavior of the swarm in response to the changes and



(a)



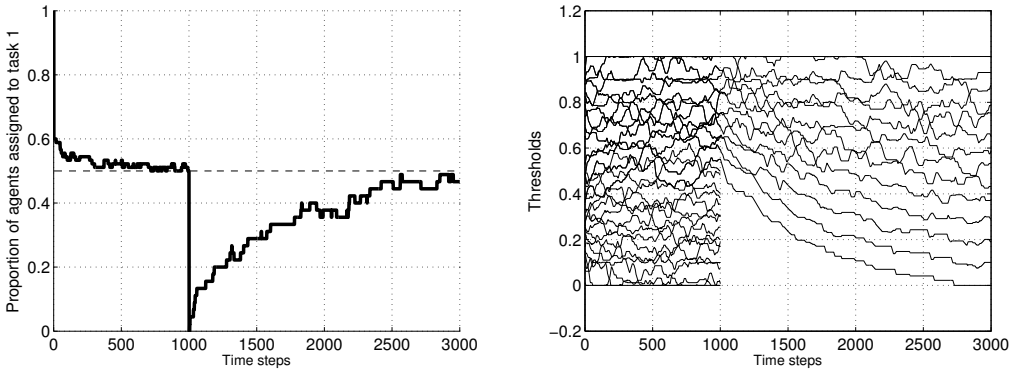
(b)

Figure 3.19: Proportion of agents assigned to each task with changes in task demand: (a) proportion of agents performing each task with $\delta = 0.01$; and (b) proportion of agents performing each task 2 with $\delta = 0.001$ (left-task 1, right-task 2).

it shows that the swarm reacts properly, regardless of the number of agents. The threshold values are re-balanced among the remaining agents after 1,500 simulation time step and the threshold difference between agents are larger than the previous period as shown in Figure 3.20(b) due to the decreasing number of swarm. The thresholds are continuously updated by the self-organizing process in the dynamically changing environment.

Results with multiple tasks

In the last experiment, we consider a distribution of a population of agents for more tasks; four tasks rather than two tasks. The proportion of each task is set to 30%, 30%, 20%, and



(a) (b)

Figure 3.20: Proportion of agents assigned to a specific task when some agents are removed from the arena during task: (a) proportion of agents performing task 1; and (b) changes of threshold values.

20% and the proportion of robots assigned to each task is shown in Figure 3.21. The swarm reacts properly and the proportion of agents assigned to each task reaches the desired level regardless of the increasing number of tasks.

3.4 Summary of Chapter 3

To perform various tasks, an individual agent in a group sometimes needs to choose the task that is most commensurate with its current state in the overall system. In this chapter, we consider a task allocation algorithm for regulating the proportion of agents performing tasks so that it is equal to the proportion of task demands. To solve these problems, we propose a decentralized strategy based on the response threshold model.

In centralized approaches, it is possible for an individual agent to have needed information concerning task demands and the tasks of other agents. They can then easily select proper tasks. However, in decentralized approaches, agents are unable to have global information due to their limited sensing abilities, requiring them to use their local knowledge in other ways. Each individual member has a constant task demand history that reflects global demand. In addition, each individual has response threshold values for all tasks and manages the task switching process in response to the stimuli of the task demands. The

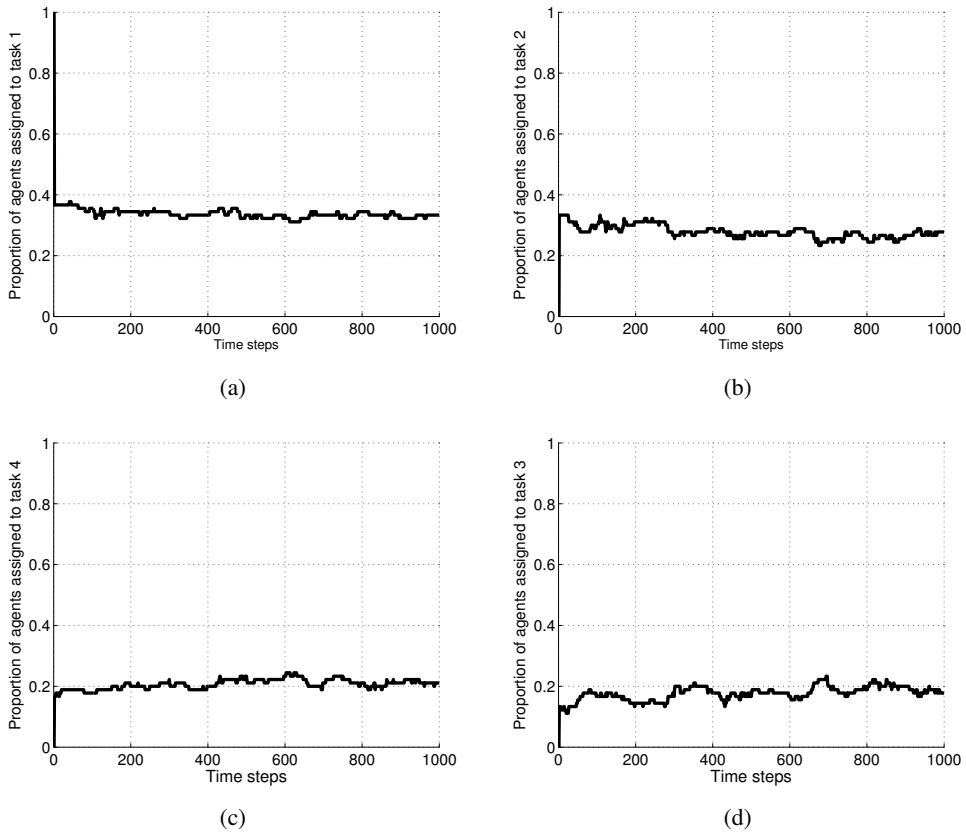


Figure 3.21: Proportion of agents assigned performing multiple tasks: (a) proportion of agent assigned to task 1; (b) proportion of agent assigned to task 2; (c) proportion of agent assigned to task 3; and (d) proportion of agent assigned to task 4.

robot then determines the task to be executed to regulate the overall division of labor. This task selection induces a specialized tendency to perform a specific task and regulates the division of labor.

We investigate performance using a simulated swarm of moving robots that perform a foraging task. The foraging task requires task allocation among multiple robots over a set of tasks. Various experiments are performed. The results show that the proposed algorithm is comparable with the conventional model. In addition, we note that the system performance varies greatly depending on the response thresholds and the variable response threshold model is more appropriate for satisfying the robust, scalable, and flexible characteristics required for multi-agent systems.

Chapter 4

Task allocation for parallel tasks

Multi-agent systems need to be adaptable to changes in the environment and the individual worker needs to switch tasks according to the task demands. The mechanisms to solve such an adaptive task allocation problem in the face of various internal and external states are thus of great interest. Inspired from the task allocation in the insect colonies, for example task assignment in honeybee is regulated via social inhibitions, we propose a flexible task allocation mechanism affected by not only tasks but also existence of other agents.

For a desired distribution of tasks over a swarm of robots, we propose a decentralized strategy following a linear differential system for task transition, which regulates the division of labor based on the response threshold model. The proposed algorithm can be applied to multiple robots for foraging tasks, each of which uses local environmental information about the task demand and the tasks assigned to neighboring robots. Robots choose or switch their tasks based on the information even in dynamically changing environment.

We investigate the response of the overall system on the change of task demands or the number of agents. We also see that the overall system converges to a desired task distribution in a group level, trying to reduce the number of task change occurrences due to the specialized tendency of the response threshold. Then we applied the proposed approach to factory domain application that minimizing the occurrence of task changes is beneficial because each change incurs additional time and material costs. The goal is to minimize the number of task changes of the overall system while it maintains the desired performance level such as throughput.

This chapter is submitted in journals [126, 128].

4.1 Methods

There are multiple candidate tasks and each robot is equally capable of completing each task. Only a single type of task can be assigned to each robot at any time and each robot is moving around in the given arena performing an assigned foraging task during a given time span. In the dynamic task environment, the number of tasks or the number of robots may be changed, for example, by adding new tasks or removing some robots. A robot may switch from its own task to another if that task is highly demanded in the given environment.

To solve task allocation problem, we focus on adjusting the proportion of robots performing each subtask adaptively. If it is assumed that all the subtasks need the same amount of time to complete, one possible solution is to maintain the fraction of subgroup members equal to the proportion of subtasks to be done. We presume that the overall swarm performance is maximized when the allocation of robots to subtasks is optimal.

In order to achieve the optimal allocation, robots in the swarm should change currently performing subtasks if necessary. Two different levels of modeling is available: microscopic and macroscopic. On the one hand, microscopic modeling focuses on the individual robot and on interactions among individual robots. It is difficult to apply mathematical methods, so microscopic modeling is performed. On the other hand, macroscopic modeling focuses on the swarm as a single system. For a large size of swarm, the individual robot behaviors can be described by a continuous-time Markov process in the absence of the task switching time. There has been recent work on designing density feedback controllers, which are functions of agent populations in different states, to drive a swarm whose states evolve according to Markov chain to a target distribution [14, 154, 50, 58]. The system of individual robots can be abstracted to a linear differential equation model. We first explain the linear differential model briefly, presented by Halasz et al. (2007) and Berman et al. (2009). Then we explain our proposed task partitioning method for adjusting the proportion of robots performing each subtask adaptively, which is based on the response threshold model.

4.1.1 Modeling

We consider N agents and each agent can be allocated to one task among M tasks. We denote the number of agents performing task $i \in \{1, \ldots, M\}$ at time t by $n_i(t)$. Then the population fraction performing task i at time t is defined as $x_i(t) = n_i(t)/N$, and the vector of population fractions is represented by $x(t) = [x_1(t), \ldots, x_M(t)]^T$. The desired number of robots for task i is defined by $\bar{n}_i$ and the desired target distribution is the set of population fractions for each task, $\bar{x} = [\bar{x}_1, \ldots, \bar{x}_M]^T$, where $\bar{x}_i = \bar{n}_i/N$.

We can model the interconnection topology between M tasks via a directed graph, $G = (V, E)$. A set of vertices, V, represents M tasks. For a set of edges, E, task i and j are adjacent, (i, j), if a robot performing task i can change its task to task j. The graph G is strongly connected if a path exists between any $i, j \in V$. To model the task transfer from one task to another, every edge in E is assigned transition rate, $k_{ij}(t)$, where $k_{ij}(t)$ means the task transition probability per unit time for one agent previously executing task i to switch to task j. The transition rate from i to j does not equal to the transition rate from task j to i, $k_{ij}(t) \neq k_{ji}(t)$. Using the transition rate, the population fraction of robots executing task i is given by the linear equation:s

$$\frac{dx_i(t)}{dt} = \sum_{\forall j|(j,i)\in E} k_{ji}(t)x_j(t) - \sum_{\forall j|(i,j)\in E} k_{ij}(t)x_i(t) \tag{4.1}$$

Since the number of agents is conserved, Eq. (4.1) can be equivalently represented as the linear model to represent the average change rate of the population fractions executing the tasks

$$\frac{dx(t)}{dt} = Kx(t) \tag{4.2}$$

where $K \in R^{M\times M}$ is a task transition matrix with the following properties

$$K^T 1 = 0 \tag{4.3}$$

with $K_{ij} = k_{ij}(t)$ for $i \neq j$ and $K_{ii} = 1 - \sum_{\forall j|(i,j)\in E} k_{ij}(t)$. Equation (4.2) is the formulation of Eq. (4.1) over all M tasks as a matrix equation and is referred as the reaction rate equation. Using this model, the steady-state distribution of the group over various tasks can be controlled by appropriately selecting the individual transition rates.

The studies of Halasz et al. (2007) and Hsieh et al. (2009) show that the system Eq. (4.2)

always converges to a unique task distribution regardless of the choice of K for strongly connected graph cases. It has a unique stable equilibrium. With the desired distribution $\bar{x}$ with M tasks, the population can automatically distribute the task accordingly among agents through the selection of the individual transition rates, $k_{ij}(t)$. They studied how to determine the set of constant transition rates that result in fast convergence and how to minimize task transition at equilibrium state. However, constant transition rates even force continuous task switching at equilibrium state and the optimal transition rates are calculated. A centralized system should compute the optimal transition rates to guide group behavior and provide rate information to each robot.

In our work, an individual agent selects its task autonomously without any centralized control and the transition rates are regulated adaptively depending on properties of the environment such as number of agents, number of tasks, area of arena, and others.

4.1.2 Task selection method

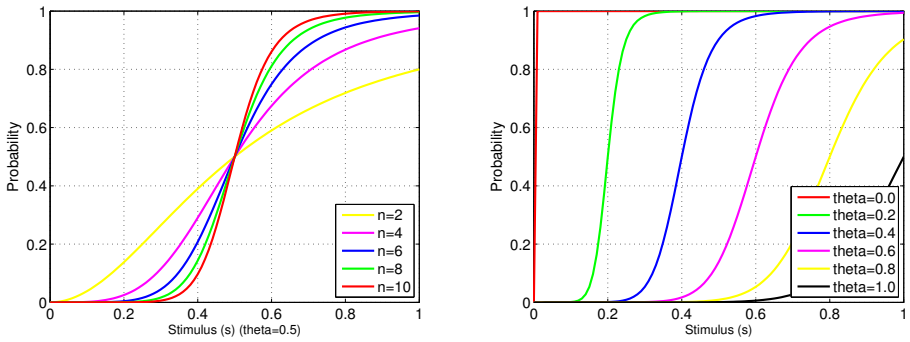
Our objective is to deploy a team of robots to achieve the desired distribution among various tasks, starting from an initial distribution, with no inter-agent wireless communication. We assume that every robot has no information about optimal transition rates k_{ij} in the task transition graph G. In the absence of the global information, each individual agent should estimate the transition rate using local environmental information. To handle this problem, we utilize a task probability function based on the response threshold model.

To explain the mathematical model, we assume that a robot $m(m = 1, ..., M)$ is given with collected task demands $s_{mi}(t)$ and the corresponding response threshold $\theta_{mi}(t)$ for task i $(i = 1, ..., N)$. Then the task probability that robot m responds to perform task i is determined by

$$P_{mi}(t) = \frac{s_{mi}(t)^n}{s_{mi}(t)^n + \theta_{mi}(t)^n} \tag{4.4}$$

with variable n $(n > 1)$, which determines the steepness of task probability. For $s_{mi}(t) < \theta_{mi}(t)$, the probability of engaging task performance is close to 0 and for $s_{mi}(t) > \theta_{mi}(t)$, this probability is close to 1. At $s_{mi}(t) = \theta_{mi}(t)$, $P_{mi}(t) = 0.5$.

Figure 4.1 shows the task transition probability depending on the choice of n and θ. Figure 4.1(a) shows that different values of n produce different probability given the same stimulus s. Large n has steep curves, and the task probability grows fast as the stimulus



(a) (b)

Figure 4.1: Task probability curves for varying n and θ values: (a) task probability for different values of n; and (b) Task probability for different values of θ. n represents the slope of curve and θ produces difference responses, given the same stimulus s.

increases. We use $n = 10$ for our experiments. Figure 4.1(b) shows that agents with smaller θ tend to respond to small stimulus values easily.

Based on the probability given in Equation (4.4), each agent decides to change the current task or not. A robot performing task i currently changes its task to task j if the probability of task j is the largest among all the possible tasks. In the aspect of an overall system, the task transition rate, $k_{ij}(t)$, can be estimated by

$$k_{ij}(t) = \frac{n_{ij}(t)}{n_i(t)} \tag{4.5}$$

where $n_{ij}(t)$ represents the number of robots currently performing task i and becoming to have the maximum probability for task j. That is, those robots change their task to task j. This value $k_{ij}(t)$ changes depending on the condition of each agent but will decrease as the overall system converges to the desired task distribution.

Our objective is to allocate a group of robots to achieve the desired distribution among various tasks. We assume that every agent has knowledge of the whole graph G, but without knowing the optimal transition rate. All individuals select a suitable task locally using local information without any communication aids among agents, with no clear picture of what is going on at the level of the overall system level. Repetitive and continuous task selection can lead to the desired performance closely.

If agents can obtain the global information about the environment, they can easily

choose tasks, just by using the fraction of each task among the whole tasks and the number of robots assigned to tasks. However, robots have difficulty in fetching the global information due to their limited sensing abilities and no communication aid. They should decide their own task based on their estimation of the environment. Here, using the task probability from Equation (2), the individual robot selects a task with the highest value. Then the task transition rate, $k_{ij}(t)$, may be estimated by counting the number of robots that has the maximum value of $P_{mj}(t)$ for task j among robots m currently performing task i. However, the information still cannot be fetched easily in the local environment.

4.1.3 History based information estimation

For estimating the needed information for task decision, we resented more improved approach inspired by the pheromone trails of insects for a memorization process to obtain a proper threshold value for each task. Pheromone named by P. Karlson and M. Lusher in the 1950s is a chemical media for delivering information by individuals of the same species. Without direct communication, ants, bees, and wasps coordinates the activities of individuals in the colony by depositing and sensing chemical markers in a shared physical environment. For example, ants initially wander randomly and upon finding food, return to their nest while laying down pheromone trails to guide other ants to food source. The ant colony optimization algorithm was proposed for searching an optimal path based on the behavior of ants seeking a path between their nest and a source of food [54] and applications to various dynamic scheduling problems were studied [146, 108, 120, 227, 213].

The characteristics of pheromone that are capable of enabling complex functions are mainly related with evaporation [55]. Evaporation represents that old information is disappearing as time passes due to the volatile tendency of pheromone. This characteristic enables agents to receive the most recent information and we want to use this tendency to update the response threshold for task.

The task demands and neighboring robots can be monitored to estimate the task probability. The recent information about the task demands and neighboring robots is stored in the history window of finite length. It is related to what tasks are highly demanded in a given environment and what tasks neighboring robots are doing. If the proportion of tasks is similar to or the same as the proportion of robots assigned to the tasks, the task allocation will be desirable. Each robot performs observation behavior periodically and only the most recent information is stored in the history queue. From the record, the robot estimates the

global task demands and the global robot distribution. The robot saves the local surrounding information with constant time interval to prevent storing duplicated information in history queue.

Estimation of task demand

For implementation of the pheromone memory based algorithm, each agent has two task queues, task demand queue and task supply queue, and stores the information of observing tasks and neighboring agents. In task demand queue, the types of detected tasks within the sensing range are stored and the task types of neighboring agents within the communication range are stored in task supply queue in order. Then, they estimate the global information approximately by using the fraction stored in queues.

If a robot m has information of the global task demand, the proportion of task demand for task i can be simply obtained with

$$\hat{s}_i = \frac{s_i}{\sum_{i=1}^{N} s_i} \tag{4.6}$$

where s_i is the task demand for task i. In the decentralized approach with no information about the global task demand, each agent should estimate the proportion of tasks by using the distribution of demands in the task demand queue as follows:

$$s_{mi}(t) = \frac{\sum_{l=1}^{L} task_{mi}^{D}(l)}{L} \tag{4.7}$$

where L is the length of task demand queue and m is the robot number. We assume that $s_{mi}(t)$ approximates the above $\hat{s}_i$ for task i. In addition, $task_{mi}^{D}(l)$ is 1 if the task type in the l-th queue of agent m is task i, otherwise it is 0.

Similarly, the proportion of robots performing task i, $x_i(t)$, can be estimated with

$$x_i(t) \approx \frac{\sum_{l=1}^{L} task_{mi}^{N}(l)}{L} \tag{4.8}$$

where $task_{mi}^{N}(l)$ is 1 if the neighboring agents of an agent m perform the task i in the l-th queue, otherwise it is 0. Each robot maintains two queues, the queue of task demand: $task_{m}^{D}$ and the queue of task types of observed neighboring robots, $task_{m}^{N}$. We assume that each robot can easily observe what tasks are being performed by the neighboring robots. In the

object foraging task to collect objects with a swarm of robots, the first queue monitors what objects have been observed by a given robot, and the second queue saves what tasks have been done by its neighboring robots.

Regulation of response threshold

According to the response threshold model [23, 212, 41], the threshold is updated depending on the performance of tasks. If a task is performed by an agent, it decreases the threshold value of that task and increases the thresholds of other unperformed tasks. The more agents perform a specific task, the lower the response threshold to this type of task, and vice versa. This threshold update process leads to the emergence of specialized agents who are more responsive to particular types of tasks with lower thresholds than the others with higher thresholds.

Base on the response threshold model, the individual robot updates its response threshold using the amount of task demand and the number of robots assigned to that task [118].

$$\theta_{mi}(t+1) = \theta_{mi}(t) - \eta(s_{mi}(t) - x_i(t)) \tag{4.9}$$

where η is a scaling factor to regulate the threshold over time and threshold θ_{ij} is constrained to the interval $[\theta_{min}, \theta_{max}] = [0, 1]$. If the proportion of task demand for task i is larger than the proportion of robots performing task i, threshold θ_{mi} is decreased to encourage more participation of robots on task i, and otherwise, threshold θ_{mi} can be increased.

Both parameters $s_{mi}(t)$ and $x_i(t)$ can be obtained from the whole set of task demands and robot states. In a decentralized approach, each robot could estimate the information using two queues to record local task demands and observed task types of neighboring robots. Then, Equation (4.9) is changed into:

$$\theta_{mi}(t+1) = \theta_{mi}(t) - \eta(\frac{\sum_{l=1}^{L} task_{mi}^{D}(l)}{L} - \frac{\sum_{l=1}^{L} task_{mi}^{N}(l)}{L}) \tag{4.10}$$

where L is the queue length, $task_{mi}^{D}(l)$ is given in Equation (4.7) and $task_{mi}^{N}(l)$ in Equation (4.8).

The thresholds update rule in Equation (4.10) indicates that if the task demand on task i is larger than the number of robots performing the task, an agent tends to specialize on the task by lowering its threshold and increases the probability of task performance. From

these repeated behaviors, each agent becomes to have a tendency to perform one specific task and this specialization reduces task changes within the overall system, producing the expected division of labor.

4.2 Analysis

In this section, we consider the stability of our proposed model.

4.2.1 Convergence to equilibrium state

In the equilibrium stat that task distribution is equally to task demands, the balancing condition is needed as

$$k_{ij}\bar{x}_i = k_{ji}\bar{x}_j, \quad \forall (i,j) \in E \tag{4.11}$$

We now show that our task allocation model described in Equation (4.1) has a stable equilibrium point that satisfies the desired target distribution $\bar{x}_i(t)$ for task i, for $i = 1, \ldots, M$.

$$\frac{dx_i(t)}{dt} = \sum_{\forall j|(j,i)\in E} k_{ji}x_j(t) - \sum_{\forall j|(i,j)\in E} k_{ij}x_i(t) \tag{4.12}$$

We argue that the system described by the above equation for $i = 1, \ldots, M$ (M is the number of all tasks) for all $(i,j) \in E$ with condition Equation (4.11), the response threshold updating rule in Equation (4.9) and the task selection function in Equation (4.4) will converge almost surely to $\bar{x} = [\bar{x}_1, \ldots, \bar{x}_M]^T$. Consider the following Lyapunov function given by

$$V = \sum_{i=1}^{M} \frac{\bar{x}_i}{2}\left(1 - \frac{x_i}{\bar{x}_i}\right)^2 \tag{4.13}$$

and the time derivative of Equation (4.13) is

$$\begin{aligned} \frac{dV}{dt} &= \sum_{i=1}^{M} \frac{(x_i - \bar{x}_i)}{\bar{x}_i}\frac{dx_i}{dt} \\ &= \sum_{i=1}^{M} \frac{(x_i - \bar{x}_i)}{\bar{x}_i}\left(\sum_{\forall j|(j,i)\in E} k_{ji}x_j - \sum_{\forall j|(i,j)\in E} k_{ij}x_i\right) \end{aligned} \tag{4.14}$$

Then, by Equation (4.11), the above equation can be changed into

$$\begin{aligned}\frac{dV}{dt} &= \sum_{i=1}^{M} \frac{(x_i - \bar{x}_i)}{\bar{x}_i} \left(\sum_{\forall j|(i,j)\in E} \frac{\bar{x}_i}{\bar{x}_j} k_{ij} x_j - \sum_{\forall j|(i,j)\in E} k_{ij} x_i \right) \\ &= \sum_{i=1}^{M} \sum_{\forall j|(i,j)\in E} \left(\frac{x_j}{\bar{x}_j} - \frac{x_i}{\bar{x}_i} \right) k_{ij} (x_i - \bar{x}_i) \\ &= \sum_{\forall j|(i,j)\in E} \left(\frac{x_j}{\bar{x}_j} - \frac{x_i}{\bar{x}_i} \right) \phi_{ij}\end{aligned} \tag{4.15}$$

where $\phi_{ij} = k_{ij}(x_i - \bar{x}_i)$.

We can set up the convergence condition such that ϕ_{ij} has an opposite sign to $(x_j/\bar{x}_j - x_i/\bar{x}_i)$. Thus, if $x_j/\bar{x}_j > x_i/\bar{x}_i$, then $\phi_{ij} < 0$ and similarly, if $x_j/\bar{x}_j < x_i/\bar{x}_i$, then $\phi_{ij} > 0$. That is,

$$\frac{dV}{dt} = \sum_{\forall j|(i,j)\in E} \left(\frac{x_j}{\bar{x}_j} - \frac{x_i}{\bar{x}_i} \right) \phi_{ij} < 0 \tag{4.16}$$

Thus, the time derivative of the Lyapunov function evaluates to negative. In addition, consider when all $\phi_{ij} = 0$ or when $x_i/\bar{x}_i = x_j/\bar{x}_j$ for all i, j, the time derivative of the Lyapunov function is always non-positive, and then the system converges almost surely to the desired distribution, $\bar{x}_i$. If the number of agents are totally conserved,

$$\sum_{i=1}^{M} x_i = 1 \tag{4.17}$$

and there are two types of tasks $(M = 2)$, then Equation (4.15) always satisfies the following condition

$$\frac{dV}{dt} = \sum_{\forall j|(i,j)\in E} \left(\frac{1 - x_k}{1 - \bar{x}_k} - \frac{x_i}{\bar{x}_i} \right) k_{ij} (x_i - \bar{x}_i) < 0 \tag{4.18}$$

The task transition rate k_{ij} has a positive value, $k_{ij} > 0$, and if $(x_i - \bar{x}_i) > 0$, then $(\frac{1-x_i}{1-\bar{x}_i} - \frac{x_i}{\bar{x}_i}) < 0$ because $\frac{1-x_i}{1-\bar{x}_i} < 1$ and $x_i/\bar{x}_i > 1$. Similarly, if $(x_i - \bar{x}_i) < 0$, then $(\frac{1-x_i}{1-\bar{x}_i} - \frac{x_i}{\bar{x}_i}) > 0$.

If there are more than two types of tasks, $(M \geq 3)$, Equation (4.15) might produce a positive value. However, if we assume that task assignment is regulated between the agents

performing task i and the other agents not performing task i, then it satisfies the convergence condition

$$\frac{dV}{dt} = \sum_{\forall j|(i,j)\in E} \left(\frac{1 - \sum_{k=1,k\neq i}^{M} x_k}{1 - \sum_{k=1,k\neq i}^{M} \bar{x}_i} - \frac{x_i}{\bar{x}_k} \right) k_{ij}(x_i - \bar{x}_i) < 0 \tag{4.19}$$

In our task, each agent has an estimated information, $s_{mi}(t)$, for the desired task distribution, $\bar{x}_m$. Based on this value, if more agents are assigned to task i than the desired proportion, some agents currently performing task i should be changed to other tasks including task j. Then sequentially, it can be assumed that task is regulated between two groups, performing task j and not performing task j. By this way, Equation (4.15) converges asymptotically to $\bar{x} = [\bar{x}_1, \ldots, \bar{x}_M]^T$.

4.2.2 Convergence of threshold update

We provide a proof that the rule for the response threshold can lead to convergence to the equilibrium of task allocation. In order to regulate threshold $\Theta = [\theta_1, \theta_2, ..., \theta_{N-1}, \theta_N]'$ for N tasks in an adaptive process, we design our adaptation laws according to the steepest descent method [78]. The iterative equation of Θ can be represented as

$$\Theta(t+1) = \Theta(t) + \eta P(t) \tag{4.20}$$

or, equivalently,

$$\triangle\Theta(t) = \Theta(t+1) - \Theta(t) = \eta P(t) \tag{4.21}$$

where the positive scalar η is the learning rate, which determines the convergence speed and the vector $P(t)$ represents a search direction.

We desire to balance the proportion of task demand and the proportion of robots working on the task. We define a cost function $J(\theta(t)) = [J_1, J_2, ..., J_N]'$, which is selected as a nonnegative function with the summation of squared errors

$$J_k(\theta(t)) = \frac{1}{2}(\hat{s}_k(t) - x_k(t))^2 \tag{4.22}$$

where $\hat{s}_k$ is the rate of task demand for the k-th task and x_k is the population fraction of robots for the k-the task for $k = 1, ..., N$ (see Equation(4.6)-(4.8)). Then the cost function

decreases at every iteration,

$$\triangle J(\theta(t)) = J(\theta(t+1)) - J(\theta(t)) < 0 \tag{4.23}$$

or

$$J(\theta(t+1)) < J(\theta(t)) \tag{4.24}$$

Thus the key idea is to choose an appropriate search direction $P(t)$ such that Equation (4.24) is satisfied for sufficiently small learning rate η. Consider the first-order Taylor series expansion of $J(\theta(t))$ about $\theta(t)$, then we have

$$J(\theta(t+1)) = J(\theta(t) + \triangle\theta(t)) \approx J(\theta(t)) + g^T(t)\triangle\theta(t) \tag{4.25}$$

where $g(t) = [g_1, g_2, ..., g_N]'$ and $g_k(t) = (\hat{s}_k(t) - x_k(t))$ represents the gradient of $J_k(\theta(t))$ evaluated at $\theta(t)$ and in order to satisfy Equation (4.24), the last term on the right-hand side in Equation (4.25) should be negative. We choose

$$P(t) = -g(t) \tag{4.26}$$

so that, from Equation (4.21)

$$g^T(t)\triangle\theta(t) = \eta g^T P(t) = -\eta g^T(t) g(t) < 0 \tag{4.27}$$

Consequently, Equation (4.20) becomes, which is the same with Equation (4.9),

$$\Theta(t+1) = \Theta(t) - \eta g(t) \tag{4.28}$$

which leads to Equation (4.9), since $g_k(t) = \hat{s}_k(t) - x_k(t)$. By the central limit theorem, the average of samples can approximate the expected rate of task demand or rate of robots working on the task. We can use Equation (4.10) with the observed samples.

We run the foraging task to collect objects with a swarm of robots. Every experiment is repeated for 10 independent runs. At the beginning, all robots are initially assigned to the $red\ task$ (clearing red-colored objects) and the initial values of all thresholds are randomized to ensure that each robot is not predetermined for a specific task. Task demand is represented as the proportion of specific objects among the whole population of objects,

and if 50% of objects are red, then we want that 50% of the robots perform $red\ task$ for clearing red-colored objects. The performance can be determined by the proportion of robots performing each task.

4.3 Simulation results

4.3.1 Robot behaviors

Each robot obtains locally-surrounding information with its own visual sensor and it can change the task depending on the task transition condition. A variety of robot behaviors is shown in Figure 4.2. Differently to the experiment in Chapter 3, the information about the task type of robots and the color of objects are stored in two queues, separately. We assume that in simulation, robots discriminate two types of objects and see what tasks neighboring robots are performing with the help of the vision camera. Robots distinguish objects and other robots with vision cameras; they can detect the color of objects as well as the color of robots. Each robot has its own task and it emits red- or green-colored light depending on what task it is currently taking. It can see what task its neighboring robots within its visual vicinity are doing.

Each robot has color-preference for foraged objects, either green or red objects. It means two types of tasks are available for each robot. Robots update the color of detected objects in the task demand queue and the task types of neighboring robots in the task supply queue (see subsection 4.1.3). The queues record the most recent information about objects and neighboring robots within a fixed time length. As a result, each robot manages the information of what objects have been observed recently and what tasks neighboring robots in its surrounding environment have completed. Each robot updates the response threshold for tasks and computes the task probability for each task. Depending on the value, the robot can switch its current task to another. The length of both history queues are set to $L = 20$ and the learning rate for the threshold is set to $\eta = 0.015$ (see Equation (4.28)).

4.3.2 Results with fixed task demands

In the first experiment, the task demand of $red\ task$ is set to 10%, 30%, 50%, 70%, and 90%, and $green\ task$ is set to 90%, 70%, 50%, 30%, and 10%, respectively. This means

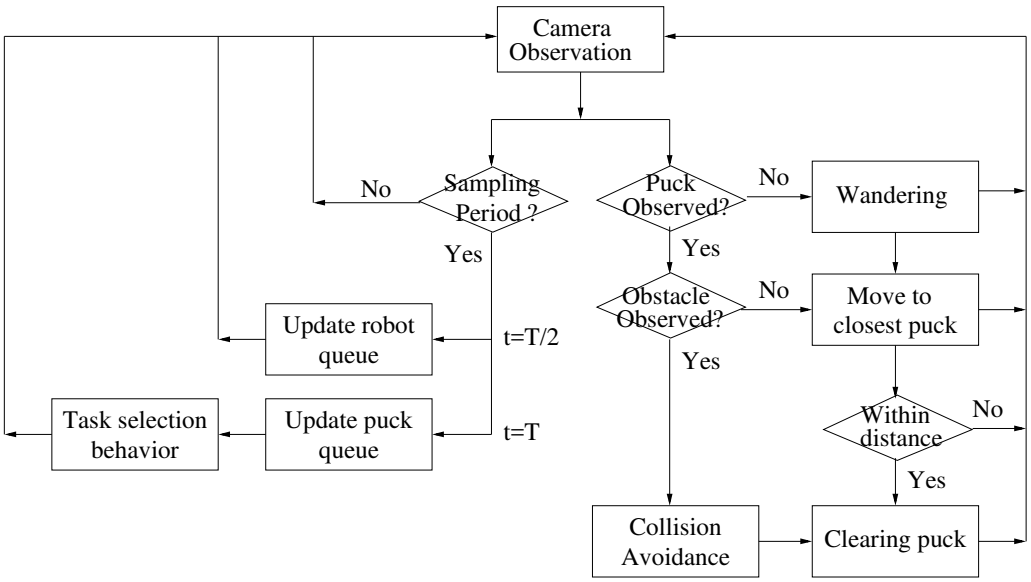
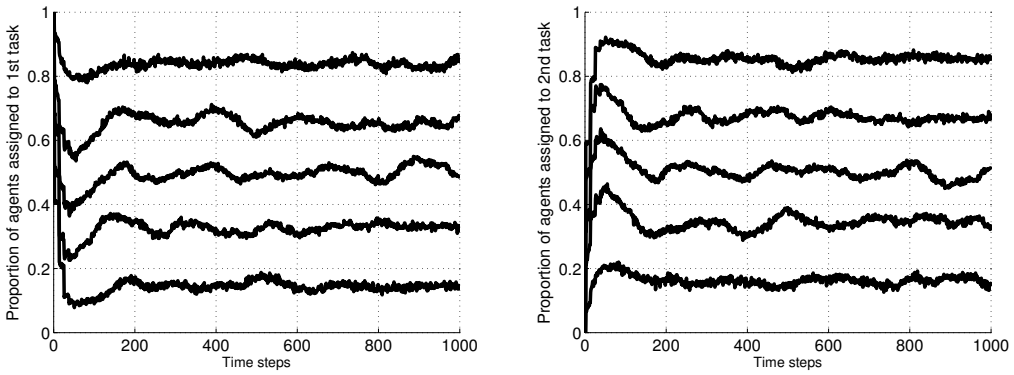


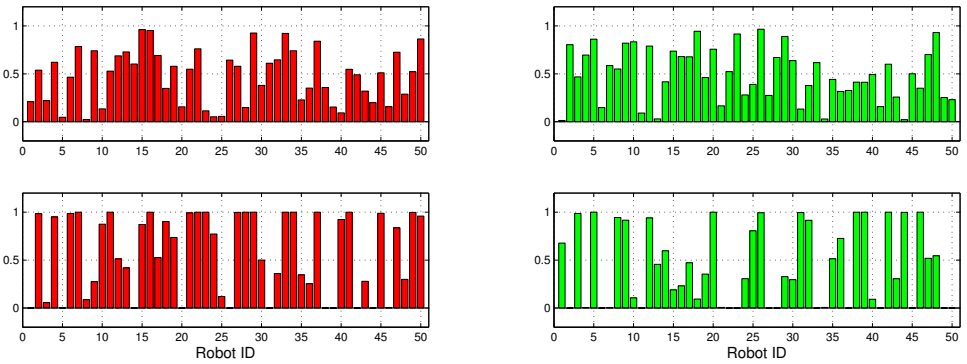
Figure 4.2: State transition of robot behaviors.



(a) (b)

Figure 4.3: Proportion of robots assigned to two different tasks; clearing red-colored objects ($red\ task$) and green-colored objects ($green\ task$): (a) the proportion of robots performing the $red\ task$; and (b) the proportion of robots performing the $green\ task$, while task demands are set to 10%, 30%, 50%, 70% and 90% for the $red\ task$ and 90%, 70%, 50%, 30%, and 10% for the $green\ task$, respectively.

that 5, 15, 25, 35, and 45 robots should be assigned to clear red objects and 45, 35, 25, 15 and 5 robots should be assigned to clear green objects. The progress of the proportion



(a) (b)

Figure 4.4: Thresholds change for two tasks in robots: (up) initial state and (down) final state; (a) the $red\ task$ threshold; and (b) the $green\ task$ threshold.

of robots assigned in each task is shown in Figure 4.3. First, all robots perform the $red\ task$, but after some time passing, the swarm is split for performing different tasks as the same proportion with the desired task demands. The swarm reacts properly to the changes in the proportion of task demands by switching a proper number of robots from $red\ task$ to $green\ task$. Even with the limited sensing range, the proportion of robots assigned to each task reaches a stable level after a while. There is some gap between the proportion of task demands and that of robots assigned to the tasks when the proportion difference of task demands is large.

This tendency is due to the specialization characteristics of the response threshold model. Specialized robots move longer distances without changing task. If the portion of some objects in demand is smaller than the others, the probability for detecting the minor group of objects and robots is relatively smaller and then more robots performing the minor task may be needed than the actual task proportion.

Figure 4.4 shows the change of thresholds for two tasks, which are assigned randomly at the beginning. At the end of the simulation, some robots have thresholds that are equal to the maximum threshold, θ_{max} or minimum threshold, θ_{min}, but other robots still have thresholds between $[\theta_{min}, \theta_{max}]$. This means that some robots are strongly specialized to tasks and others are softly specialized to tasks. Task allocation within individual agents can be explained by assuming that $red\ task$ specialists have low thresholds for $red\ task$ and high thresholds for $green\ task$, whereas $green\ task$ specialists have opposite thresh-

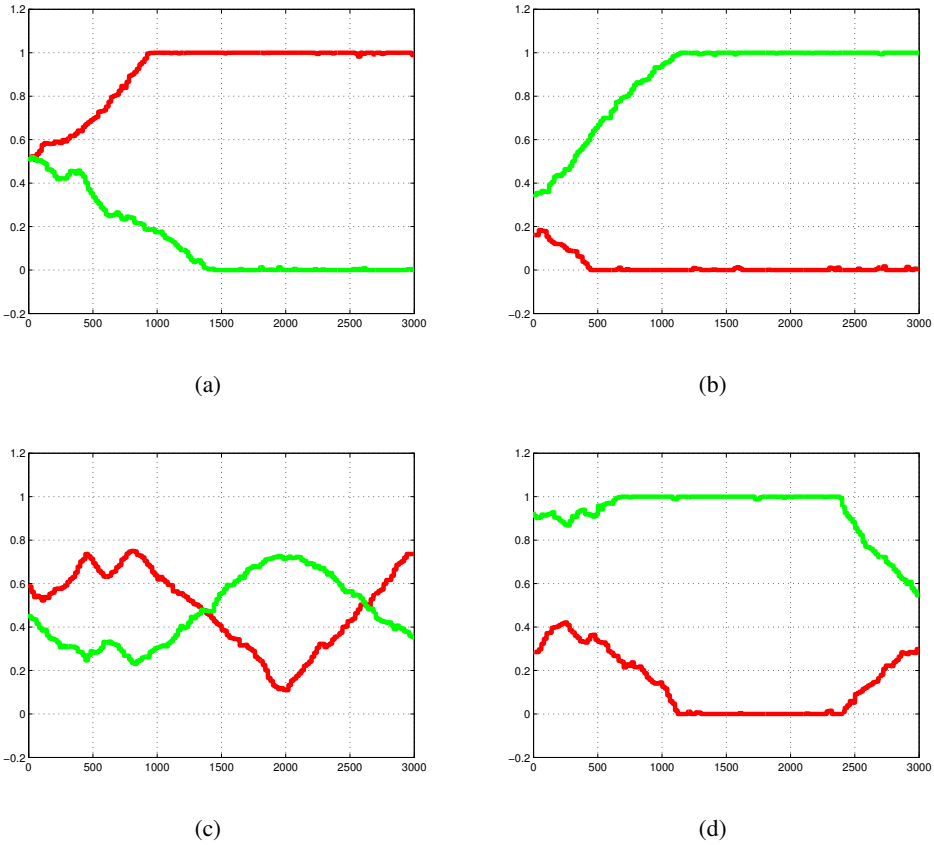


Figure 4.5: Various cases of specialized tendency: (a) strongly specialized to $green\ task$; (b) $red\ task$; (c) Softly specialized to both tasks; and (d) strongly specialized to $red\ task$ changes to softly specialized to $red\ task$ to increase the probability to green task and decrease the probability of red task.

old rates. As soon as the number of $green\ task$ specialists decreases, the associated task stimulus will increase until it exceeds the corresponding thresholds of $red\ task$ specialists. Consequently, some $red\ task$ specialists will perform $green\ tasks$ until the corresponding stimulus falls below their thresholds. This threshold update process provides the emergence of specialized agents more responsive to particular task types.

There are two types of specialization in a strong or soft manner. Strongly specialized individuals perform only one or a few activities. However, dynamic real-time environmental changes or the fluctuation over the portion of members within the colony may need an adaptive task allocation ability for the colony survival. In this case, softly specialized indi-

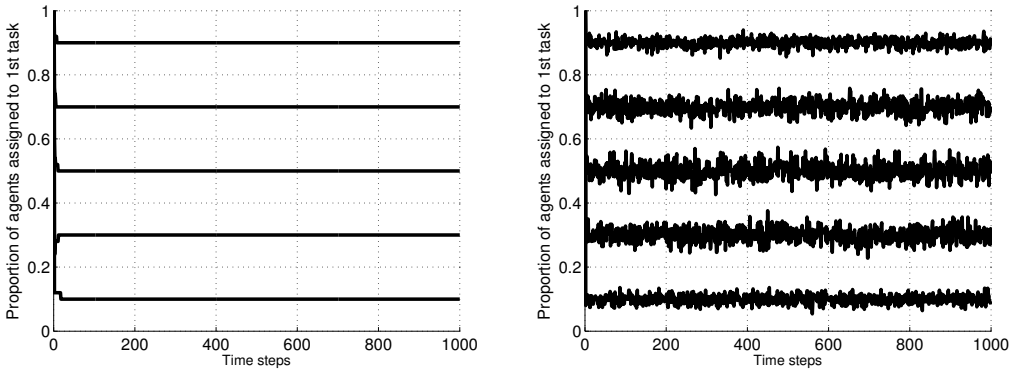


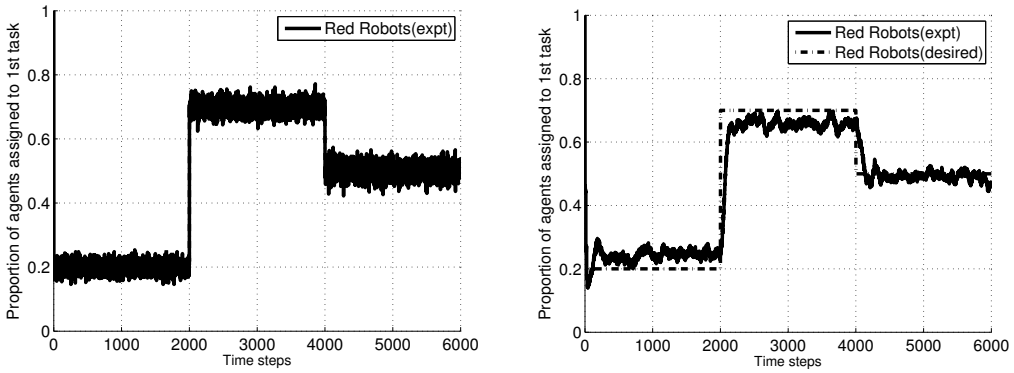
Figure 4.6: Proportion of robots assigned to $red\ task$: (a) all information is given to robots due to unlimited sensor range; and (b) constant task transition rate is applied in task transition model.

viduals perform several activities, but tend to join the activities most needed by the group by changing their tasks flexibly. For example, some ant species specialized in cleaning can carry out foraging that they would normally not perform, if the number of foraging ants decreases. This mechanism can be explained well in our experimental results.

These various tendencies are shown in Figure 4.5, which shows four types of threshold progress patterns. Two figures in the upper side show the result of strongly specialized to the $green\ task$ and $red\ task$, respectively. In the lower side, the left figure shows agents softly specialized to both tasks. The thresholds are changed depending on the situation. The right figure shows the case that the agent is strongly specialized to $red\ task$, but its $red\ task$ threshold is increased and the $green\ task$ threshold is decreased to respond and perform the $green\ task$.

The same experiments were repeated in Figure 4.6, when the information about task types of robots and objects can be calculated with unlimited sensor range in our method (Figure 4.6(a)), and robots use the constant task transition rate in the task transition model [79] (see Equation 4.1) calculated by using the global information (Figure 4.6(b)).

Table 4.1 shows the comparison of the total number of task change occurrences. In both cases, the proportions of robots assigned to the $red\ task$ result in the convergence to the desired task distribution with different variance levels, and continuous task changes with more fluctuation can be observed with a constant transition rate in the task transition



(a) (b)

Figure 4.7: Proportion of robots assigned to $red\ task$ in a group with changes in task demands: (a) constant task transition rates; and (b) estimated information.

model. If we count the number of task changes of swarm robots, our proposed method has more advantage in terms of task changes. It also handles the task allocation with only local information, without global communication among robots.

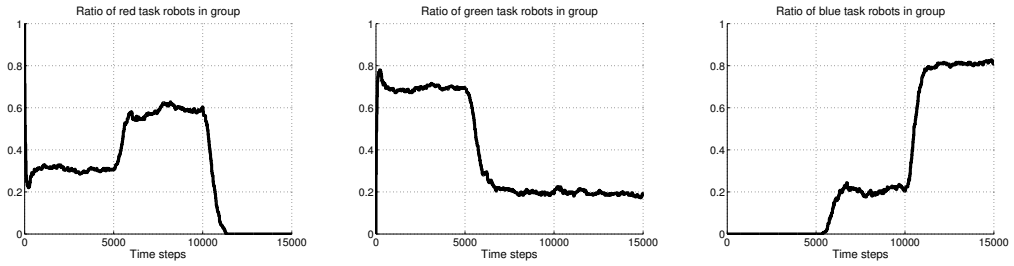
4.3.3 Results with changes in task demands

For the next experiments, we tested variation of task demands in time course. Figure 4.7 shows the results with task demand changes. The tasks demands of two tasks, (*red task*, *green task*), were initially set to 20% and 80%, respectively. At time step 2,000, the task demands were then changed to 70% and 30%, and at time step 4,000, the demands were again changed to 50% and 50% respectively. When we test the task allocation with more tasks,

Figure 4.8 shows the results of three types of tasks: $red\ task$, $green\ task$, and $blue\ task$. Task demands were set to 20%, 80%, and 0% at first. At time step 5,000, the task demands were changed to 60%, 20%, and 20%, then at time step 10,000, the demands

Table 4.1: Overall comparison for the number of task changes.

	Figure 4.6(a)	Figure 4.3(a) (Our proposed method)	Figure 4.6(b)
Count	5	106	8,240



(a) (b) (c)

Figure 4.8: Proportion of robots assigned to three tasks with changes in task demands: (a) $red\ task$; (b) $green\ task$; and (c) $blue\ task$.

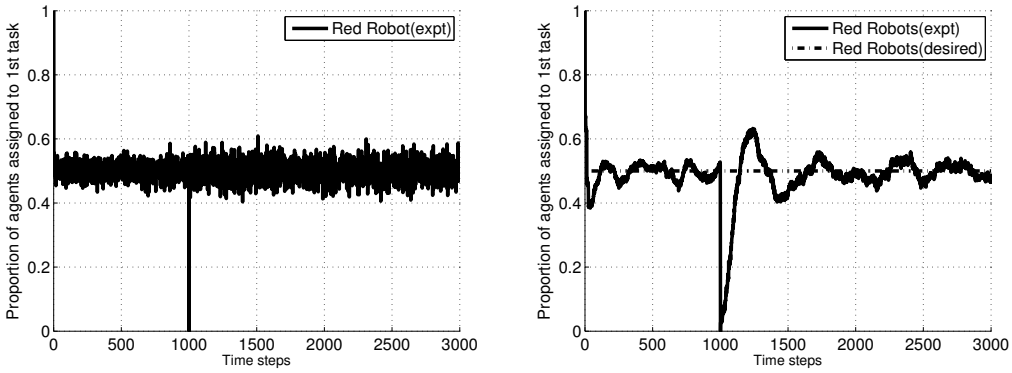
were changed to 0%, 20% and 80% respectively. As the figures demonstrate, the swarm immediately reacts with the changes of task demands. By changing the task demands, the proper fractions of robots were re-assigned and the desired distribution was obtained in the overall system level.

4.3.4 Results with changes in number of agents

In the next experiment, we investigated the adaptability of the swarm to the change in the number of robots. Task demand of each task was initially set to 50% and all agents assigned to $red\ task$ were suddenly removed from the swarm at time step 1,000. Figure 4.9 represents the behavior of the swarm in response to the change. The swarm reacts to the abrupt change by switching the task of some robots. The mechanism behind this reaction is as follows: when a number of robots are removed from the system, the robots with small threshold value for $red\ task$ start to work on that task by changing their thresholds values into lower level. Our proposed method shows an adaptive ability to the environmental change, only using local estimation of the task demands and task types of neighboring agents.

4.4 Application to Factory Domain Applications

Using data from various experimental situations, changes in experimental environments and parameters in the proposed algorithm, we analyze the performance for a practical task in factory domain applications.



(a) (b)

Figure 4.9: Proportion of robots assigned to $red\ task$: (a) with constant transition rate; and (b) with local estimation. The demands of the two tasks were initially given as same with 50% and 50%. At time step 1,000, all agents assigned to red task were removed from the swarm.

4.4.1 Task description

Among various tasks, truck painting problem ([162, 163]) is an appropriate task that the specialized tendency is needed for improving performance due to the extra costs for task changes. Each truck can require any paint color, and the colors required by trucks are not known in advance. Only one color can be set for a paint booth at a time, and some amount of remaining paint will be wasted each time a booth changes its current color to paint the different color. Thus, frequent color changes in a paint booth incur both time and material costs. A good system to assign trucks to paint booths thus reduces such color changes with maximizing the total number of painted trucks in all the paint booths.

The basic scheduling paradigm is shown in Figure 4.10. A paint booth is an agent, and multiple agents compete to perform task of painting truck. One truck arrives per every unit time and is assigned to an agent using its bidding or choice process. If a paint booth machine is assigned more than one task, each task is stored in its queue to be performed one at a time in order. The length of each machine's queue is set to three and if a machine's queue is full, that machine does not participate in the bidding process until there is a empty space in the machine's queue. In the event that no booth makes a bid (because all of the paint booths are either broken or have full queues), the truck is waiting in a temporary storage and assigned when there is a possible booth.

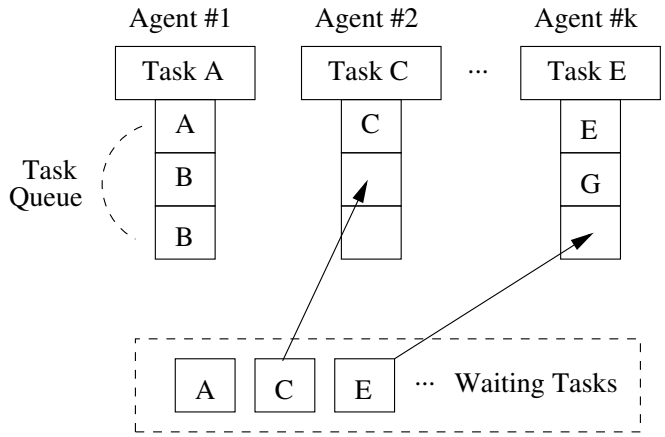


Figure 4.10: Paradigm of dynamic scheduling model in truck painting problem.

4.4.2 Modified task selection method

The objective of the dynamic scheduling algorithm is to minimize the number of task changes of the overall system while it maintains the desired performance level such as throughput; the number of tasks completed during a given time span. In the previous sections, we showed that history based approach could be an effective method. In this section, we want to implement based on the history of recently processed tasks.

Each individual agent has a limited and constant-sized memory queue (task history) that stores the recent processed tasks by itself. The appropriated thresholds values are obtained using the information in its own task history. The schematic diagram of a task history queue used for each agent is shown in Figure 4.11. This model is composed of three parts, task history queue, main booth, and task queue. Each agent currently performs one task and the waiting tasks are stored in queue in an assigned order. In addition, each individual agent maintains a limited, constant-sized task history queue storing its recently processed tasks. Thus, we use the moving average of the information in task history to update the response threshold values. The basic concept is the same with the pheromone. Old information in the task history is excluded automatically, just as pheromone disappears over time, which makes the system adaptive to environmental change.

The pheromone evaporation strategy is obtained by computing a weighted moving average of the last N values, which is the length of the task history queue. That is, we count the number of tasks stored in a task history queue and update the threshold value for each task in direct proportion to that value. Here, a drop of pheromone is represented as memory ele-

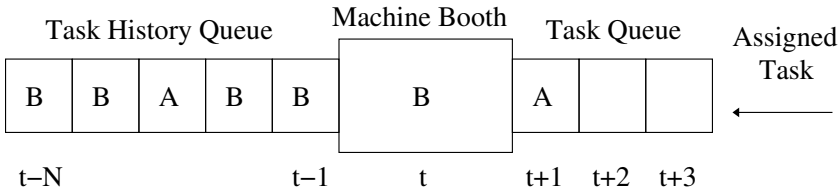


Figure 4.11: Schematic diagram of a task queue. This agent is currently processing a task of type B, and a task of type A is waiting in its queue. The agent last processed a task of type B, and it has four tasks of type B and one task of type A in its task history queue.

ment in a history of tasks and the list of elements in a queue is pheromone on a trail. Then the pheromone information $pheromone_{kj}$ of agent *k* for task *j* is estimated as follows:

$$pheromone_{kj} = \frac{1}{N} \sum_{l=1}^{N} w_l \cdot task_{kj}^{l} \tag{4.29}$$

where N is the length of task history queue and $task_{kj}^{l}$ is calculated by task in the l-th task history of agent *k*. If the *l*-th task is *j*, then $task_{kj}^{l} = 1$, otherwise, it is 0. The weight w_l is a scaling factor in the task history. More recent processed tasks can have higher weights, or the system can have a uniform distribution of weights across a given number of tasks [124].

The pheromone is computed over the last N values; older values are not included, which emulates the evaporation of the pheromone. Then the threshold values are updated in direct proportion to that average, as in Equation (4.30).

$$\theta_{kj}(t) = \frac{pheromone_{kj}}{\sum_{i=1}^{M} pheromone_{ki}} \tag{4.30}$$

where M is the number of all possible tasks. Thus, the probability $P_{kj}(t)$ is given by

$$P_{kj}(t) = S_j(t)^2 + \theta_{kj}(t)^2 \tag{4.31}$$

where the stimulus S_j is proportional to the amount of time the job has been waiting for assignment to a booth. The outputs from Equation (4.31) are compared among the booths, and the task is assigned to the booth with the highest value.

The threshold updating method differs slightly in Equation (4.4). However, we can describe it in a similar form with the original approach. If we assume that the weight

parameter w_l is 1. If a truck j of color j is completed in booth k, then booth k updates its task history queue by adding color j and deleting the oldest information. Then the threshold value θ_{kj} for color j is updated as follows:

$$\theta_{kj}(t+1) = \theta_{kj}(t) - 1 \tag{4.32}$$

and the threshold value for color m of a deleted task m in task history is increased:

$$\theta_{km}(t+1) = \theta_{km}(t) + 1, \quad \forall m \in 1, 2, ..., M \tag{4.33}$$

This means that the threshold values of all booths are regulated in direct proportion to the number of processed tasks in their individual histories. The basic concept of pheromone computation is counting the number of tasks recently processed by each agent.

In our algorithm, we need a rule for breaking ties. If more than two agents make the same highest bid, the one with the shortest task completion time will be selected, as determined by the following equation.

$$P'_k(j) = \frac{1/\triangle T_k(j)}{\sum_{i=1, i\neq k}^{n} 1/\triangle T_i(j)} \tag{4.34}$$

where $\triangle T_k(j)$ is defines as the time until truck j is painted in booth k, as determined by the following equation:

$$\triangle T_k(j) = qt^{proc} + nt^{setup} + t^{working} \tag{4.35}$$

where q is the number of waiting trucks in the current queue of booth k, t^{proc} is the time required to paint one truck, n is the number of requiring color changes for the waiting trucks, t^{setup} is the time required for a color change, and $t^{working}$ is the remaining time to finish the currently painted truck in booth k. In this way, the agent with a smaller queue length has a higher probability of taking a new task. If there is still more than one competing agents, the task is randomly assigned to one of competing agents.

4.4.3 Comparison with other conventional methods

We detail a comparison between our history based approach, market-based approach, and insect-inspired model approach on the task allocation problem for parallel multi-purposed agents using the problem of assigning trucks to paint booths.

Before starting the simulation, we set the system parameters for each algorithm. Cicirello and Smith (2001) compared Cicirello's, Campos, and Morley's approaches. We used the same parameter settings for each method. In Morley's algorithm (called Market), K=91, C=1791, and L=4. In Cicirello's algorithms (called R-Wasps), the parameters are set to $\delta_1 = 100$, $\delta_2 = 10$, and $\delta_3 = 1.05$, and the initial thresholds θ_{k,c_j} are all set to θ_{min}, with $\theta_{min} = 1$ and $\theta_{max} = 10,000$. In Campos's algorithm (called ABA), α=617.188, β=4.66797, ξ=7.85156, and ϕ=17.7344.

Our pheromone memory approach needs two kinds of parameters. One is N, the length of the task history queue, and another is w_l, the weight parameter for the task history queue. To analyze the effect of each parameter, we use three different parameter settings. The first one (called History(1)) is set to N = 1, and another (called History(2)) is set to $N = 10$ both with a uniform distribution of weights, that is w_l is 1 for all l = 1,...,N. The third one (called History(3)) is set to N=10 with the weight parameter related to the square of the order in the task history queue. From the last stored task, the weight is set to N^2, $(N - 1)^2$, ... ,2^2, 1^2.

To compare the performance, we mainly measured on the average number of state changes occurrences (setup process) in all the booths, which is related to the wasted time and material costs. We also measured the average needed time per truck (cycle time), the average number of painted trucks (throughput), and the average length of the occupied task queue (queue length) for each agent. For all the experiments, we ran the simulation for 1,000 time steps and repeated 100 independent runs to calculate the averaged performance of each algorithm. Furthermore, we measured the performances of all the algorithms after 100 time steps from the start to remove the effect of the initial conditions.

Results with original experiment environments

In the first experiment, we evaluate our model using the same environments as Morley‘s original problem: seven truck paint booths and 14 paint colors. All paint booths are initially set for a specific color chosen at random and have three limited queues for trucks. Each truck arrives at a rate of one per unit time (here, a minute), the painting time for one truck is 5-minute, and the time needed for a color change is 1-minute. Each truck can be assigned

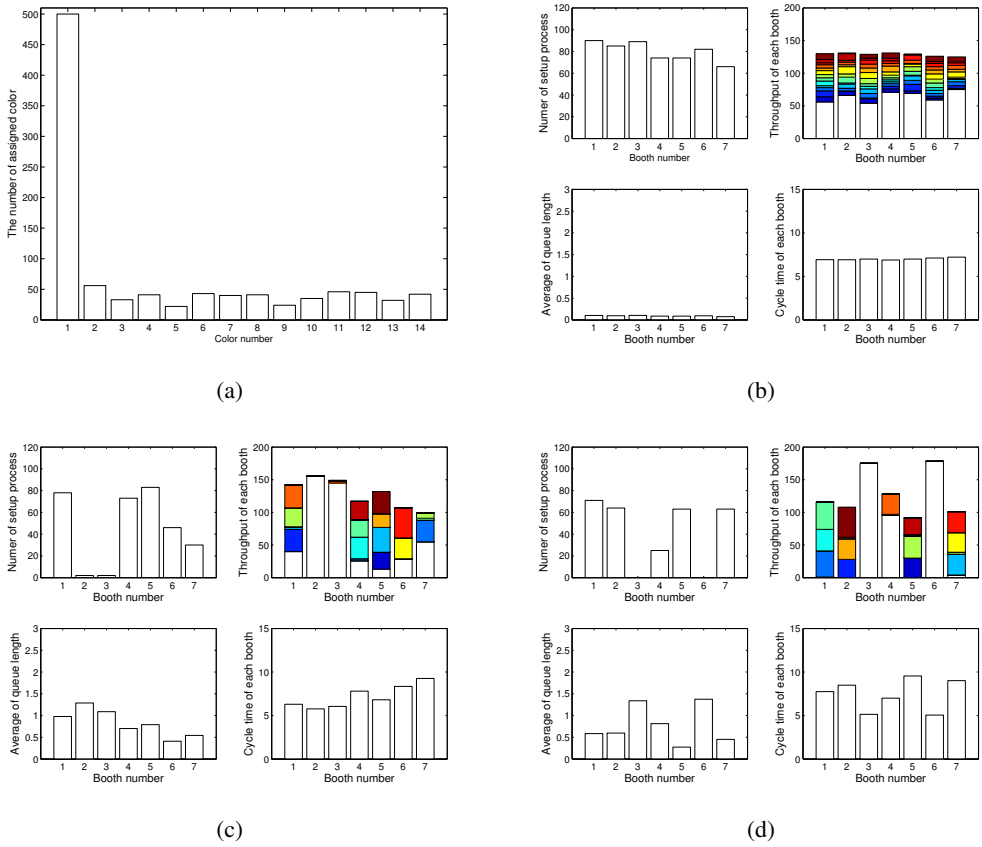


Figure 4.12: Distribution of assigned truck colors (a) and results for the first experiment (b)-(d); (a) Approximately 50% of the trucks require one specific color (color 1), and the other 50% require colors randomly chosen from among the other 13 colors; (b), (c), and (d) show results from the Market, ABA, and R-Wasps algorithms, respectively, and each figure comprises the total number of state changes among all agents, throughputs for each agent, average length of occupied queues, and cycle times, that is the average consumed time between the previously processed truck and the currently processed truck. The various colors in the throughputs figures represent the sum of tasks processed by each agent.

to any of the 14 colors, and the required colors are not known in advance. Approximately 50% of the trucks require one specific color, and the other 50% require colors are drawn randomly from among the other 13 colors.

The experimental results of the market-based approach and the two insect-inspired approaches are shown in Figure 4.12, and the results of our proposed algorithm inspired by

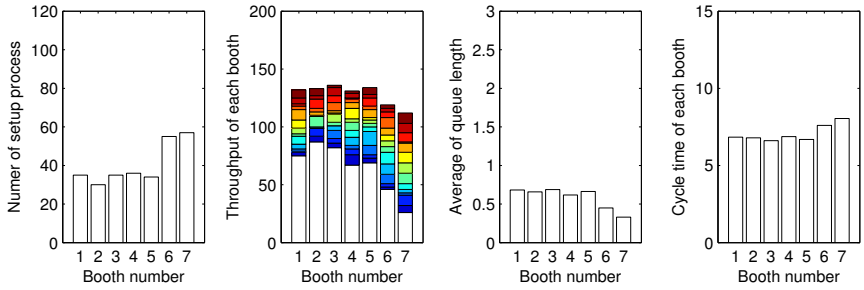
Table 4.2: Comparison results from the original experimental environments.

	Num. of change	Cycle time	Throughput	phQueue length
Market	559.30±11.99	7.00±0.00	900.10±0.74	0.10±0.00
ABA	342.70±13.52	7.12±0.06	901.00±1.97	0.95±0.04
R-Wasps	284.20± 8.95	7.33±0.08	900.00±2.50	0.78±0.12
History(1)	269.80±12.44	7.11±0.06	900.60±3.06	0.57±0.05
History(2)	291.80±9,81	7.39±0.08	899.60±2.32	0.90±0.10
History(3)	294.50±8.80	7.37±0.06	899.80±3.94	0.92±0.08

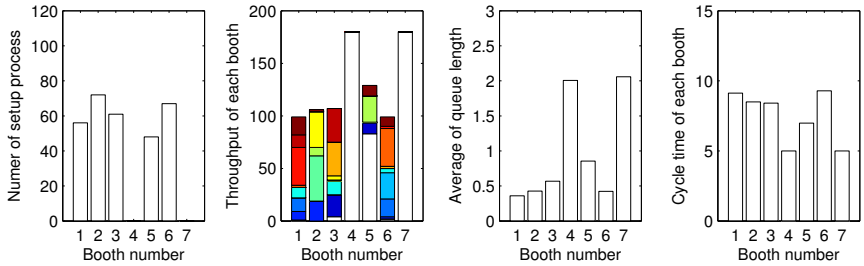
pheromone memory are shown in Figure 4.13. Figure 4.12 shows the distribution of assigned truck colors and the performances of the Market, ABA, and R-Wasps algorithms, respectively. In (a), it shows that one specific color (color 1) captures 50% of all the colors. Because the simulation ran for 1,000 time steps and the truck is arrived at a rate of one per unit time, there are almost 500 trucks assigned to one color. In (b)-(d), various colors in the throughput figures of each algorithm represent different tasks processed by an agent, and particularly, the white color in the throughput graph represents the task that accounts for 50% of all assigned tasks.

Most agents process several tasks, but the results differ among the algorithms. On the first hand, in the market-based algorithm, all agents equally process almost all kinds of tasks, so the average lengths of the occupied queues are low and state changes occur frequently in all agents. Cycle times of booths are also almost same. On the other hand, the results of the insect-inspired algorithms show quite different patterns with the market-based algorithm. In the ABA and R-Wasps algorithms, some agents process one kind of task almost exclusively. For example, in the ABA algorithm, agents 2 and 3 mainly process one task and show few state changes. However, because of the waiting time in the queue, the average length of occupied queues and cycle time increase compared with the market-based algorithm. At the same time, in the ABA and R-Wasps algorithms, the other five booth agents process all the other 13 colors, and all state changes occur among those agents. Those results can be explained using agent specialization characteristics in an insect-based approach. Some agents are specialized for a specific single color that composes a huge portion of the tasks, which reduces not only the state changes of the specialized agents but also the total number of state changes among all agents. The specialization ability of R-Wasps is better than that of ABA. The color 1 is almost never processed in other agents than the specialized agents 3, 4, and 6.

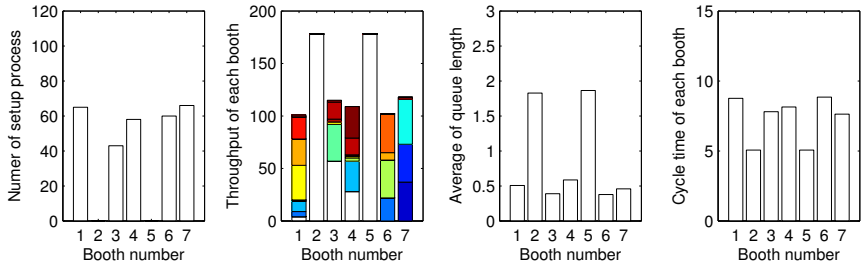
Figure 4.13 shows the performances of the pheromone memory algorithms. In His-



(a)



(b)



(c)

Figure 4.13: Results of the pheromone memory algorithm for the first experiment: (a) History(1) algorithm; (b) History(2) algorithm; and (c) History(3) algorithm.

tory(1) with a short length of task history queue, the individual agent is less specialized, and each agent processes many kinds of tasks, similar to the market-based approach. In History(2), the task distribution pattern is similar to those of the other insect-based algorithms. It shows a specialized tendency of some agents (agent 4, 5, and 7), which means that the pheromone memory algorithm can adjust the specialization of agents just by changing the length of the task history. Also, History(3) shows results similar to those of History(2) with a little less specialized tendency. The overall comparison results for all algorithms are

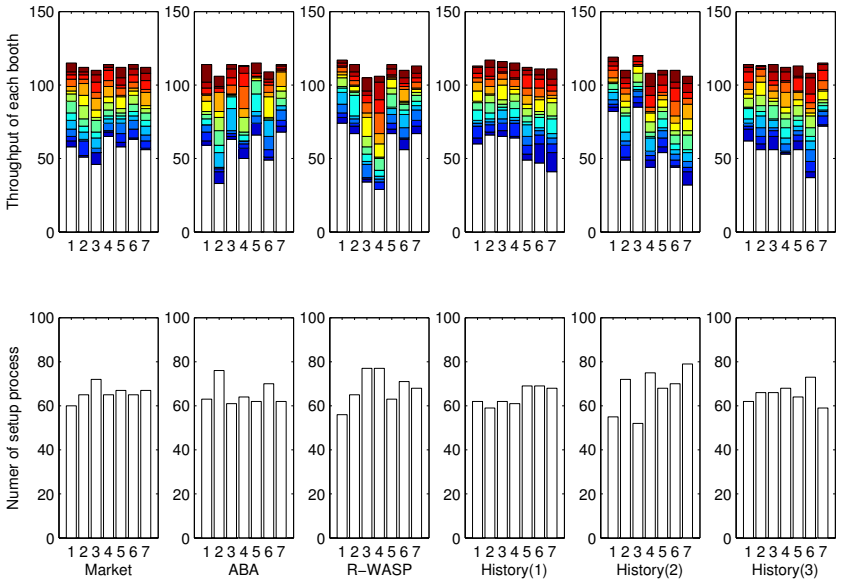


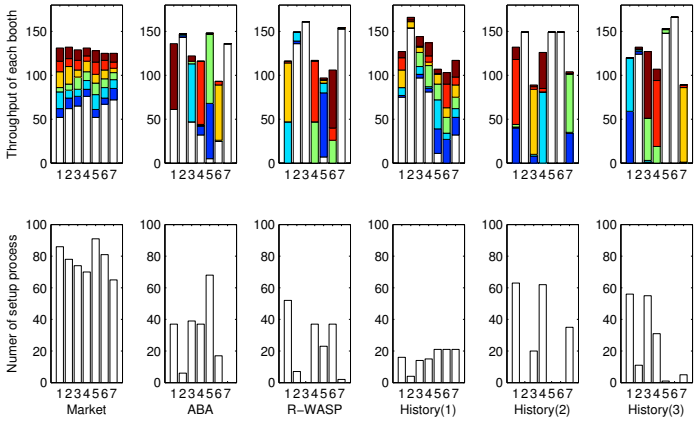
Figure 4.14: Comparison of results with a new environmental setup; the process time is changed to 6-minute, and the setup time is changed to 3-minute.

shown in Table 5.1. The History(1) has the best performance in the number of state changes and it is better in the cycle time and queue length than the other insect-inspired approaches when all the test algorithms maintains the similar performance in throughput as its competitors. In the original experiment environments, due to the non-uniform task distribution, specialization characteristics may have a negative effect on the number of task changes.

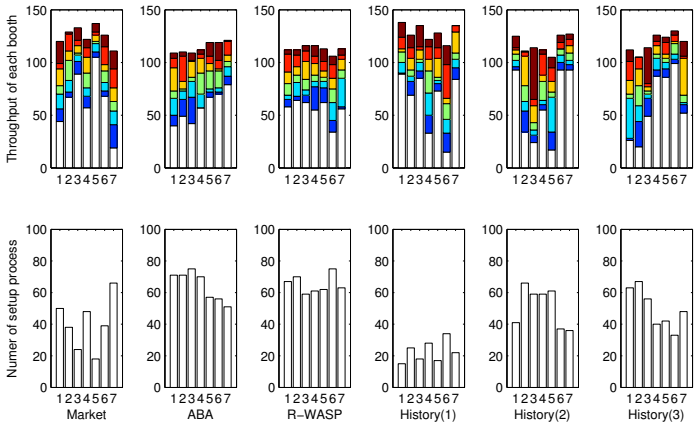
Results with changes in experimental environments

Next, we change the parameters, such as process time for truck painting, setup time for color change, number of assigned tasks, and the distribution of tasks, to check the robustness in various environments. The performance of each algorithm is evaluated in two aspects of throughput and the number of state changes occurrences because the cycle time and queue length have no significant difference among the algorithms.

Non-uniform task distribution



(a)



(b)

Figure 4.15: Comparison of results when the number of assigned tasks is changed from 14 to 7 with the same number of agents: (a) 5 minutes process time, and 1 minute setup time; and (b) 6 minutes process time, and 3 minutes setup time.

First, we change the process time from 5-minute to 6-minute and the setup time from 1-minute to 3-minute. Those conditions are the maximum values that all agents can process the tasks without loss of product. In those conditions, no specializations occur among the agents because the lack of spare queues prevents agent specialization; thus the agents process all the tasks equally, as shown in Figure 4.14. The total number of setups for color

Table 4.3: Overall comparison of results when the number of uniformly distributed tasks is 7 with (a) 5-minute process time and 1-minute setup time and (b) 6-minute process time and 3-minute setup time.

(a)

	Num. of change	Cycle time	Throughput	Queue length
Market	569.90±14.65	7.00±0.00	900.10±0.32	0.12±0.00
ABA	149.90±83.17	8.03±00.98	901.40±1.96	0.78±0.17
R-Wasps	40.50±11.51	7.02±0.30	900.80±1.81	0.56±0.04
History(1)	54.10±11.16	7.01±0.12	900.10±1.73	0.46±0.02
History(2)	40.30±9.84	7.03±0.03	900.20±1.14	0.56±0.04
History(3)	38.20±12.66	7.02±0.02	900.70±1.49	0.55±0.04

(b)

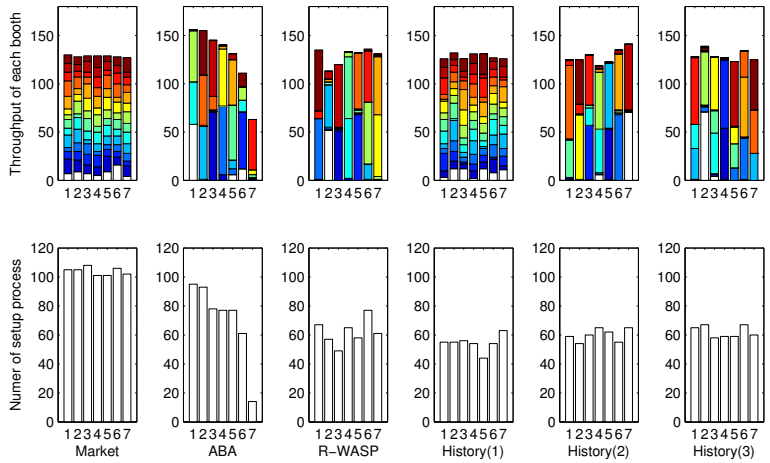
	Num. of change	Cycle time	Throughput	Queue length
Market	336.30±54.60	7.36±0.25	858.30±31.00	2.25±0.23
ABA	434.20±16.28	7.86±0.09	802.30±9.09	2.66±0.05
R-Wasps	434.20±26.61	7.88±0.11	799.90±11.49	2.65±0.10
History(1)	157.40±10.27	7.04±0.02	900.40±2.27	1.20±0.05
History(2)	368.30±63.76	7.65±0.21	829.00±25.41	2.46±0.22
History(3)	319.50±79.62	7.48±0.22	848.50±29.80	2.30±0.34

changes is almost the same in all the algorithms.

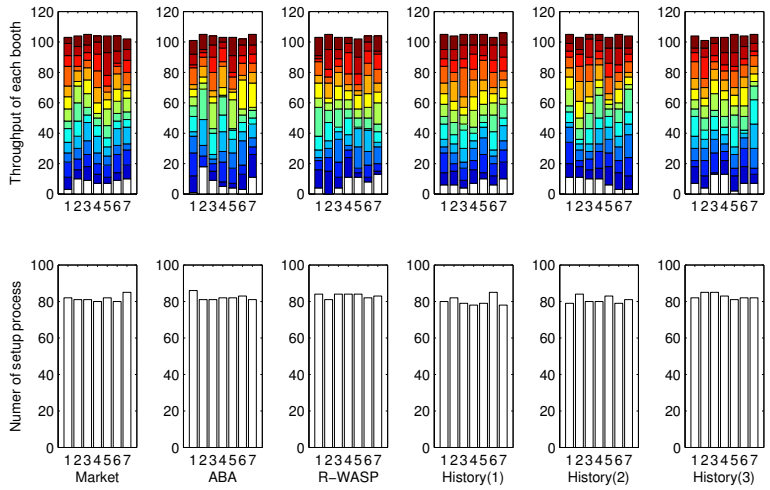
Second, we change the assigned tasks from 14 to 7 with the same number of agents, and the results of the two cases are shown in Figure 4.15. In (a), we set the process time as 5-minute and the setup time as 1-minute, and in (b), we set the process time as 6-minute and the setup time as 3-minute. Those results are clearly similar to the results of the original experiments. The History(1) algorithm shows the best performance in the number of setup processes, and History(2) and History(3) show specialization characteristics.

Uniform task distribution

We repeat the previous four experiments with a uniform distribution of color conditions in the task (the probability that each color is assigned to a truck is the same, so the required colors are selected randomly among all possible colors). The results are shown in Figure 4.16-4.17, with 14 tasks in Figure 4.16 and 7 tasks in Figure 4.17. In Figure 4.16, we obtain results similar to those of the previous experiments. The History(1) algorithm shows the best performance in the number of setup processes, History(2) and History(3) show specialization, and the lack of spare queue prevents an agent specialization.



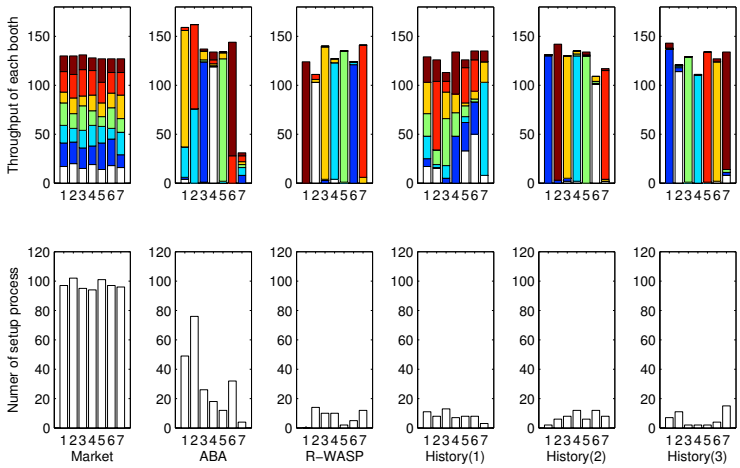
(a)



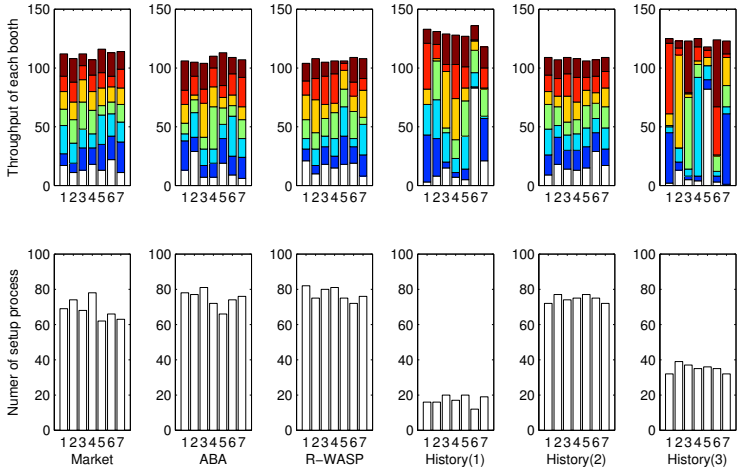
(b)

Figure 4.16: Comparison results of uniform task distribution experiments with 14 tasks: (a) 5 minutes process time, 1 minute setup time; and (b) 6 minutes process time, 3 minutes setup time.

In Figure 4.17, agent specialization is clearly visible in these experimental results and the comparison of the results in Figure 4.17 are summarized in Table 4.3. When the number of tasks is the same as the number of agents, each agent should specialize in a task to



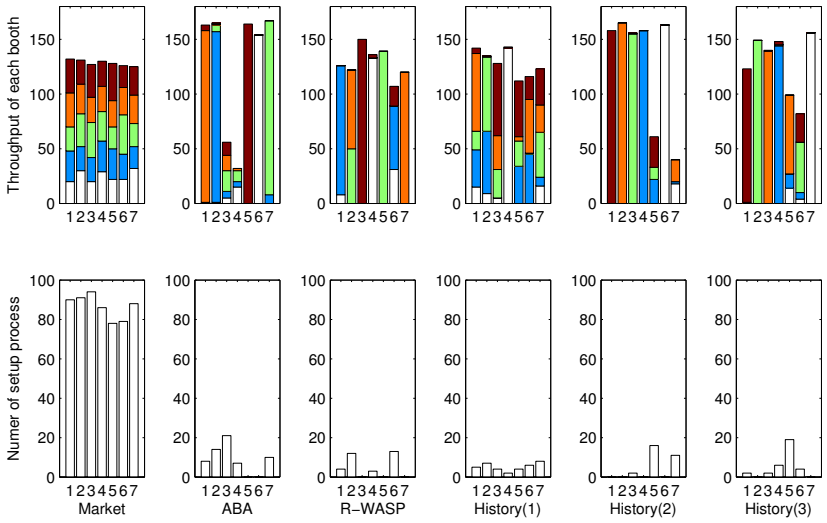
(a)



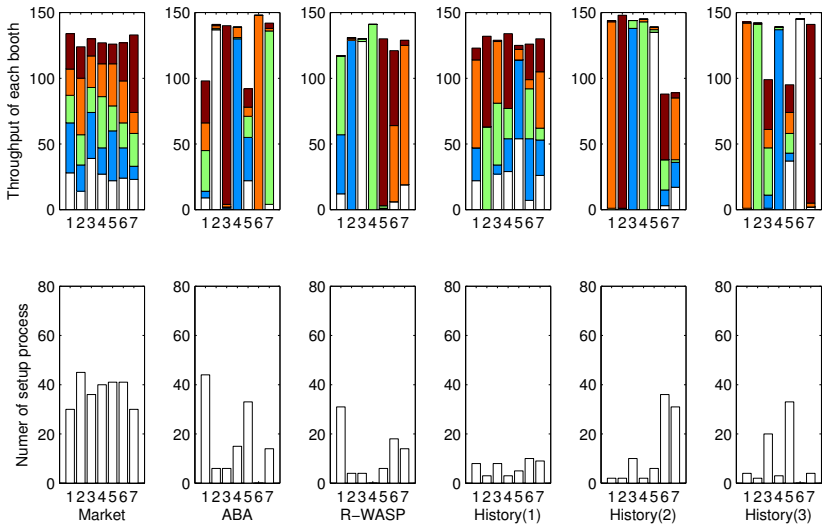
(b)

Figure 4.17: Comparison results of uniform task distribution experiments with 7 tasks: (a) 5 minutes process time, 1 minute setup time; and (b) 6 minutes process time, 3 minutes setup time.

minimize the occurrence of setup processes. So ideally, if each agent processes a single task, no setup processes will occur, and Figure 4.17(a) shows those expected results. The History(2) and History(3) methods show that each individual agent processes one task al-



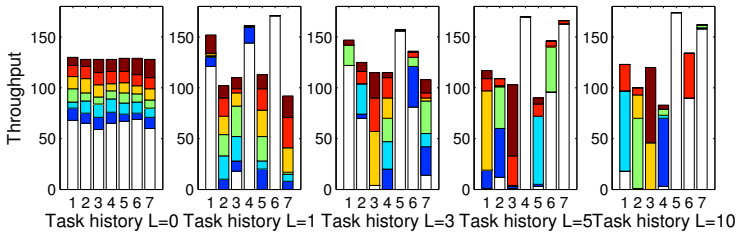
(a)



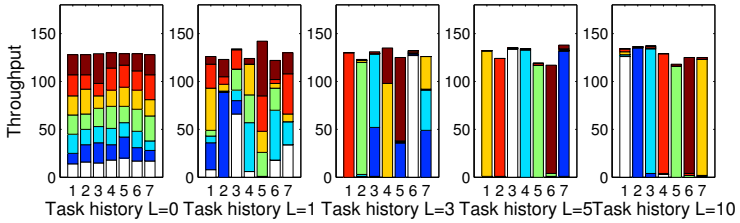
(b)

Figure 4.18: Comparison of results with 5 uniform distribution tasks: (a) 5 minutes process time, 1 minute setup time; and (b) 6 minutes process time, 3 minutes setup time.

most exclusively, though each agent shows a few task changes. This tendency induces better specialization ability than R-Wasps. In History(1), each agent normally processes four or



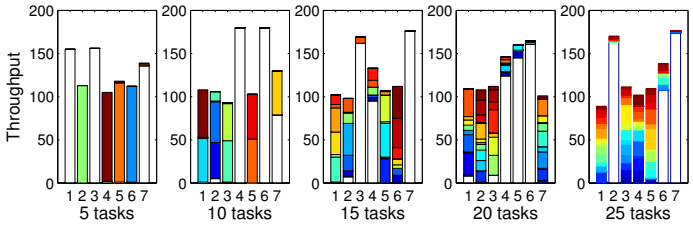
(a)



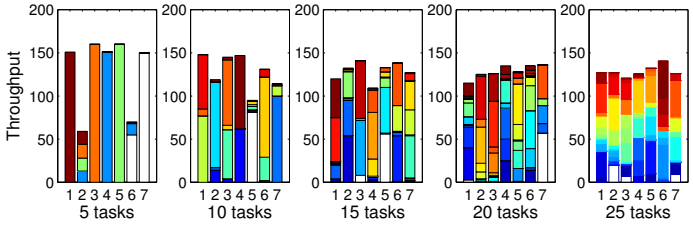
(b)

Figure 4.19: Comparison of results depending on the length of the task history (7 tasks for 7 agents, 5-minute process time, 1-minute setup time): (a) non-uniform task distribution; and (b) uniform task distribution.

five tasks. When the process and setup times become longer, as shown in Figure 4.17(b), the specialization pattern disappears, and the number of state changes increases as in the previous results. In the number of task changes, the History algorithms show similar performances, and History(1) has the best performance among the five algorithms. In addition, History(3) has the properties found in both History(1) and History(2): specialization as in History(2) and various task assignments as in History(1). Because of its long history of task queue, History(3) could have a tendency to specialization, but because of the weighted history, that tendency becomes a little weaker. Because of those merged characteristics, History(3) shows better best performance than History(2) in the number of setup processes and throughput of each booth as shown in Table. 4.3(b). The results when the number of tasks is smaller than the number of agents are shown in Figure 4.18, and they confirm the tendency of each algorithm shown in the previous experimental results. History(1) or History(3) shows good performance, depending on the environmental situation.

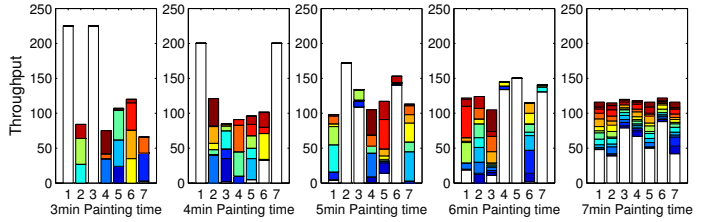


(a)

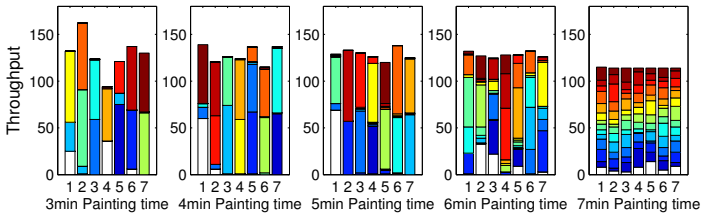


(b)

Figure 4.20: Comparison of results depending on the number of tasks in History(2) (7 agents, 5-minute process time, 1-minute setup time): (a) non-uniform task distribution; and (b) uniform task distribution.



(a)



(b)

Figure 4.21: Comparison of results depending on the process time in History(2) (14 tasks for 7 agents, 5-minute process time, 1-minute setup time): (a) non-uniform task distribution; and (b) uniform task distribution.

Results with changes of parameters

We analyze the effect of system parameters in our proposed algorithm. Figures 4.19, 4.20, and 4.21 show the changes in system performance in various environments, such as task history length, the number of tasks, and the process time under two different patterns of task distribution. Agent specialization is well shown in Figure 4.19. Depending on the pattern of task distribution, some agents or all agents process only one task, and that effect increases with the longer length of the task history. Even when the variety of tasks is much larger than the number of agents, if the portion of one specific task is larger than the throughput of a single agent, specialization is maintained, as shown in Figure 4.20. For example, in Figure 4.20(a), there are 25 tasks for 7 agents, and some agents (agent 2, 6, 7) still process one task mainly. However, if few spaces are available in the queue of an individual agent, then specialization disappears, as shown in Figure 4.21.

One study [111] focuses on market-based approach and Campos's insect-inspired model approach and finds a trade-off between the cycle time and color changes. Recently, Lichocki et al. (2012) reported that an algorithm's system parameters can be tuned using an evolutionary algorithm or hand-tuning technique to obtain an optimized performance in the division of labor based on the response threshold model . Thus, it is important to select appropriate parameters to maximize system performance. All our experiments might show improved performances with better choices for the parameter values of each algorithm. However, evolutionary algorithm or fine-tuning is not recommended in factory-domain applications because it generally takes a long time to get an optimal solution and it is not easy to adapt system parameters to dynamic environmental changes.

In addition, our proposed algorithm generally shows better performance than insect-inspired model approaches. All algorithms apply the response threshold model based on the stimulus-threshold relation, but the core difference is how they regulate the threshold values according to task performance. In two insect-inspired model approaches, the threshold value of the performed task is decreased and the threshold values of non-performed tasks are increased. To have the specialization characteristics, the gap between the threshold value of performed and the threshold values of non-performed tasks is spread. However, in our algorithm, the threshold value of the recent performed task is decreased and the threshold value of the earlier performed task is increased. This principle may be more effective in dynamic scheduling problem based on the stimulus-threshold relation because the threshold is changed depending on its own result of task performance.

In our pheromone memory algorithm, the main parameter that affects the performance

is the size of pheromone memory, that is, the length of the task history. We only need to adjust the length of the task history according to the environment to optimize performance, which could be an easy tuning process. In general, a shorter task history produces faster convergence to the desired division of labor, and as the length of the task history increases, agent specialization capability increases. So History(1) shows better performance than History(2) or History(3) in the original experiment environments, and History(2) or History(3) show better performance than History(1) when the number of tasks is the same as or smaller than the number of agents.

Until now, we have presented a new pheromone memory approach to solving a dynamic scheduling problem to minimize the total number of state changes in a factory application problem. Each machine maintains a limited, constant-sized task history storing its recently completed tasks, and the model calculates appropriate threshold values using a pheromone-based approach. The specialized tendency is controlled by the length of the pheromone memory, and various experimental results show that the proposed method performs mostly better than other conventional methods.

4.5 Summary of Chapter 4

In this chapter, we study the effect of the variable threshold model. In a decentralized approach, it is important to estimate task demands and regulate threshold values. To accomplish this, each agent stores local information about sensed tasks in a history window of finite length. The agent performs sensing behavior periodically, and the types of detected tasks within the sensing range are stored in sequence. New information replaces the oldest information, and the proportion of each task in the history is used as a measure of the needed information. This measure is generated without any centralized control methods. The method can produce varying tendencies to perform tasks by changing the response thresholds given in the task selection function. This ultimately promotes a desired group-level task distribution and reduces the number of task changes. The suggested method is tested using a simulation of multiple robots, and the results of the dynamic task allocation process demonstrate that the proposed method is effective even when only local information about the environment is given to an individual agent in a group.

Scheduling problems in factory domain applications are usually concern many parallel machines, with each machine is able to process several tasks. In most cases, changing the

current machine state to another state to process a different task incurs additional material costs and time. If the overall system can maintain expected performance, minimizing those state changes is very beneficial, and agent-based approaches inspired by task allocation strategies in several social insects have gained increasing attention as offering possible solutions. The basic concept is based on a stimulus-threshold relation that an individual agent determines whether to perform a given task or not based on two factors: the environmental external stimuli of the task and the internal threshold values of all possible tasks. In this approach, obtaining proper threshold values is directly related to overall system performance, and we present a pheromone-based approach obtaining appropriate threshold values. Each agent maintains a limited, constant-sized task history queue of recently processed tasks, and each agent's information is individually used to calculate the threshold values of tasks. From the various experimental results, we show that the performance of the proposed method performs is comparable to that of other conventional methods.

Chapter 5

Task allocation for sequential tasks

In this chapter, we present a self-organized task allocation based on the response threshold model with decomposing a task into sequentially interdependent subtasks and allocating a group of agents to the subtasks. Many insect societies use various task partitioning strategies and the tasks are often divided into sequentially connected subtasks. It is known that the task partitioning is greatly helpful for a colony's survival [186, 197, 64, 191].

In swarm robotics, there have been studies of task partitioning to handle the transfer of objects [178, 181, 28, 121]. Our proposed method shows a self-organized process that originates from the individual decision of robots based on the surrounding local information. The method handles an indirect transfer of food objects in the foraging task. A swarm of agents interact in the cache transfer area for their task change. The method does not require a central controller and the robots do not require communication with each other. Each robot decides whether to switch between subtasks depending on the local information available individually. The response threshold model ultimately controls the task transition rate depending on the task demand and the number of neighboring agents for each task. The repetitive and continuous task selection leads to the desired task distribution in a group level and we analyzed the convergence of task distribution.

In addition, we analyze the effective of task partitioning in various conditions. Then we will show that it ultimately improve the foraging performance. Existing works on task partitioning in swarm robotics have mostly considered object transfer among homogeneous robots. They focused on the object retrieval with a group of robots without any differenti-

ation of robots. Here, we handle robot agents with varying moving speeds. We tested the foraging task with varying moving speeds of agents and found that the task can be self-organized efficiently into subtasks with differentiated agent classes.

This chapter has been submitted in journals [126, 129], accepted for publishing in a journal [129], and prepared to submitted in a journal [133].

5.1 Methods

We describe the problem of task allocation for a foraging task consisting of sequential subtasks. The goal of the method is to determine an allocation of robots to subtasks needed to complete the overall transfer task. It is balancing the number of agents performing each task and maximizing the swarm-level performance, that is, the number of food objects retrieved per time unit. In some cases, robots collect objects in an environment and immediately remove them on that spot [92]. Or robots are supposed to obtain items from the resource and deliver them to another common location or the central place such as home or nest [174, 76]. In our foraging task, collected objects are transported indirectly via a cache area to the nest.

5.1.1 Description of foraging task

We assume that a foraging task is composed of two interdependent subtasks, resource-harvesting and resource-storing tasks. The two primary behaviors will be performed and the simplified state diagram of robots performing the foraging task is shown in Figure 5.1. Agents harvest from resources and deposit food in a cache area. Food from the cache area is supposed to be stored in a central nest, and another agents transport food items to the nest. Harvesting agents travel to a resource area, pick up a food pellet and deliver it so that it can be processed later by a storing agent. To facilitate the transfer of food items to the nest, harvesting agents deposit their food items in a cache. Storing agents travel to a cache area, pick up a food pellet if one is present and deliver it to the central nest. The set of sequential tasks (harvesting from a specific resource area and storing food obtained from the cache in the central nest) are inter-linked. The harvesting task is a prior condition to the resource-storing task and the whole foraging task that transfers one object from the resource area to the nest is finished if both subtasks are completed.

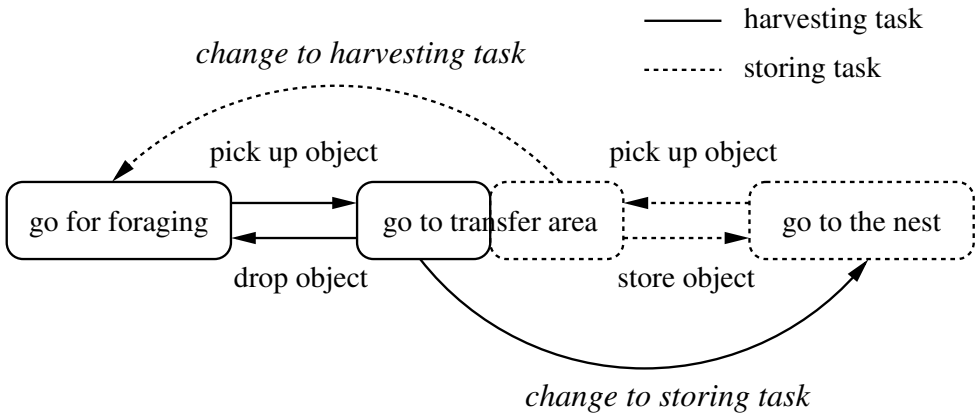


Figure 5.1: Simplified state diagram for robots performing the foraging task. Solid line belongs to the harvesting subtask and dashed line belongs to the storing subtask. The behavior of each robot is determined by the subtask it is currently performing.

The performance of the swarm, the total number of objects stored in the central nest, relies on the number of robots in each area working on the two different subtasks and how the robots interact for object transfer. Varying the proportion of agents to each subtask yields different performances at the overall swarm level. The subtask for storing agents is greatly affected by the subtask for harvesting agents, because food items should be transported by harvesting agents before they are stored in the nest. As mentioned earlier, we consider the foraging task in which object transfer at the cache area is indirect. A robot that arrives at the cache area (transfer area) drops its carrying food object and returns to the resource area. To obtain an optimal allocation, the number of robots working on the storing subtask should be regulated appropriately to transport objects located in the transfer area to the nest. To maximize the overall performance, the proposed method relies on balancing the number of objects located in the transfer area and the number of robots working on the storing task adaptively. The total performance also depends on the characteristics of subtasks or the information about the surrounding environment.

5.1.2 Task selection method

Each agent autonomously decides whether to change the current task or not. The individual changes the current task according to the task selection function. The agent has a changeable

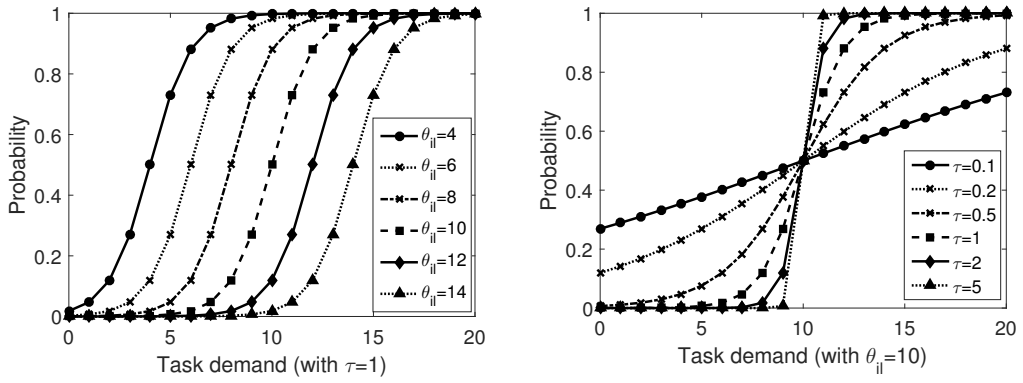


Figure 5.2: Task performing probability curves for various values of (a) $\theta_{il}(t)$ and (b) τ with a given range of task demand $d_{il}(t)$. τ represents the slope of curve and $\theta_{il}(t)$ produces different response value given the same value of $d_{il}(t)$. This is also true of τ, except at the point where all curves intersect.

threshold for each task and responds differently to the same stimulus, which can coordinate the foraging behavior effectively. As a result, a collection of each agent's response produces a swarm-level performance in a self-organized manner.

To explain a mathematical model, we assume that each agent l has a response threshold $\theta_{il}(t)$ for task i. Then an individual agent's willingness to perform task i per unit time is a stochastic term calculated by a sigmoid function based on the response threshold model. For each agent, the task selection function for calculating the probability is given by

$$P_{il}(t) = \frac{1}{1 + e^{-\frac{1}{\tau}[d_{il}(t) - \theta_{il}(t)]}}, \quad \forall i = 1, \ldots, M \tag{5.1}$$

with task demand of task i for agent l, $d_{il}(t)$, the threshold value $\theta_{il}(t)$, and the control parameter τ, which determines the slope of transition probability. Task demand $d_{il}(t)$ is the actual number of agents required to perform task i and $d_{il}(t)$ is the detected task demand by agent l within a limited sensing range.

Each agent decides to change the current task or not based on the probability obtained from Equation (5.1). A agent currently performing task i changes to task j if the probability of task j is the largest among all the values of tasks. The threshold $\theta_{il}(t)$ decreases and the task selection probability for task i increases. If the task demand is relatively lower than the

fraction of agents performing that task, the threshold of the corresponding task increases. In the aspect of an overall system, the task transition rate, $k_{ij}(t)$, can be defined as

$$k_{ij}(t) = \frac{n_{ij}(t)}{n_i(t)} \tag{5.2}$$

where $n_{ij}(t)$ represents the number of robots currently performing task i and becoming to have the maximum probability for task j. That is, those robots change their task to task j. This value $k_{ij}(t)$ changes depending on the condition of each agent but will decrease as the overall system converges to the desired task distribution.

Figure 5.2 shows various response curves for different values of τ and $\theta_{il}(t)$ with a given range. Figure 5.2(a) shows that varying τ's produce the different task transition probability, given the same task demand and threshold. Smaller τ has flatter task transition probability curves and higher τ has steeper curves (the task transition probability grows fast as the task demand increases). Figure 5.2(b) shows that an agent with smaller θ_i tends to respond to small stimulus values and an agent with higher $\theta_{il}(t)$ will not respond to small stimulus values. For $d_{il}(t) < \theta_{il}(t)$, the probability of a given task is close to 0 and for $d_{il}(t) > \theta_{il}(t)$, this probability is close to 1. At $d_{il}(t) = \theta_{il}(t)$, $P_{il}(t) = 0.5$. In the rest of the paper, we consider the case $\tau = 1$ simply, but similar results can be obtained for any τ ($\tau > 0$). In constrat to Equation (4.4), task selection function in Chapter 4, $d_{il}(t)$ and $\theta_{il}(t)$ is not normalized. We directly use the counting information not the normalized proportion.

Regulation of response threshold

To obtain the desired task distribution in a swarm of robots, the task selection probability for a specific task can be increased by lowering the threshold of that task, or decreased by increasing the threshold. This threshold-updating process results in the emergence of specialized agents who are more responsive to tasks with lower thresholds than the others and this tendency produces a gradual migration of task allocation.

In the response threshold model, the threshold is usually updated after performing task [23, 212, 41]. Completing a task induces a decrease in threshold of that task and an increase the thresholds of the other tasks not performed. The more often an agent performs a specific task, the lower its response threshold to this type of task, and vice versa.

In our task allocation algorithm, the individual agent updates its response threshold considering not only the associated task demand but also using local information from other

agents, that is, a task state of other agents. This tendency can be represented as follows:

$$\theta_{il}(t+1) = \theta_{il}(t) - \eta\left\{d_{il}(t) - n_{il}(t)\right\} \tag{5.3}$$

where $\theta_{il}(t)$ is constrained to the interval $[\theta_{min}, \theta_{max}]$ and η is a learning rate for strengthening or weakening factors to regulate the threshold over time. In this paper, the range of threshold is set to $[\theta_{min}, \theta_{max}] = [1, 50]$ for a simple application and η is selected as 1 to change the tendency of individual for a specific task 2% per unit time.

Equation (5.3) means that threshold $\theta_{il}(t)$ for task i is regulated by the fraction of agents performing that task. If the task demand $d_{il}(t)$ is above the quorum, $n_{il}(t)$, the threshold of the corresponding task is decreased and thus the task selection probability for that task is increased. We refer to such a tendency as task specialization and this is further accelerated until $d_{il}(t)$ drops below $n_{il}(t)$. If there are more task demands than the fraction of agents being able to perform that task, then another agent has a chance to work on the task by lowering its threshold and it increases the probability to perform the task. From the repeated process, each agent becomes to have a tendency to perform one specific task and this specialization reduces task changes at equilibrium state, maintaining the expected division of labor.

5.1.3 Convergence analysis

We show that the microscopic behaviors and interaction rules are automatically generated through maximizing the performance evaluation function of the overall system.

Analysis with behavior

In a given foraging task, each agent has no information about the desired target distribution, $\bar{x}_i$, and the current distribution, x_i, then they instead regulate the task distribution with the following assumption,

$$\frac{x_j}{\bar{x}_j} \simeq \frac{n_j/N}{d_j/\sum_{j=1}^{M} d_j} \propto \frac{n_j}{d_j} \tag{5.4}$$

where n_j is the number of agents performing task j and d_j is the demand of task j; the demand can be estimated with the number of observed objects. Equation (5.4) indicates

that task demand, d_j, is instead used to estimate the desired distribution, $\bar{x}_j$, and the number of agents observed in the vicinity, n_j, is used to estimate the current distribution, x_j. Then Equation (4.15) can be written as

$$\begin{aligned}\frac{dV}{dt} &= \sum_{\forall j|(i,j)\in E}\left(\frac{x_j}{\bar{x}_j}-\frac{x_i}{\bar{x}_i}\right)k_{ij}\bar{x}_i\left(\frac{x_i}{\bar{x}_i}-1\right)\\ &\simeq \sum_{\forall j|(i,j)\in E}\left(\frac{n_j}{d_j}-\frac{n_i}{d_i}\right)k_{ij}\frac{\bar{x}_i}{d_i}\left(n_i-d_i\right)\end{aligned} \tag{5.5}$$

The $\bar{x}_i$ and d_i are non-negative values. If we design the $k_{ij}(n_i - d_i)$ has an opposite sign to $(n_j/d_j - n_i/d_i)$, by design if $n_j/d_j > n_i/d_i$, then $k_{ij}(n_i - d_i) < 0$; the transition rate k_{ij} has an opposite sign to $(n_i - d_i)$, and if $n_j/d_j < n_i/d_i$, then $k_{ij}(n_i - d_i) > 0$; the transition rate k_{ij} has an same sign to $(n_i - d_i)$, Equation (4.16) can converge to the desired distribution.

$$\frac{dV}{dt} \simeq \sum_{\forall j|(i,j)\in E}\left(\frac{n_j}{d_j}-\frac{n_i}{d_i}\right)k_{ij}\frac{\bar{x_i}}{d_i}(n_i-d_i) < 0 \tag{5.6}$$

The obtained design rule for converging to the desired performance is reflected in the threshold updating rule in Equation (5.3). If the task demand $d_{il}(t)$ of task i is relatively higher than the number of agents $n_{il}(t)$, in the neighborhood, performing task i, it needs to increase the probability $P_{il}(t)$ by decreasing the threshold $\theta_{il}(t)$ for task i, which induces the increasing number of agents performing that task and the transition rates k_{ij} and k_{ji} are regulated adaptively in a overall system level.

Analysis with object flows

Modeling material flows is another view whether task allocation can be regulated by a common transfer area. For easy understanding, we first explain based on our simulation environment for the foraging task with, the assumption that there are three tasks: harvesting, transferring, and storing tasks.

The first step is to model the rate at which the harvesting robots search for, find, and transfer the collected objects into the first transfer area. We assumed that collecting productivity scales with the density of prey. The rate of the transferred objects, $d_h(t)$, increases with an increasing numbers of harvesting agents, $n_h(t)$, and a decreasing radius of the har-

vesting area, r_h, with a constant factor of α_h. Then the amount of collected food items per unit simulation time step is given by:

$$\frac{d_h(t)}{dt} = -\alpha_h \frac{n_h(t)}{r_h^2} d_h(t) \tag{5.7}$$

This formulation describes the dynamics of the collected items. The first area is conceived of as the number of food items that are located in the environment and that are waiting for transporters to take them to the nest.

In a sequential task, the transfer area is an important location because it is regulated on both sides by robots performing injection tasks. To describe the dynamics of the objects in the transferred area, we have to consider the flow of objects. The number of objects increases with the influx of the collected materials from the harvesting area. In parallel, it decreases with the flux of materials that are transported from the transfer area to the next area to be stored in the nest by storing robots. In the equilibrium state, the number of transferred food items should be maintained. This means that if the rate is the same, then the number of robots between the injected areas is indirectly related to the size of area. If the area is large, then more robots will have to be assigned to that task in order to maintain the same transfer rate.

To explain the mathematical model, we assume that this rate depends on the number of working robots, $n_t(t)$, and objects, $d_t(t)$. A higher density of objects in the environment results in quicker encounters with objects by transporter robots, and thus the transportation rate will increase as the density of objects increases. Thus transport rates in the transferred area will be changed as:

$$\frac{d_t(t)}{dt} = \alpha_h \frac{n_h(t)}{r_h^2} d_h(t) - \alpha_t \frac{n_t(t)}{r_t^2} d_t(t) \tag{5.8}$$

Similar to Equation (5.8), the dynamics of the objects inside the storing area can be described:

$$\frac{d_s(t)}{dt} = \alpha_t \frac{n_t(t)}{r_t^2} d_t(t) - \alpha_s \frac{n_s(t)}{r_s^2} d_s(t) \tag{5.9}$$

where $d_s(t)$ is the number of objects in the storage area and $n_s(t)$ is the number of robots performing the storing task. We assumed that the objects are stored in the nest over time with a constant rate, α_s. The total number of objects stored in the nest can be described by

the second term of Equation (5.9). Then, the overall state equation can be merged as below:

$$\begin{bmatrix} d_h(t)/dt \\ d_t(t)/dt \\ d_s(t)/dt \end{bmatrix} = \begin{bmatrix} -\alpha_h n_h(t)/r_h^2 & 0 & 0 \\ \alpha_h n_h(t)/r_h^2 & -\alpha_t n_t(t)/r_t^2 & 0 \\ 0 & \alpha_t n_t(t)/r_t^2 & -\alpha_s n_s(t)/r_s^2 \end{bmatrix} \begin{bmatrix} d_h(t) \\ d_t(t) \\ d_s(t) \end{bmatrix} \tag{5.10}$$

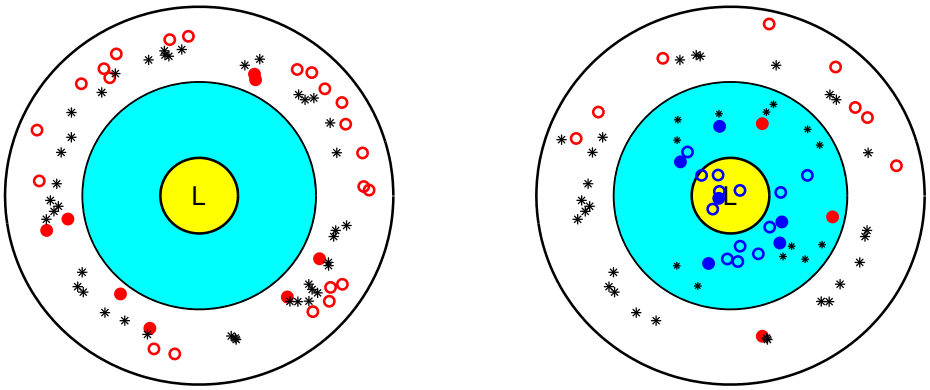
In case of M-subtasks,

$$\begin{bmatrix} \frac{d_1(t)}{dt} \\ \frac{d_2(t)}{dt} \\ \vdots \\ \frac{d_{M-1}(t)}{dt} \\ \frac{d_M(t)}{dt} \end{bmatrix} = \begin{bmatrix} \frac{-\alpha_1 n_1(t)}{r_1^2} & 0 & \cdots & 0 & 0 \\ \frac{\alpha_1 n_1(t)}{r_1^2} & \frac{-\alpha_2 n_2(t)}{r_2^2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \frac{\alpha_{M-2} n_{M-2}(t)}{r_{M-2}^2} & \frac{-\alpha_{M-1} n_{M-1}(t)}{r_{M-1}^2} & 0 \\ 0 & 0 & 0 & \frac{\alpha_{M-1} n_{M-1}(t)}{r_{M-1}^2} & \frac{-\alpha_M n_M(t)}{r_M^2} \end{bmatrix} \begin{bmatrix} d_1(t) \\ d_2(t) \\ \vdots \\ d_{M-1}(t) \\ d_M(t) \end{bmatrix} \tag{5.11}$$

In a sequential task, task changes only occur between the injected tasks. Both harvesting and storing robots can be changed only to the transferring task, while transferring robots can be changed to all other tasks. So, the state matrix can be represented as the lower triangular matrix and all diagonal entries are negative. This means that all the states converge to the stable state, even in multiple stages transfer cases.

5.2 Simulation environment

In our foraging task scenario, the overall foraging task is partitioned into two subtasks; the harvesting subtask and the storing subtask. The harvesting and the storing subtasks have a sequential interdependency as they should be performed one after the other in order to complete the overall task sequence; transporting an object from the resource to the nest. We first explain the experimental environment. Then we evaluate the method using a swarm of



(a) (b)

Figure 5.3: Snapshot of simulation experiments: (a) snapshot of the initial state that all robots are assigned to harvesting task; and (b) snapshot of a desired state that the proper number of robots are assigned to each subtask, harvesting and storing tasks, according to its task demand. A swarm of robots are allocated to two subtasks, harvesting and storing tasks that are sequentially interdependent. Robots working on the harvesting task are represented by red color circles and robots working on the storing task are represented by blue color circles. Unladen robots that move around to pick up empty objects are shown with circles and robots that transfer objects to their destination are shown with color-filled circles, respectively.

robot agents in various simulation experiments.

5.2.1 Environment

The environment is partitioned into three areas. We refer to the three areas marked with three different ground colors as the harvesting area, containing food items to be collected, the cache area (transfer area) as a common storage where objects are collected or dropped, and the central nest where objects are stored finally. Figure 5.3 is a snapshot of the simulation experiment. The yellow-colored circular area in the center stands for the central nest, the middle zone in sky-blue color indicates the cache area and the outer white-colored area is the harvesting area where food objects represented by star shapes are randomly scattered. The radius of the inner circle is 1 m while the middle and outer radius that limits the range of the harvesting and storing task are 3 m and 5 m, respectively. Additionally, a light source

is located at the center of arena. All the robots can sense the direction of the light source, which plays a role of a reference compass.

Each robot is equally capable of completing each subtask and only one type of subtask can be assigned to a robot at any time. A robot working in the harvesting area picks up an object and transports it to the cache area. By dropping the carrying object in the cache area, the object is transferred indirectly to robots working on the storing task. The robots working in the cache area collect objects only in that area and transport them to the nest. Since objects are transferred indirectly, an additional behavior for object transfer between robots is not necessary.

Each robot moves around in the harvesting area, picks up one food pellet at a time, and delivers it to the cache (transfer) area. To transfer objects, robots head towards the light source until they arrive in the transfer area. They drop the carrying objects at random location on the way to the light source and then go back to the harvesting area and move around to pick up another object again. Robot working on the harvesting subtask shuttles between a food resource and the transfer area. Robots performing the storing task can pick up food pellets only within the transfer area and transport them to the central nest. In Figure 5.3, robots working on the harvesting task are represented by red-colored circles and robots working on the storing task are represented by blue-colored circles. Unladen robots are shown with red or blue-color circles and laden robots that transport food objects to their destination are shown with color-filled circles, respectively.

There are a number of objects in the harvesting area and no depletion occurs. The black star shapes stand for objects. In the experiment, an object is picked up by a robot agent and the new one is generated in an arbitrary place in the harvesting area only after the object is dropped in the transfer area. When an object is transported from the transfer area to the central nest, that object is just cleared. The central nest has an infinite capacity for storage.

There are a fixed number of objects in the environment at any given time, given that they are simulated to appear and disappear, and no depletion occurs. The black star shapes stand for objects. An object is picked up by a robot in the harvesting area and is transported to the central nest via the transfer area. A new one is generated in an arbitrary place in the harvesting area only after the object is dropped in the central nest, that object is just cleared. The central nest has an infinite capacity for storage.

5.2.2 Robot behaviors

To imitate real robot behaviors, we take a Khepera-like robot model for simulation [159] using MATLAB. The robot has a round shape with a diameter of 12 cm. It can hold an object to be transported. The robot is a two-wheel, differential drive mobile robot. It can freely control the wheel motor speeds to turn its moving direction. It is equipped with infrared sensors, used to perceive obstacles up to a distance of 20 cm and sense the direction of a light source up to a distance of 16 m. Ground sensors positioned underneath the robot can detect the color of the floor ground. An omni-directional camera is mounted on the top of a robot to perceive objects as well as neighboring robots up to a distance of about 3 m. Each robot can emit a light depending on what type of task it is currently doing, which helps count the number of neighboring agents for each task. There is no communication between robots and the maximum moving speed of robots is 10 cm per time step.

Each robot performs a foraging task by collecting the closest object. It finds an object near the robot and moves to grip the object. Robots can detect objects and neighboring robots by using their omni-directional camera. Simultaneously, a robot performs collision avoidance behaviors for preventing collision with other robots and the outer wall of the arena. Each robot uses eight infrared sensors to read how close other robots or obstacles are positioned to them. The sensors are positioned uniformly around the robot to cover 180 degrees on the front side of robot. A robot can change its moving direction when the sensors detect any obstacle within their sensing range.

Initially, all robots start at random positions in the harvesting area. When robot i with a food object arrives in the transfer area, it drops the carrying object and updates its threshold $\theta_{il}(t)$. In our foraging task, a robot just decides whether to change the current task or not, only one threshold $\theta_{il}(t)$ for the storing task is needed. For this, a robot handles the visual information from the omni-directional camera. The necessary information for the threshold updating is a task demand $d_{il}(t)$, which is estimated as the number of objects in the transfer area and the number of robots $n_{il}(t)$ performing the storing task. We assume that a robot can distinguish objects and robots from visual information obtained using an omni-directional camera mounted on the top of a robot to perceive objects as well as neighboring robots up to a distance of about 3 m.

Each robot initially has the minimum threshold $\theta_{il} = \theta_{min}$ and updates it. Harvesting agents pick up a food pellet in the harvesting area and drops it in the cache area. If its task selection probability becomes lower than that of the other tasks, they decide to change it to the other subtask. They stay in the cache area and start to perform the storing task from

that moment. Here, there is no delay or extra cost to change its task. Robots working on the storing task also decide whether to continue to perform the current task or change it to the harvesting task based on local information. While agents perform the storing task, they continuously update their thresholds at every fixed period (every 20 time steps).Harvesting agents update the thresholds after they drop the food object in the cache area.

5.3 Simulation results

In the dynamic task allocation environment, the amount of task demands or the number of robots may be changed, for example, by adding new objects or removing some robots. A robot may switch its task depending on the environmental situation. In a typical foraging task, the performance can be determined by the number of foraged objects, exhausted time, or consumed energy. If there needs some cost for changing a task, it is also recommended to minimize the task switching maintaining the desired division of labor. In the simulations, we measure the number of objects located in the cache area, the number of robots working on the storing task, and the proportion of agents assigned to each subtask. There are 50 robots and 50 food objects and each simulation runs for 2,000 time steps. The same number of robots as food items is used to compare the performance with the theoretical expected results. Each simulation is repeated twenty times.

For sensors data acquisition and motor control for movement are considered with the inclusion of noisy signals to imitate real robots. However, they never fail to obtain information from the camera image and picking or dropping behaviors for transferring an object. These behaviors in real robots may be slightly different from those in simulated robots. Therefore, the performance of real swarm robots may be slightly different from that of simulated robots.

5.3.1 Result of task allocation

In the beginning of foraging task, all robots are in the harvesting area (all robots are assigned to the harvesting task) and the number of food objects transported to the transfer area increases as time passes. Some robots with lower thresholds start to switch to the storing task, and n_s (number of robots for the storing task) increases and n_h (for harvesting task) decreases. Then a smaller number of objects in the harvesting area will be delivered to the

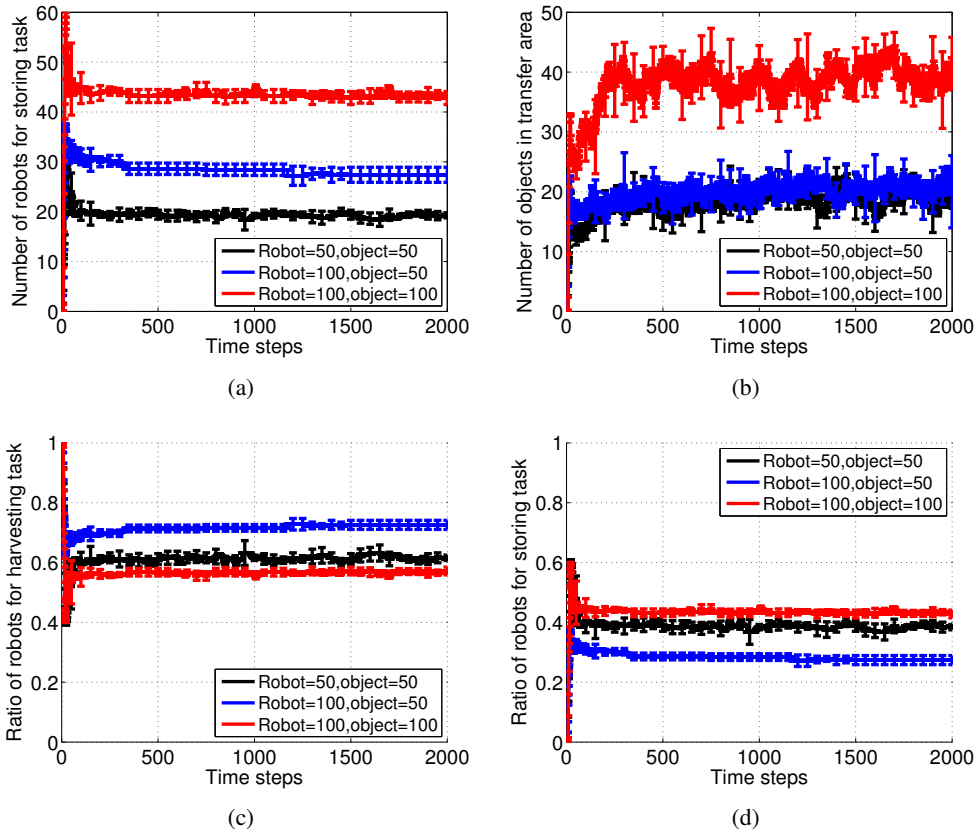


Figure 5.4: Various results of an original simulation: (a) change of the number of robots working on storing task; (b) change of the number of objects remaining in the transfer area; (c) progress of the proportion of robots assigned to harvesting task; and (d) progress of the proportion of robots assigned to storing task. Since all robots are assigned to harvesting task initially, they all start with harvesting task. Then the swarm is split properly by switching a proper number of robots from harvesting task to storing task.

cache (transfer) area from the resource and an appropriate number of robots are assigned to each subtask. The proposed task allocation algorithm can regulate the labor by adjusting the number of robots performing the storing task properly according to the change in the number of food pellets in the cache area.

Figure 5.4 shows the overall task allocation process of the proposed algorithm (black line). For comparison, different number of robots and objects run for comparison (red and blue colored lines). The maximum number of robots working on the storing task is 20 and

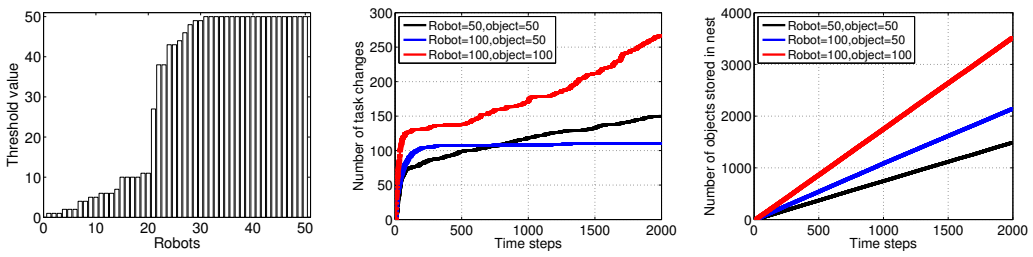
the average number of objects remaining in the transfer area is also about 20. One robot is needed to collect an object in the transfer area. Thus, the same number of storing agents as the number of food objects in the transfer area will be maintained. Both values show the dynamic change of states with some amount of variances as shown in Figure 5.4(a)-(d), but the proportion of robots assigned to each task converges to the stable states as shown in Figure 5.4(c)-(d).

When the number of robots and objects are simultaneously doubled (red colored line), the similar proportions of robots are assigned to harvesting and storing task, respectively as shown in Figure 5.4(c)-(d), while the number of robots for storing task and the number of objects in the transfer area are doubled as show in Figure 5.4(a)-(b). However, when only the number of robots is doubled (blue colored line), the remaining objects in the transfer area is still 20 and less robots compared to the total number of robots are needed to perform the storing task. So the proportion of robots assigned to harvesting task is higher than others.

In a foraging task, when the group harvesting from the resource confronts a higher task demand at the transfer area by observing many food objects, some harvesting agents get to have a higher probability to switch to the storing subtask. When the storing group experiences a lower task demand, some robots storing objects will therefore be more likely to switch to the harvesting task. The number of objects remaining in the transfer area will gradually reach the equilibrium state. Additionally, the proportion of the two groups will become equal. There is some overshoot in an earlier time due to the tipping effect. This is because the initial thresholds of all robots are given completely same and it will be improved if the threshold fluctuates.

Figure 5.5 shows the changes of the threshold of all robots and the number of task changes of an individual robot. All robots start with the minimum threshold (θ_{min}) and after updating the threshold, the distribution of thresholds are changed as shown in Figure 5.5(a). Each robot has a different threshold and this tendency induces a different response to the changes of environment and finally leads to the division of labor. A robot with lower thresholds has more specialized tendency to perform a specific task without task change and this leads to less frequent task changes for the individual robots. This kind of task adaptability needs frequent task changes at the initial state and a small number of task changes are needed after the task distribution converges to the stable state.

As shown in Figure 5.5(b), during the first 200 simulation time steps, a steep increase is observed, but after that time, the slope becomes to follow a gentle curve. The number of task changes becomes small while the task partitioning converges to the desired stable state.



(a) (b) (c)

Figure 5.5: Change of (a) threshold from initial threshold (θ_{min}) to final threshold, (b) the total number of task changes of an overall group, and (c) objects stored in nest.

The difference in the performances are evident if there are costs related to task changes. In a constraint condition with a consumed energy for task change, if the number of task change increased, the total energy wasted in the foraging task increased rapidly. When there are more robots than objects (blue-colored line), the number of task changes is not increased. More detailed analysis about the effect of the ratio between the robots and objects is left our future work. The performance of an overall system can be also evaluated by the number of objects stored in the nest. As shown in Figure 5.5(c), the performance increases linearly without fluctuation despite the change of task changes, since the task partitioning is effective. More improved performance is obtained with increasing the number of robots and objects. In our previous works [130], we show that an individual robot has to exert energy for wandering steps, gripping food objects, and task changes. The total number of wandering steps is almost the same and the energy required for performing foraging task is mainly depending on the number of foraged objects and the number of task changes. Therefore, the difference in the performances among results is evident when there are difference in the number of task changes and gripping behaviors. If the cost of each behavior increases, the total energy wasted in the foraging task increases rapidly.

Figure 5.6 shows the effect of using the response threshold model. Instead of using thresholds for task selection, each robot can just determine its task based on the instantaneous information as given below:

$$P_{il}(t) = \frac{1}{1 + e^{-\frac{1}{\tau}[d_{il}(t) - n_{il}(t)]}}, \quad \forall i = 1, \ldots, M \tag{5.12}$$

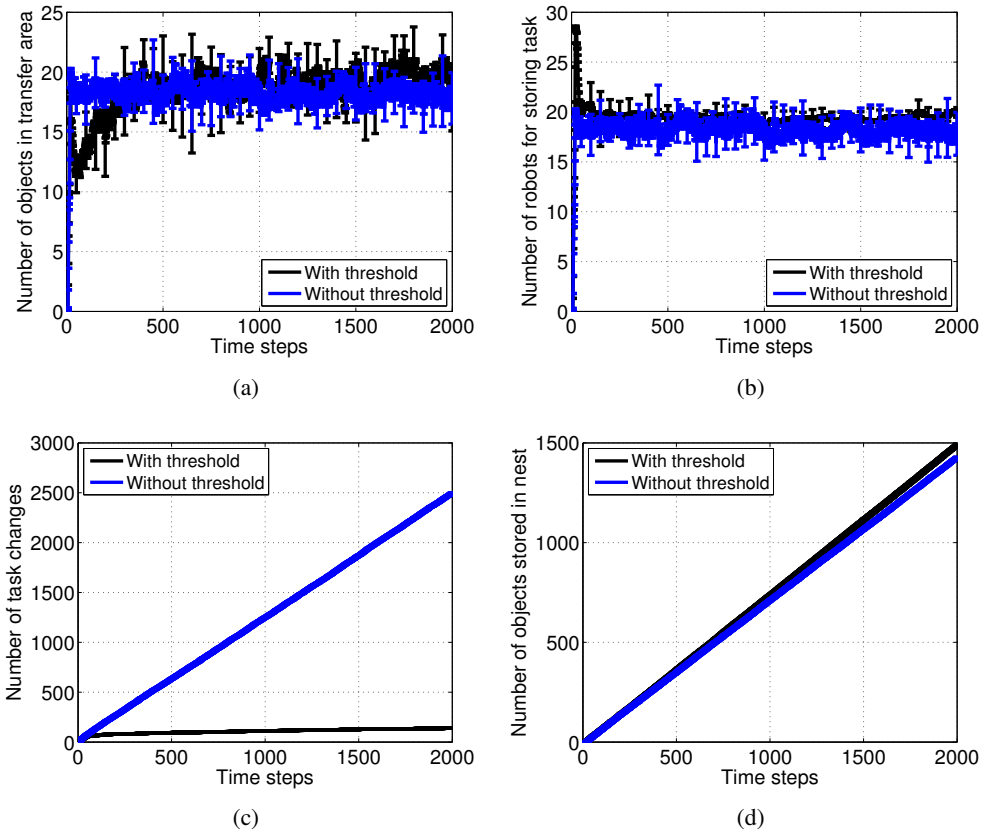
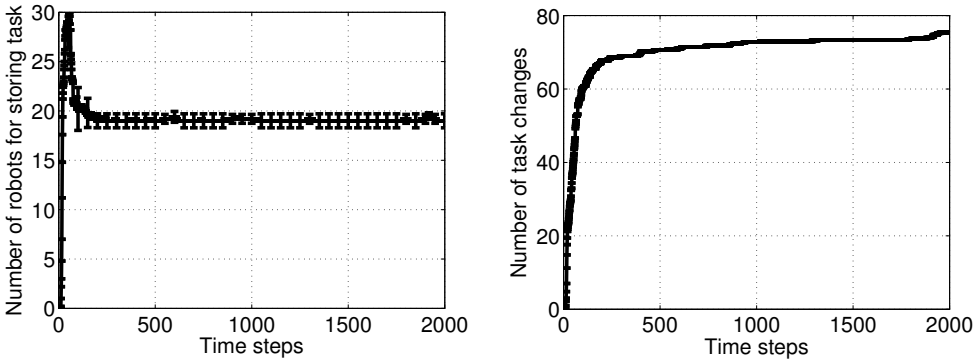


Figure 5.6: Comparison of the results with threshold and without threshold for task transition function: (a) the number of objects in the transfer area; (b) the number of robots for the storing task; (c) the total number of task changes for the overall group; and (d) the total number of food objects stored in the nest.

As shown in Figure 5.6(a), the number of objects in the transfer area reaches to the desired level faster in case of without thresholds, but less robots with lager variation are assigned to storing task and much more task changes are needed even with the similar performance as shown in Figure 5.6(b)-(d).

To confirm that the proposed response threshold model has a stable equilibrium point that satisfies the desired target distribution, we assume that each robot knows the desired number of robots remaining in the transfer area. The desired number of robots is proportional to the size of area they move around. Then the threshold can be updated only depend-



(a) (b)

Figure 5.7: Results using only the number of objects for threshold regulation, but holding information of the desired target distribution: (a) the number of robots assigned to the storing task; and (b) the total number of task changes.

ing on the current number of robots in the transfer area as follow:

$$\theta_{il}(t+1) = \theta_{il}(t) - \eta\left\{\bar{n}_{il} - n_{il}(t)\right\} \tag{5.13}$$

Fast time to the stable state and less task changes are needed in this approach as shown in Figure 5.7. The number of robots for storing task is the same with the result in Figure 5.4(a). In Figure 5.4(a) and Figure 5.7(a), the number of robots performing storing task (n_s) is about 20. This is the similar with the theoretical expected value.

If there are two types of foraging tasks in a circular arena, the desired task distribution can be calculated by mathematically. To balance the workload among sequential subtasks in Euclidean space, the desired number of robots for each task is in proportional to the number of objects in the given area and in inverse proportion to the area they move around if all robots are homogeneous in their ability for performing tasks. Then in a stable equilibrium point, the balancing condition is defined as

$$\frac{d_s}{r_s^2} n_s = \frac{d_h}{r_h^2} n_h \tag{5.14}$$

where d_s and d_h are the number of objects in storing and harvesting area with radius r_s and

r_h, respectively. If the number of robots and objects are the same, Equation (5.14) can be rewritten as

$$\frac{n_s}{r_s^2} n_s = \frac{(N - n_s)}{r_h^2}(N - n_s) \tag{5.15}$$

where N is the number of robots and objects. Then, it is

$$\frac{n_s^2}{r_s^2} = \frac{(N - n_s)^2}{r_h^2} \quad \rightarrow \quad \frac{n_s}{r_s} = \frac{(N - n_s)}{r_h} \tag{5.16}$$

Equation (5.16) means that the proportion of task partitioning is decided according to the radius of foraging area. Finally, the number robots performing storing task can be defined

$$n_s = \frac{r_s}{r_s + r_h} N \tag{5.17}$$

Then $n_s = 3/(3 + 5) \times 50 = 18.8$. From the result, we note that our proposed method converges to the optimal task distribution.

5.3.2 Result with changes of arena size

To study the influence of environments on the performance of the proposed method, we run several cases by environmental conditions or agent parameters. First, we investigate the adaptability to the changes in the size of foraging area. In an original simulation, the radius of the harvesting area is 5 m (black colored line). We change it to 15 m (blue colored line). Figure 5.8 shows the comparison results with the previous simulations. The number of robots performing storing task is about 20 when the radius of harvesting area is 5 m. When the radius of harvesting area is changed to 15 m, the number of robots is a little less than 10. This is the same with the expected results; $n_s = 3/(3 + 15) \times 50 = 8.3$ when radius r_h is 15 m.

Due to the increasing size of foraging arena, robots need to move more distance to food objects and transport them to the transfer area, the object-transfer rate is reduced and more robots are assigned to the foraging task. Then half as many as robots are sufficient to perform the storing task to handle food objects in the transfer area under that environment. Depending on the environmental situation, an adaptive task allocation within the swarm group autonomously regulates the division of labor for sequential subtasks.

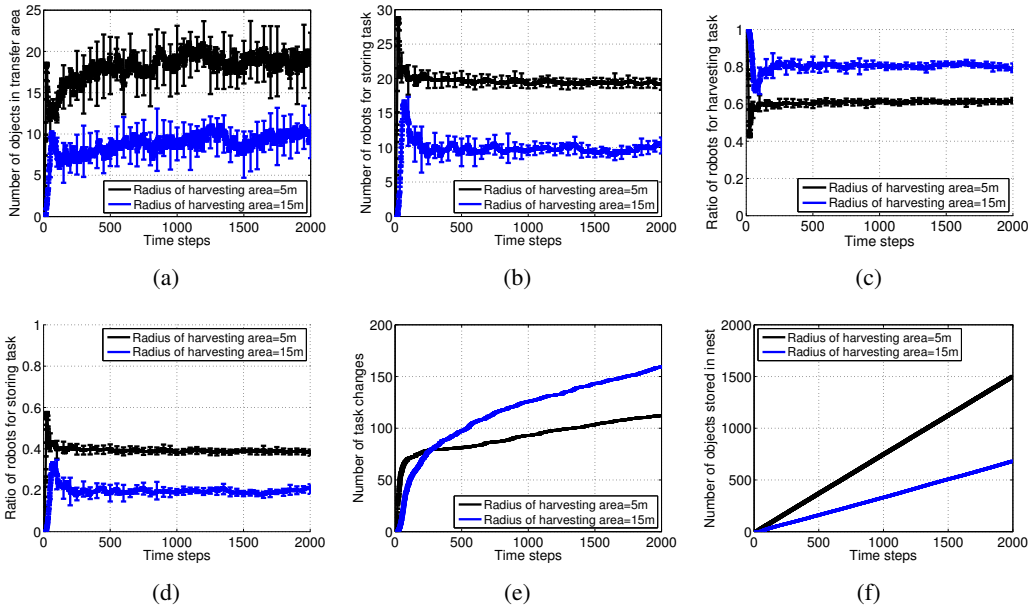


Figure 5.8: Comparison of the results with different foraging area size: (a) number of objects remaining in the transfer area; (b) number of robots assigned to storing task; (c) proportion of robots assigned to harvesting task; (d) proportion of robots assigned to storing task; (e) task changes of an overall group; and (f) objects stored in nest. The radius of harvesting area is changed from 5 m to 15 m.

5.3.3 Result with changes of moving speed

In the next simulation, we investigate adaptability of the swarm to change in the moving speed of robots. To make the difference in completing each subtask, we change the speed of robots performing the storing task, while the speed of robots performing the harvesting task keeps the same as in the previous results.

Figure 5.9 represents the results. If the robot's speed is the same in the two groups, similar results (black colored line) were achieved as shown in Figure 5.4(d) that the proportion of robots assigned to the storing task converges to about 40%. When we reduce the speed of robots working on the storing task by half (blue colored line), more robots are assigned to that task to reduce the number of objects collected in the cache transfer area. If the speed of robots performing the storing task is four times faster than robots performing the harvesting task (red-colored line), fewer robots are needed to transport objects to the nest and the pro-

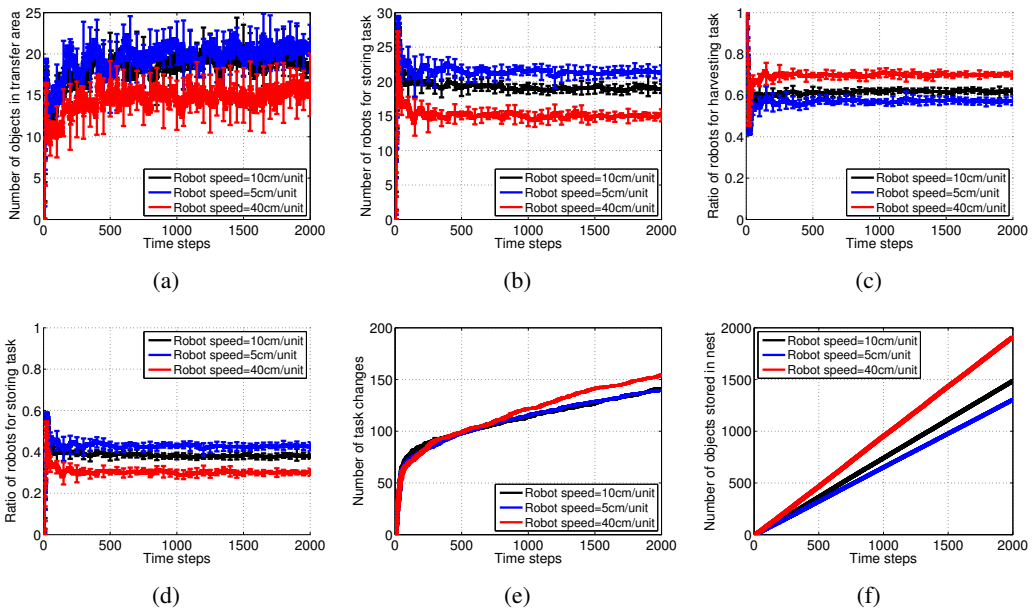


Figure 5.9: Comparison of the results with different moving speed of robots performing storing task: (a) number of objects remaining in the transfer area; (b) number of robots assigned to storing task; (c) proportion of robots assigned to harvesting task; (d) proportion of robots assigned to storing task; (e) task changes of an overall group; and (f) objects stored in nest.

portion of robots performing the storing task is reduced to 30%. In each case, the number of objects in the transfer area is changed as shown in Figure 5.9(a) because our algorithm in Equation (5.3) depends on the number of robots working on the storing task. We observe that our method dynamically allocates the task to a swarm of agents.

5.3.4 Result with time delay for changing task

In our simulations, we assume that there is no delay or extra cost to change its task. However, in real robots or applications, there needs some cost for changing a task; exhausted time, or consumed energy. Figure 5.10 shows the comparison results when the robots pause at their current location for 50 simulation steps after changing their task. If an extra time is required to change the task, the performances are totally worse. There are much overshoot, delayed time to be stable, and more number of task changes. But, it eventually converges to

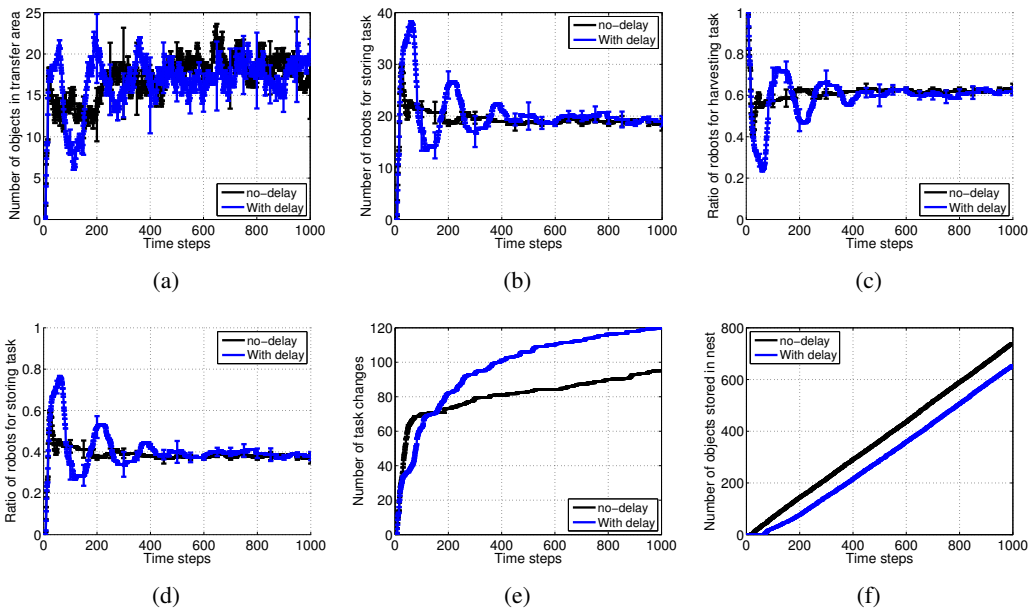


Figure 5.10: Comparison of the results with time delay and without delay for task change: (a) number of objects remaining in the transfer area; (b) number of robots assigned to storing task; (c) proportion of robots assigned to harvesting task; (d) proportion of robots assigned to storing task; (e) task changes of an overall group; and (f) objects stored in nest.

the same results in task distribution. The same number of objects is remained in the transfer area and the same number of robots perform storing task. The algorithm is still effective even if the assumption is violated.

5.3.5 Result with changes in number of agents

We investigate how the system responds robustly to an abrupt change of the task distribution. At time step 1,000, all agents assigned to the storing task are kidnapped from the swarm. The environment is the same with the original simulation. Figure 5.11 represents the behaviors of swarm in response to the change and shows that the swarm reacts properly by allocating a proper number of robots to the storing task. The task allocation among the remaining robots performing the harvesting task is regulated and the robots with low thresholds switch their current task to the storing task. Thus, it lowers the task demand of the storing task. A new proportion of robots are assigned to each task again, and the pro-

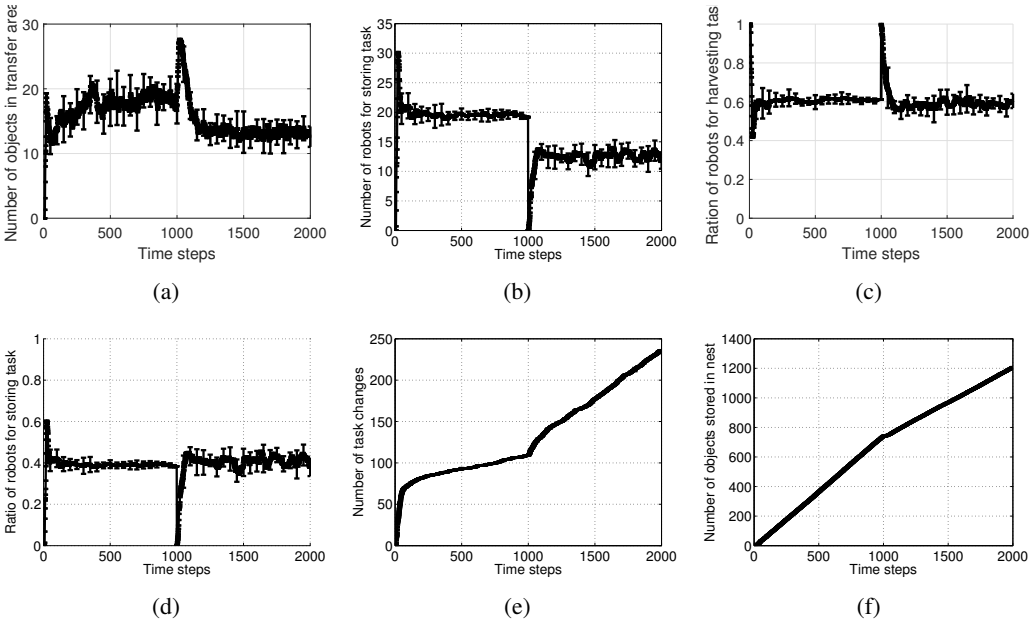


Figure 5.11: Results with sudden changes in the number of robots: (a) number of objects remaining in the transfer area; (b) number of robots assigned to the storing task; (c) proportion of robots assigned to the harvesting task; (d) proportion of robots assigned to the storing task, (e) task changes of an overall group; and (f) objects stored in nest.

portion converges to a stable state within a short time.

5.3.6 Results with multiple tasks

Lastly, to confirm that the proposed algorithm could be used for multiple subtasks, we performed a similar foraging task in which a swarm of robots were allocated to three subtasks: harvesting, transferring and storing tasks. Harvesting robots drop the collected objects in the transfer area and the transferring robots transfer them to the storing area. The transferring area is added between harvesting and storing area as shown in Figure 5.12(a). The outer radius of the transferring and harvesting area are 9 m and 15 m, while the outer radius of storing area is 3 m, the same radius with the original experiment as shown in Figure 5.3. For task partitioning with multiple transfers, each robot has three different threshold values for three tasks and selects the task with the maximum probability.

In a sequential task, task changes are occurred between the injected tasks. The harvest-

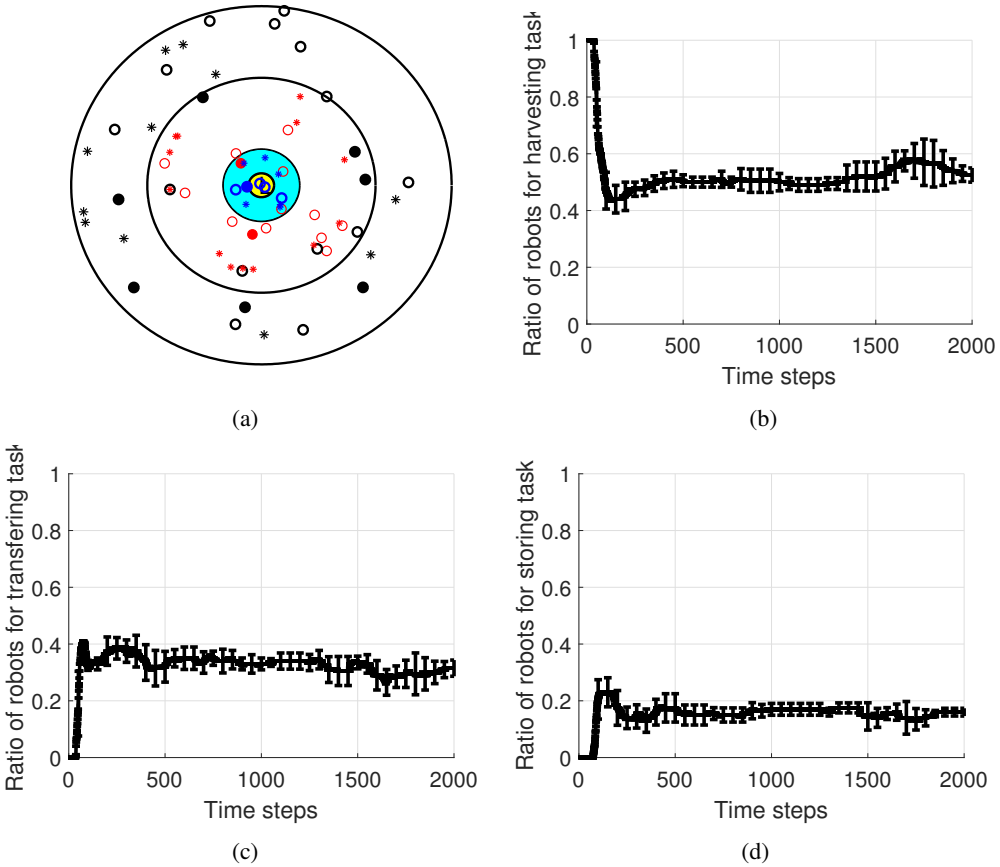


Figure 5.12: Results with multiple tasks: (a) arena composed of three tasks, harvesting, transfer and storing: (b) proportion of robots assigned to the storing task; (c) proportion of robots assigned to the transferring task; and (d) proportion of robots assigned to harvesting task.

ing robots can only change to the transferring task and the storing robots can change to the transferring task. In case of transferring robots, they can be changed to all others tasks. Figure 5.12(b)-(d) show the proportion of robots performing each task. All robots started with the harvesting task, and an appropriate number of robots changed from the harvesting task to the transferring task and storing task. The proportion of robots assigned to each task converged to the desired equilibrium and this accorded very closely with theoretical expectations. In a stable equilibrium point, the balancing condition is as follows:

$$\frac{d_s}{r_s^2} n_s = \frac{d_t}{r_t^2} n_t, \quad \frac{d_t}{r_t^2} n_t = \frac{d_h}{r_h^2} n_h \tag{5.18}$$

where d_t and n_t are the number of objects and robots in a transferring area with radius, r_t. The number of robots performing each task for multiple tasks can then be defined as:

$$n_s = \frac{r_s}{r_s + r_t}(N - n_h), \quad n_t = \frac{r_t}{r_t + r_h}(N - n_s) \tag{5.19}$$

By solving Equation (5.19) with the condition that the number of robots be conserved, $N = n_s + n_t + n_h$, we noted that the proper number of robots for each task is directly proportional to the area of the arena for each subtask. This is the same result with Equation (5.17). We can then expect that the desired proportion of multiple tasks is as follows: $n_s = 3/(3 + 9 + 15) = 11.1\%$; $n_t = 9/(3 + 9 + 15) = 33.3\%$; and $n_h = 15/(3 + 9 + 15) = 55.6\%$. In Figure 5.12, we can see that the proportions of each task converges to the expected task distribution.

Figure 5.13 shows the distribution of the response threshold values for each subtask after finishing a task. In Figure 5.13(b),(c), we can see that the response thresholds for current performing tasks are lower than that of others and there is also difference among robots assigned to the transferring task. This means that some robots were strongly specialized to specific tasks and others are softly specialized to all tasks. This tendency an adaptive induces task allocation in a group. In case of the transferring task, thresholds of other tasks are the same with the maximum limitation. This means that robots performing the transferring task specialized strongly to current performing task and minimized the probability to respond to perform other tasks. In case of the harvesting task, the threshold values are generally higher than others. This means that robots performing harvesting task will change to transferring task easily if there is a little increasing in task need for the transferring task.

To evaluate the proposed task selection algorithm with regard to the effects of environmental influences, we investigated the algorithm's adaptability in response to changes in the size of the foraging area. The radius of the harvesting area, which was 15 m in the original experiment, was changed to 30 m, doubling the radius of an original area. Figure 5.14 shows the proportion of robots assigned to each task. Due to the increasing size of the harvesting arena, more robots were assigned to the harvesting task to balance the transfer rate among sequential tasks. Under these environmental conditions, half as many as robots were needed for the transferring task to handle objects in the transfer area. Depending on the environ-

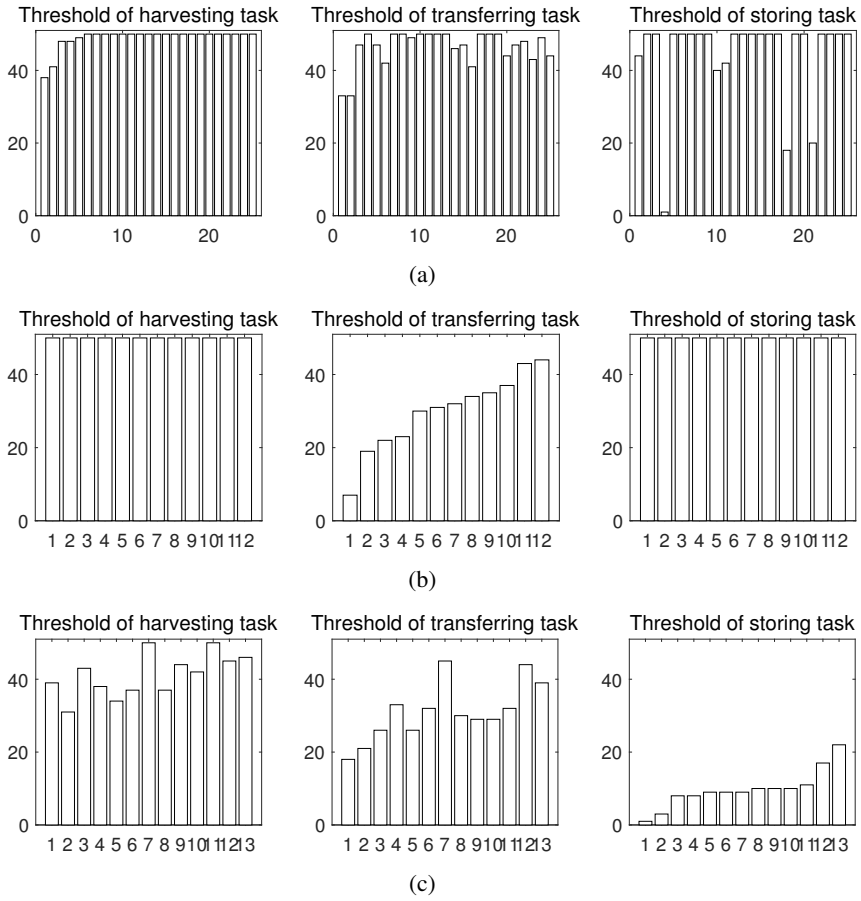
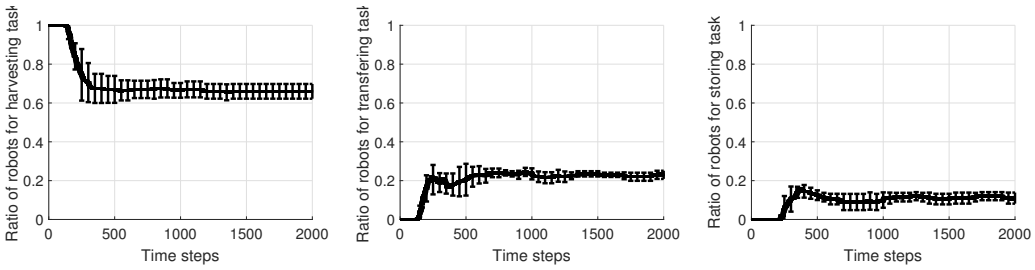


Figure 5.13: One example of thresholds distributions for three tasks after simulation: (a) thresholds of robots assigned to harvesting task; (b) thresholds of robots assigned to transferring task; and (c) thresholds of robots assigned to storing task.

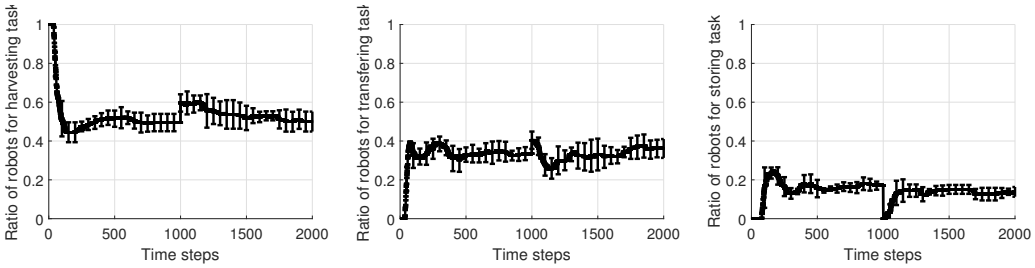
mental situation, an adaptive task allocation within the swarm group autonomously regulates task distribution for sequential tasks. In the experiment there was a delay in task allocation, but it eventually converged to the expected state with $n_h = 30/(3+9+30) = 71.4\%$, $n_t = 9/(3+9+30) = 21.4\%$, and $n_s = 3/(3+9+30) = 7.1\%$.

In case some agents failed, are kidnapped or, disappear, the remaining agents should be reassigned to a task depending on the current condition of the colony. Figure 5.15 shows the adaptability of the swarm to change in the number of agents. With the same conditions as in the original environment, all agents assigned to the storing were removed from the arena at



(a) (b) (c)

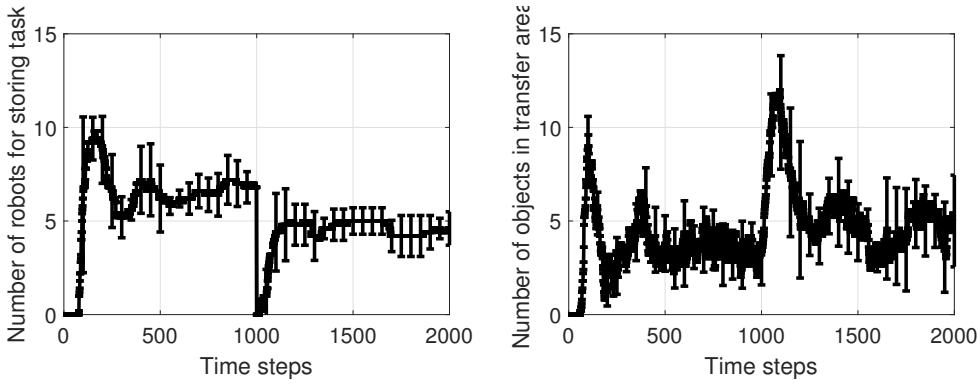
Figure 5.14: Results with following changes in harvesting area size: (a) proportion of robots assigned to the harvesting task; (b) proportion of robots assigned to the transferring task; and c) proportion of robots assigned to the storing task. The radius of the harvesting area is changed from 15 m to 30 m.



(a) (b) (c)

Figure 5.15: Results with changes in the number of robots: (a) proportion of robots assigned to the harvesting task; (b) proportion of robots assigned to the transferring task; and c) proportion of robots assigned to the storing task.

1,000 simulation time steps. Overall systems reacted appropriately. Figure 5.16 shows more detailed analysis. Seven or eight robots were performing the storing task before the change, and five robots performed the storing task after the change. After robots were disappeared, the number of objects located in the transfer area increased temporarily. But after some robots were assigned to storing task again, the system returned to a stable state. In case of multiple tasks, the algorithm worked well.



(a) (b)

Figure 5.16: Results with changes in the number of robots: (a) number of robots assigned to the storing task; and (b) number of objects remaining in the transfer area.

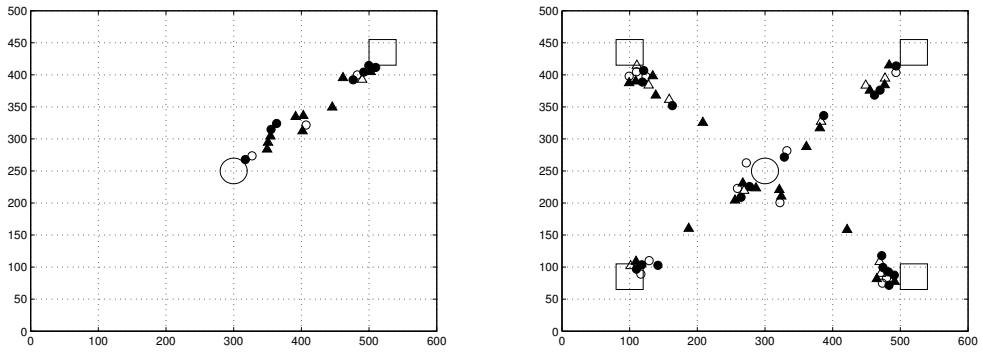
5.4 Discussion

Several social insects use one of the task partitioning strategies based on bucket brigades in their foraging tasks. These are easily observed in nature. A worker passes its load to one of the unladen workers. There is no pre-determined transfer location between the departure and the destination of the resource. Bucket brigades can take place in many situations such as moving broods or food to the nest and moving garbage out of the nest. Bucket brigades (BBs) is one of the multi-stage task partitioning methods shown in the foraging behaviors of the seed-harvesting ant *Messor barbarus*, the African stink ant *Pachycondyla tarsata*, and the grass cutting ant *Atta vollenweideri* In this section, we present the effect of BBs for a multi-robot foraging task to study why task partitioning is used.

5.4.1 Task description

We test a foraging task with multiple robots under various environmental conditions to see the effect of task partitioning. The goal of the robots is set to collect as many objects as possible in a given time span among various demands. Robots are supposed to harvest objects from the resource area and deliver them to a base station called the nest by imitating social insects collecting food pellets from the resource.

The overall system performances of task partitioning and non-partitioning groups are



(a) (b)

Figure 5.17: Snapshot of simulation experiment: (a) one food source; and (b) four food sources. The nest is represented by an empty circle at the center of a given area and four food sources (squares) are located at each corner in a rectangular arena. Small circles and triangles represent robots with different moving speeds. The speed of robots with circular marks is twice as fast as robots with triangular marks. The empty figures of robots are unladen and color-filled figures are laden robots carrying items.

compared by counting the total number of collected objects after a given amount of time. We will often refer to the resource area as the food source, robots as agents, the base station of robots as the nest, and collecting particles or objects in the resource area as collecting food pellets to make the foraging task understood easily using a biological metaphor.

The foraging task is simulated in a rectangular arena as shown in Figure 5.17. The arena has a size of 600 cm by 500 cm. The nest is represented by an empty circle located at the center of the arena in the coordinates of $(x, y) = (300, 250)cm$. The size of the nest entrance will change in the experiments to see the effect from its size. One square at a corner or four squares located at each corner with a side length of 40 cm indicate the food sources (reservoirs of objects). If robots arrive at a resource area, they pick up objects like food pellets and return to the nest. After they reach the nest, they drop the carrying item at the nest and then move to the resource area again to obtain another object. They repeat the behavior in the simulation.

5.4.2 Robot behaviors

We test a swarm of robots running for the foraging task. For food transfer behavior, a

Algorithm 1 Robot behavior

```
while running do
  if robot has no obstacle in front then
    if the robot delivering the food object is in the nest area then
      drop the carrying object
      move to the resource area
    else if the robot collecting an object is in the resource area then
      pick up an object
      move to the nest
    else
      if the robot has no object then
        move to a food source
      else
        move to the nest
      end if
    end if
  else
    if object transfer is needed then
      pass the carrying object or receive the object
    else
      avoid collisions with robots or obstacles
    end if
  end if
end while
```

pair of agents should come close within the vicinity on the trail. Then the two agents stay for a while to complete the transfer instead of simulating releasing and grasping an object. After transfer, the agents change their moving directions to their destination, to the food source or to the nest. The behavior thus requires the food transfer delay.

Small circles and triangles shown in Figure 2 represent robots performing the forging task with different moving speeds. Two different shapes represent varying speeds of robots. The speed of the robots with circular marks is 2 cm per unit time step and the robots with triangular marks move 1 cm every time step. That is, the former group of robots can move twice as fast as the latter group. Empty circles or triangles indicate unladen robots moving to the food source and color-filled robots are laden robots carrying an object from the source to the nest. A robot can pick up one object at a time and after delivering it to the nest or passing it to another robot, it can try to take another object.

The algorithm for the robot behavior is briefly explained in Algorithm 1. At the start of

the simulation, all robots are randomly located near the nest and move to the food source. If there are multiple food sources, all robots are divided equally into subgroups and an equal number of robots are assigned for each food source; each agent has its preferred food source. To skip exploration process for food, we assume that each agent chooses a direct route to one of food sources or to the nest. There will be chances to meet an obstacle or a robot for food transfer before reaching the destination. The agent will choose obstacle avoidance behavior or food transfer behavior depending on its current state.

To harvest food particles, robots head towards the food source and continue in a straight line until they meet other robots or reach the destination. They may need to avoid colliding with other robots on the trail. For collision avoidance, each robot uses infrared sensors to read how close other robots or obstacles are positioned to them. The infrared proximity sensors are uniformly spaced to cover 180° on the front side of robot. The detecting range of the sensors is 2 cm and a robot can change its moving direction when the sensors detect any obstacle within the vicinity. We set the turning angle to 10 degrees. For collision avoidance, a robot turns right when the left sensors detect any obstacle with high intensity or turns left when the right sensors detect any. To avoid collision, a robot moves in a curve at about 25% of the maximum speed. Random noise 10% is added to the two wheel motor actions, and the turning angle is affected by the two wheel motor actions. The robots deviate a little from the direct route to their destination by changing the direction they head if they detect other robots on the front side. This obstacle avoidance behavior has the highest priority among robot behaviors.

There are a number of food pellets in the resource areas and no depletion occurs. Each agent moves to one of the resource areas, picks up one pellet at a time, and delivers it to the nest. It then tries to go back to the resource area to collect food again. Each agent shuttles between a food source and the nest. If the agent is close enough to the center of the nest, then it drops the carrying object. Similarly, if it is close enough to the resource center, then it can pick up a food pellet.

If there is no collision avoidance, a robot moves with its maximum speed. The maximum forward speed of fast-moving robots is 2 cm per time step and half speed for slow-moving robots. A robot is also mounted with GPS to monitor the current position relative to the resource or the nest. We assume that the robot has the location information of resource areas and the nest in advance. As a result, robots do not waste time in finding the nest or the food sources. We assume that robots can distinguish fast-moving and slow-moving robots with a vision camera, since each robot has its own marker light on the top, depending on its

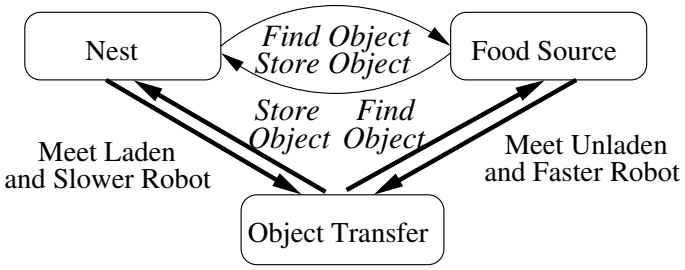


Figure 5.18: Representation of robot behaviors using task partitioning based on the difference in moving speeds; the thick solid lines indicate a task partitioning strategy that an object is transferred from the slower to the faster individual.

speed class.

5.4.3 Task selection mechanism

We used a task partitioning strategy inspired by ants' task partitioning based on body size [185, 7]. A large ant can take a food item from a smaller ant but not from a larger one, and large ants are often faster than smaller ants. Here, the task partitioning strategy based on the moving speeds of robots is briefly introduced in Figure 5.18.

A robot moves from a food source to the nest, carrying a food pellet. If a laden robot meets an unladen robot moving to the food source at a faster speed, the laden robot passes its currently carrying object to that unladen robot. We call this a BBs (bucket brigade strategy). The BBs passing an object directly from a slower agent to a faster agent occurs at this moment. The robot with the faster moving speed receives the object and transports it to the nest instead of the robot with the slower moving speed. The robot passing the object becomes unladen, and moves back to take another object in the resource area. If the laden robot sees another robot with much faster speed, it passes the item again to the faster robot.

In our proposed task partitioning strategy, the food pellet is directly transferred from the slower to the faster individual; we call it ascending-speed task partitioning or ascending-order BBs. For comparison, we consider another task partitioning that is performed in an opposite manner, that is, the object is transferred from the faster to the slower individual; we call it descending-speed task partitioning or descending-order BBs.

In the foraging task, if a laden robot comes close to an unloaded robot, task partitioning occurs by passing the object being carried to the other robot; the speed of the receiver robot

is higher in ascending BBs and lower in descending BBs. The results of the two groups, ascending-order BBs and descending-order BBs, are compared with a no-transfer group under various experimental conditions. For a no-transfer group (non-partitioning group), each robot picks up an object, moves to the nest, and drops the object at the nest without transferring to another robot.

We can consider another type of BBs in which a laden robot can pass the food item to any unladen robot that it meets on the trail, regardless of the robot speed. We call this no-order BB. We will compare the above BBs with no-order BBs to see whether the moving speed of agents is an important factor. The speed of robots can be chosen as one of two levels or multi-level. Initially we will test two levels of speed, high and low, for robot movements and later we will compare the performances with two levels and five levels of speed in robots.

5.4.4 Results with various environmental conditions

We simulate various experimental cases by changing the size of the nest entrance, the number of food sources, and the number of foraging robots. Then we analyze the effect of the BB process to reduce the traffic jam near the nest entrance. In addition, for comparison between partitioned tasks and non-partitioned tasks, we apply a task transition delay that is inevitable in the object transfer process. A robot should spend some delay time passing object to another robot in their interaction. We will check if the occurrences of collision avoidance decrease with task partitioning, and if task partitioning exhibits a better performance than non-partitioning with a no-transfer job.

Performance evaluation

Generally, the foraging performance is influenced by several factors: the number of food sources, the task transfer time between a pair of agents, the size of the nest entrance and the number of agents. Task partitioning does not always provide the best performance. According to our experimental results, which strategy will be helpful depends on the environmental conditions. Under the condition of heavy traffic congestion, task partitioning with BBs significantly improves the foraging performance and a fine-grained level of moving speeds in agents, i.e., more diverse characteristics of agents, can help the division of labor to some degree by setting up the object transfer between different speed levels of agents.

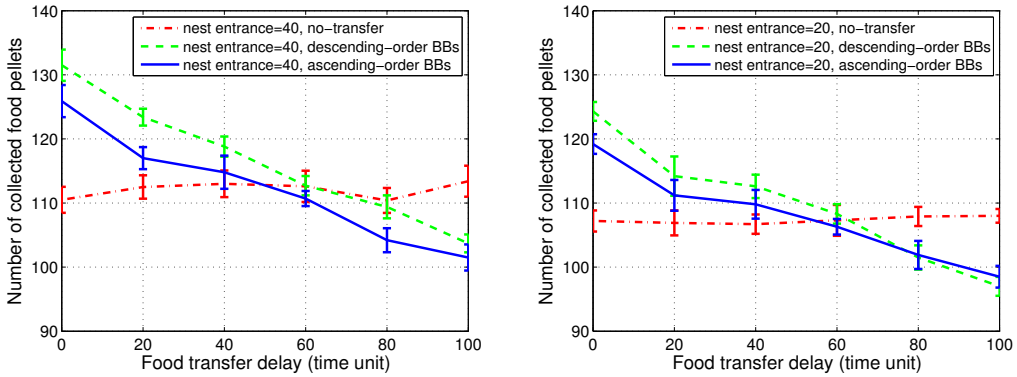
The performances of foraging task using various methods are evaluated by counting the number of collected objects at the nest depending on the object-transfer delay. We tested varying transfer delays, ranging from 0 to 100 time units at intervals of 20 time units. One simulation runs for 10,000 time steps and ten trials are tested for each experimental case to evaluate the average performance and its variance.

In addition, we provide the analysis of variance (ANOVA) models for statistical significance. Without any date transformation, a three-way ANOVA analysis was selected to investigate the effects of different factors (task partitioning method, size of nest entrance, and the number of robots) and their interactions among them. The statistical data is obtained correspond to the total number of collected food pellets of each food transfer delay time. The aim is to determine the differences and similarities among task partitioning methods in various experimental cases.

Size of nest entrance

In the first experiment, we tested varying sizes of the nest entrance in the foraging task with twelve robots. Here, we use only one food source located at (510, 375) cm (see Figure 2(a)). The radius of the nest entrance is set to 40 cm for a wide entrance and 20 cm for a narrow entrance. The performance is measured with the total number of collected objects at the nest. Figure 5.19 shows the averaged performance over ten trials with two task partitioning strategies and the no-transfer strategy with narrow and wide entrances of the nest, respectively. Dash-dot lines indicate no-transfer groups, that is, non-partitioning groups. Solid lines and dash lines display the results with ascending-order BBs and descending-order BBs groups, respectively. Figure 5.20 shows the results of performing a multiple comparison of the group means in Figure 5.19.

From the result, we see that the performances of the narrow nest entrance are worse than those of the wide nest entrance (see Figure 5.20(a)) and as the food transfer delay increases, the performances become worse and they linearly decrease in most of cases. This is an expected result because task partitioning groups should spend some amount of time on passing food, which delays food delivery to the nest. With a wide nest entrance, the foraging performance improves since the food pellets can be easily stored at the nest area by less traffic congestion, regardless of the applied strategies. Task partitioning groups, ascending-order BBs and descending-order BBs, have better performances than the non-partitioning group with a short food transfer time. If the food transfer time exceeds 60 time units, the



(a) (b)

Figure 5.19: Results of collected objects using 12 robots with task partitioning strategies and a non-partitioning strategy (blue solid line: ascending-order BBs, green dash line: descending-order BBs, red dash-dot line: no-transfer); one food source is available: (a) the radius of the nest entrance is 40 cm for a wide entrance; and (b) the radius of the nest entrance is 20 cm for a narrow entrance (the curves show the average performance with 95% confidence intervals by assuming t-distribution)

task partitioning becomes worse; see the cross point between the performance curves of the task partitioning groups and the non-partitioning group. The narrow nest entrance often allows for a larger transfer delay limit, which means that the task partitioning groups are more effective and efficient in collecting food with the narrow entrance.

Ascending-order BBs have a little worse performance than descending-order BBs (but not statistically significant for large transfer delays). With a single food source, ascending-order BBs have heavy traffic congestion at the source area since high-speed robots reach the source area early and then directly return to the nest in the first round without any food transfers. Slow robots reach the source later and the traffic congestion at the source area becomes intense. In contrast, slow robots with descending-order BBs receive the food items from high-speed robots in the first round while the high-speed robots with food are returning to the nest, and then the slow robots change direction towards the nest, which can reduce the traffic jam at the source. It ultimately increases the performance.

With a short object-transfer delay (for example, 20 time units), the performances of both ascending-order BBs and descending-order BBs are better than the no-transfer group in the above experiment with a single resource The allowable delay limit may change depending on the simulation environment. Nevertheless, we can say that task partitioning groups may

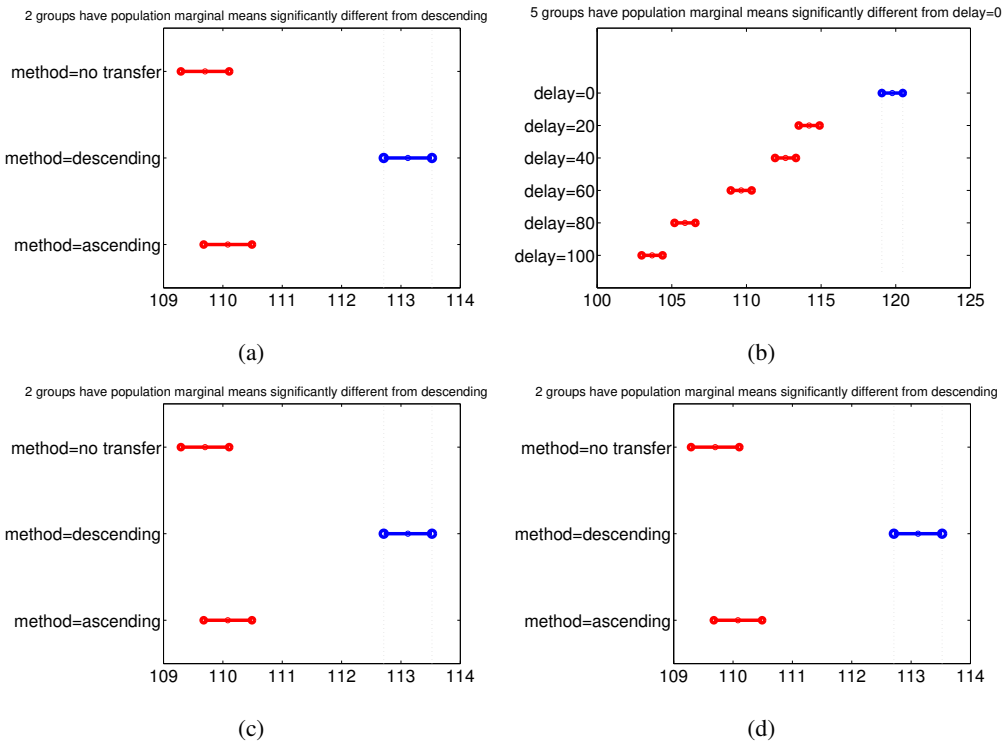


Figure 5.20: Performance results with four food resources (blue solid: ascending-order BBs, green dash: descending-order BBs) and non-partitioning group (red dash-dot); the radius of the nest is 20 cm (narrow entrance): (a) 12 robots; (b) 20 robots; (c) 40 robot; and (d) 60 robots

Table 5.1: ANOVA table for the results (T:Task partitioning strategy, S:Size of nest entrance, and F:Food transfer delay)

Source	Sum of Squares	Degree of freedom	Mean Squares	F	P-Value
T	840.9	2	420.43	58.25	2.4517E-22
S	2624.4	1	2624.4	363.6	6.9791E-55
F	10274.3	5	2054.85	284.69	3.1768E-116
T+S	106.4	2	53.2	7.37	0.0007
T+F	6092.9	10	609.29	84.41	4.9974E-84
S+F	66.5	5	13.31	1.84	0.104
T+S+F	123.7	10	12.37	1.71	0.0766
Error	2338.6	324	7.22		
Total	22467.6	359			

have improved foraging performances compared to the non-partitioning group, if a short task transition delay in the food transfer is available.

Table 5.1 summarizes the ANOVA results clearly showing dependent and independent variables. The ANOVA table shows that except the interaction between size of nest entrance and food transfer time, all others factors including second-order interactions are significant (p-value$<$5%). The p-value of each factor is extremely small enough to conclude that the mean responses are significantly.

Multiple food sources

We tested multiple food resources instead of a single source. Figure 5.21 shows the results of four food sources with 12, 20, 40, and 60 robots. The number of foraging robots is constant, but the number of food sources increases four times. The nest entrance is narrow with a radius of 20 cm. With four food sources, robots are divided equally into four sub-groups and each sub-group is assigned to one of four food sources. The number of robots moving to a food source decreases as the number of food sources increases. In this environment, the traffic congestion on the trail greatly decreases, but more traffic jams can occur at the nest area.

In the previous experiments, task partitioning groups have decreasing performances as the food transfer delay increases. Similar patterns are observed with multiple food sources as in Figure 5.21(a)-(b). The overall system performance improves with either ascending-order BBs or descending-order BBs with a short transfer delay. The effect with multiple resources is more greatly observed with the task partitioning methods, compared with the non-partitioning method. With more food sources, there are more chances to meet other robots near the nest area, but fewer chances near the food source areas or on the trail. With a narrow nest entrance, a traffic jam is found more often at the nest area with more food resources. Multiple food sources, however, distribute a group of robots to resources in different directions. Thus, task partitioning becomes more effective with multiple resources if a limited food-transfer time is available, e.g. less than 60 time units. With descending-order BBs, the task transfer occurs near the nest and the performance is relatively lower than the performance of ascending-order BBs. Table 5.2 shows the ANOVA results. Similar pattern with the previous ANOVA table is observed that only except the interaction between the number of robots and food transfer delay, all others factors are significant (p-value$<$5%).

Increasing the number of robots changes the pattern of foraging performance in the same environment with the fixed food sources and the constant size of nest entrance (see Figure

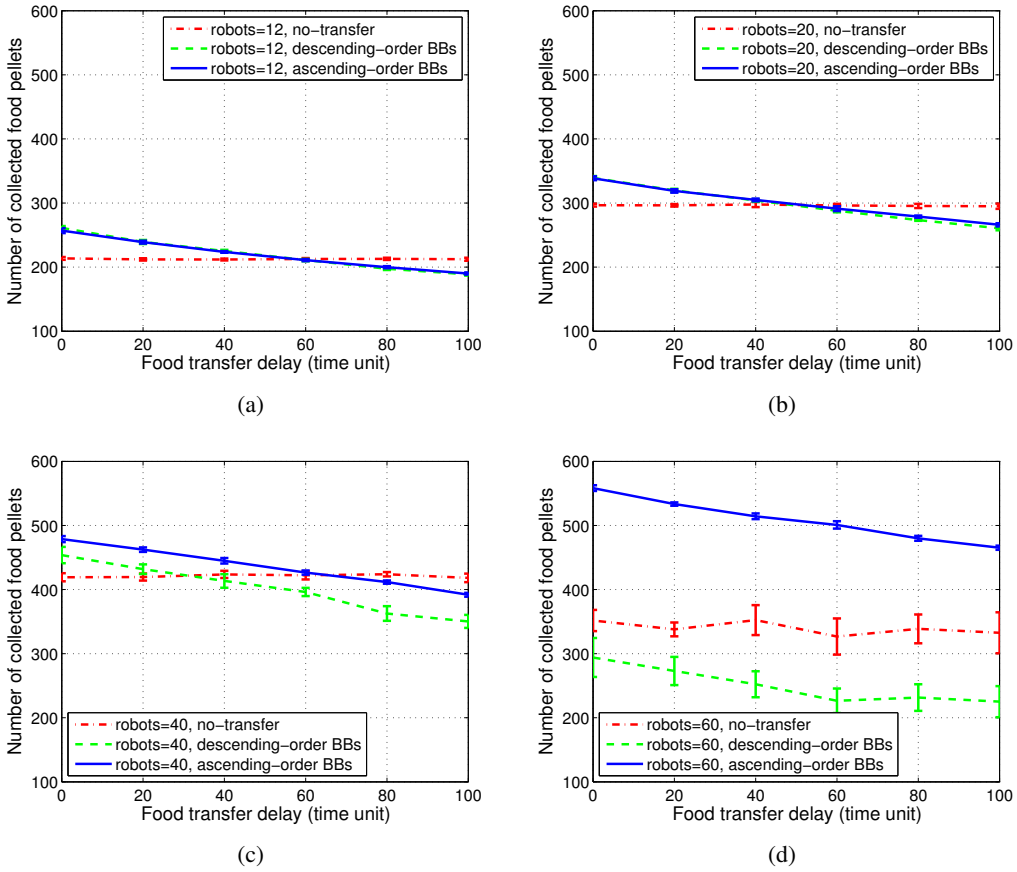


Figure 5.21: Performance results with four food resources (blue solid: ascending-order BBs, green dash: descending-order BBs) and non-partitioning group (red dash-dot); the radius of the nest is 20 cm (narrow entrance): (a) 12 robots; (b) 20 robots; (c) 40 robot; and (d) 60 robots.

5.21(c),(d)). Task partitioning groups show a very different pattern of performances unlike the previous experiments. Ascending-order BBs always exhibit better performance than the non-partitioning group, and descending-order BBs have the worst performance among the three strategies regardless of the task transfer delay. The average number of objects collected with 60 robots over varying object transfer delays is 337, 249 and 511 for the non-partitioning group, descending-order BBs and ascending-order BBs, respectively. The average performance compared to the non-partitioning group is improved by up to 152% with ascending-order BBs, and is decreased to 73.8% with descending-order BBs.

Table 5.2: ANOVA table for the results (T:Task partitioning strategy, N:Number of robots, and F:Food transfer delay)

Source	Sum of Squares	Degree of freedom	Mean Squares	F	P-Value
T	681260.6	2	340630.3	1270.98	5.2976E-225
N	4127309.2	3	1375769.7	5133.35	0.0000
D	231759.7	5	46351.9	172.95	9.7105E-117
T+N	1447170	6	241195	899.96	2.1862E-310
T+D	142205.4	10	114220.5	53.06	1.3260E-77
N+D	6514.7	15	434.3	1.62	0.0634
T+N+D	17774	30	592.5	2.21	0.0003
Error	7173668.2	648	268		
Total	6827661.6	719			

We observe the traffic congestion near the nest area greatly increases with a large number of robots. With the descending-order BBs the food transfer often occurs near the nest, which degrades the performance. During the object transfer between a pair of agents, the agents pause at the current position waiting for the completion of object transfer without moving at all. Neighboring agents consider those agents waiting for object transfer as obstacles and need to spend some additional time for collision avoidance. It then aggravates the traffic jam, and the foraging performance becomes worse than that of the non-partitioning group. In contrast, with the ascending-order BBs, food transfer often occurs on the trail. The traffic congestion near the nest is not intense, and it improves the foraging performance to a large extent.

Number of robots

From the above experiments, we see that task partitioning is not always the best choice. However, it could obtain an improved performance under the limited condition that there is a short food transfer delay between a pair of robots, or there are multiple food sources available. We tested varying numbers of robots with a single source or four food sources. More robots can cause more intense traffic jams and we observed a positive effect of task partitioning.

We considered task partitioning methods and non-partitioning method under various parameter conditions; the number of robots, the number of food sources, and the size of the nest entrance. The object transfer time is fixed to 50 time units and the number of robots ranges from 10 to 100. Figure 5.22 shows the performance results. When there is a single

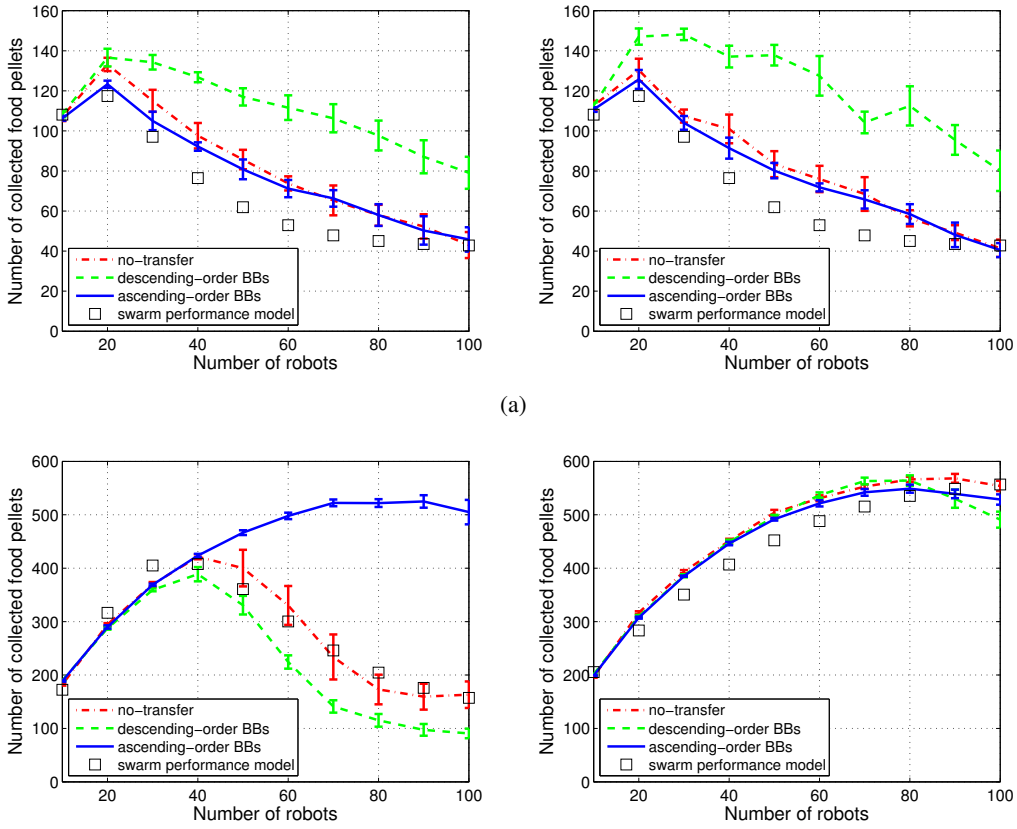


Figure 5.22: Performance results for collected objects depending on number of robots: (a) a single food source; and (b) four food sources. Object transfer time is set to 50 time units, and left and right plots represent narrow and wide entrances of the nest, respectively (the curves show the average performance with 95% confidence intervals (t-distribution)).

source available with more than 20 robots, the performance of descending-order BBs is always better than the other methods regardless of the size of the nest entrance (see Figure 5.22(a)). However, when the number of food sources is changed to four, this tendency changes drastically as shown in Figure 5.22(b).

Ascending-order BBs with a narrow nest entrance show significant improvement in some cases regardless of the delay time for object transfer. As the number of robots increases up to 60 robots, ascending-order BBs show two to three times better performance than the non-partitioning group, while descending-order BBs show a worse performance

than the non-partitioning group. However, with a wide nest entrance, similar performances are observed, regardless of task partitioning strategies. It seems that the density of robots, especially near the nest, is a very critical factor for choosing the appropriate strategy.

According to a model of swarm performance [80], we can evaluate the swarm performance depending on the swarm size. We applied the model to our foraging task. As shown in Figure 5.22, the model (square marks) can be fitted to the empirical data. We note that the performance of the non-partitioning group is closely fitted to the swarm performance model, since the model assumes the swarm system without cooperation.

5.4.5 Analysis

We have seen that task partitioning can be effective depending on the environmental conditions. In addition, how robots interact with each other influences the BBs (bucket brigades). Importantly, traffic congestion is deeply related to task partitioning strategies and the congestion patterns are changed according to the task partitioning strategies. We note that task partitioning may not always be the best choice in a given environment. The performance relies on many environmental conditions such as the size of the nest entrance, the number of routes to move around the nest as a result of several food sources, or the size of the swarm. We show the effect of the task partitioning strategies in more detail.

To see the effect of traffic jams on the foraging performance, we first observed a distribution of collision occurrences for each experiment. Figures 5.23 and 5.24 show locations where collision avoidance occurs when the number of robots is 60, the object-transfer delay is set to 50 time units, and the nest entrance is tested with a radius of 20 cm and 40 cm, respectively. Figure 5.23 displays collision occurrences when there is only one food source. With a single source, most robots are trapped near the food source and collisions are mostly found near the food source regardless of the nest entrance size. However, the collision patterns with descending-order BBs over a narrow nest entrance are quite different from those with the other methods. Many collisions are additionally found near the nest. With this strategy, robots moving faster arrive at the food source earlier and pass their food pellets to robots moving slower and closer to the nest. Before the traffic jam near the food source becomes heavy, there are more opportunities for robots to escape the traffic jam area, and more objects can be transported to the nest. Thus, descending-order BBs mostly have better foraging performance as shown in Figure 5.22(a) regardless of nest entrance size.

With a single food source, the descending-order BBs have significantly better perfor-

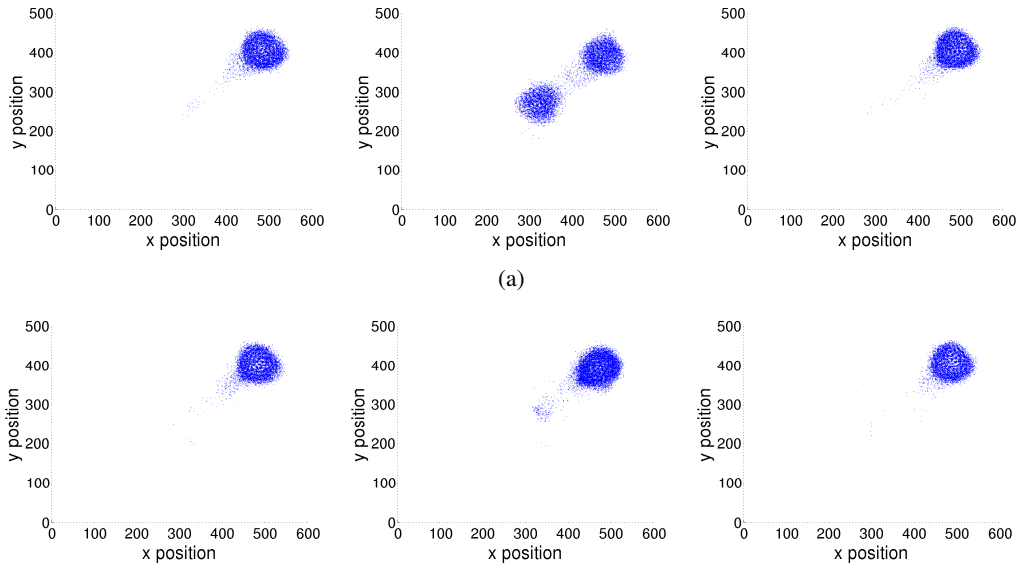
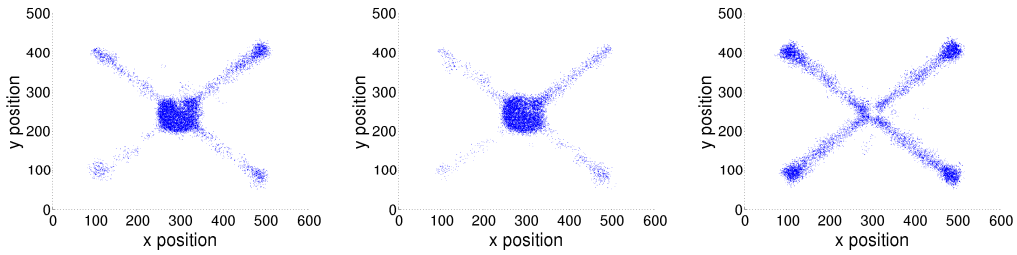


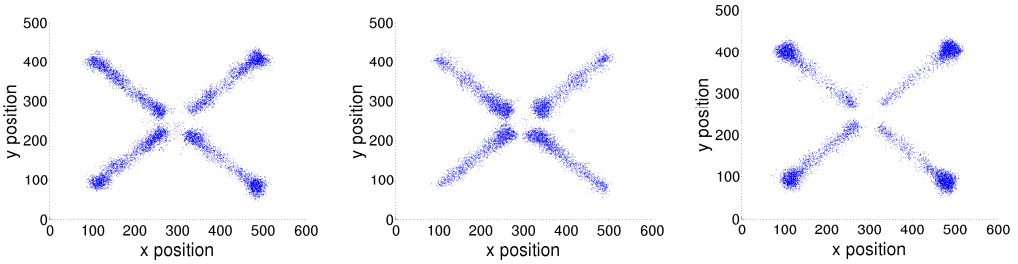
Figure 5.23: Distribution of collision occurrences with a single food source and 60 robots; the object transfer time is set to 50 time units: (a) radius of nest entrance is 20 cm (narrow nest entrance); and (b) radius of nest entrance is 40 cm (wide nest entrance) (left: no-transfer method, middle: descending-order BBs, right: ascending-order BBs).

mance than the ascending-order BBs. The ascending-order BBs are similar to or even worse than the non-partitioning group although it is not statistically significant. A heavy traffic jam is observed near the food source with this method, since there is only one food source available. However, when there are four food sources with narrow nest entrance, the pattern of collision occurrences is completely changed (see Figure 5.24). Collisions are frequently observed on the trail at the initial stage, and as time goes on, the collisions mostly occur at a central location around the nest. More food sources lead to several routes converging at the nest and the traffic jam becomes heavy around the nest. Most collisions are found near nests with narrow entrances, especially with descending-order BBs and the non-partitioning method as shown in Figure 5.24 and Figure 5.25.

Yet with ascending-order BBs, collisions are spread throughout the trail and near the food sources, but not near the nest. The traffic jam near the nest is lower and agents have more chances to transport food pellets from the food source to the nest. Figure 5.25 shows cumulative collision occurrences along the trail with 60 robots. We can see that the traffic

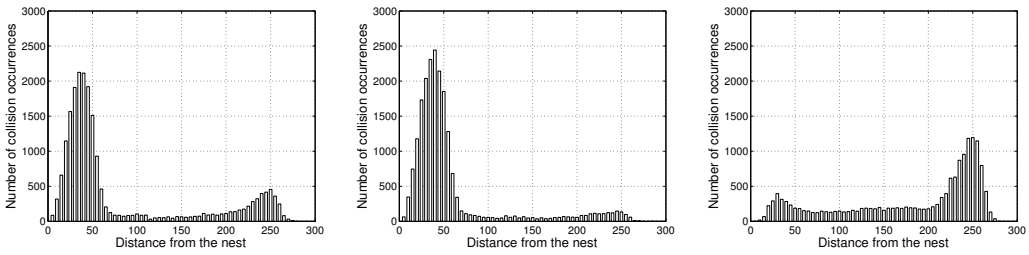


(a)



(b)

Figure 5.24: Distribution of collision occurrences with four food sources and 60 robots; the object transfer time is set to 50 time units: (a) radius of nest entrance is 20 cm (narrow nest entrance); and (b) radius of nest entrance is 40 cm (wide nest entrance) (left: no-transfer method, middle: descending-order BBs, right: ascending-order BBs).

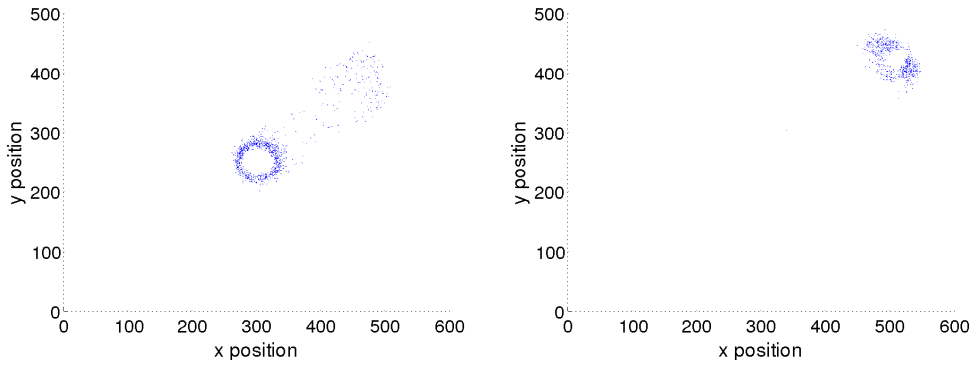


(a) (b) (c)

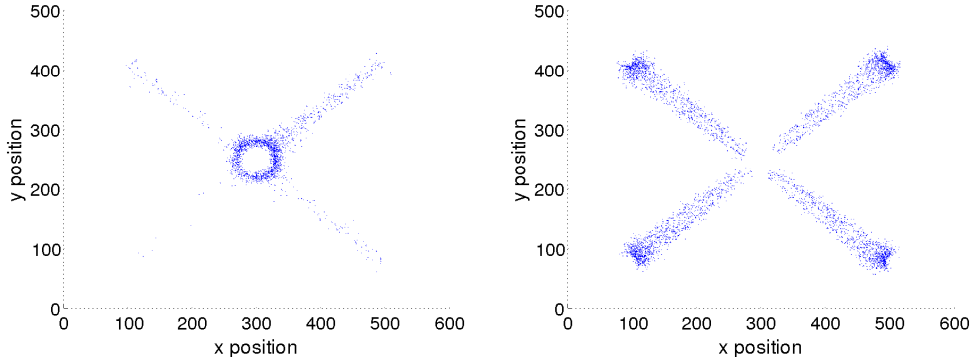
Figure 5.25: Cumulative collision occurrences along the trail: (a) non-partitioning group; (b) descending-order BBs; and (c) ascending-order BBs. 60 robots are tested for four food sources, and the nest entrance has a radius of 20 cm.

jam near the nest is lower with ascending-order BBs, compared with descending-order BBs and the non-partitioning methods. The total number of cumulative collision occurrences is inversely proportional to the number of collected objects.

For a non-partitioning group with no object transfer, collisions mostly occur near the



(a)

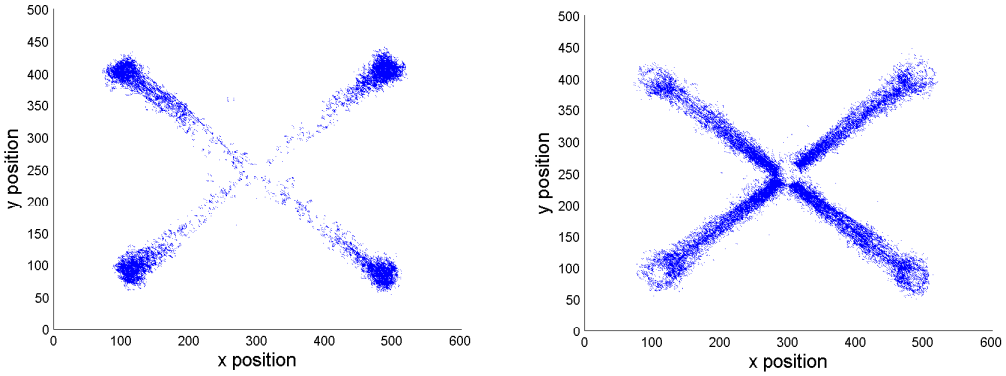


(b)

Figure 5.26: Location of object transfer occurrences with 60 robots: (a) one food source; and (b) four food sources. The object transfer delay is set to 50 time units and the nest entrance has a radius of 20 (left: descending-order BBs, right: ascending-order BBs).

nest and some collisions are found near the food source area. For descending-order BBs, more collisions are observed near the nest than the non-partitioning group. With the two methods, the food-collecting performance degrades due to traffic jams at the nest. The descending-order BBs exhibit an even worse performance than the non-partitioning method. With the method, fast agents pick up a food pellet in the resource area and pass it to slow agents, and slow agents should drop it at the nest area. They begin to stagger around the nest and experience traffic jam. Fast agents tend to come close to the nest to pass the food pellet to slow agents. Many agents get together near the nest and a lot of collisions occur.

Figure 5.26 shows the location of object transfer occurrences with ascending-order BBs and descending-order BBs for one food source or four food sources, respectively. With



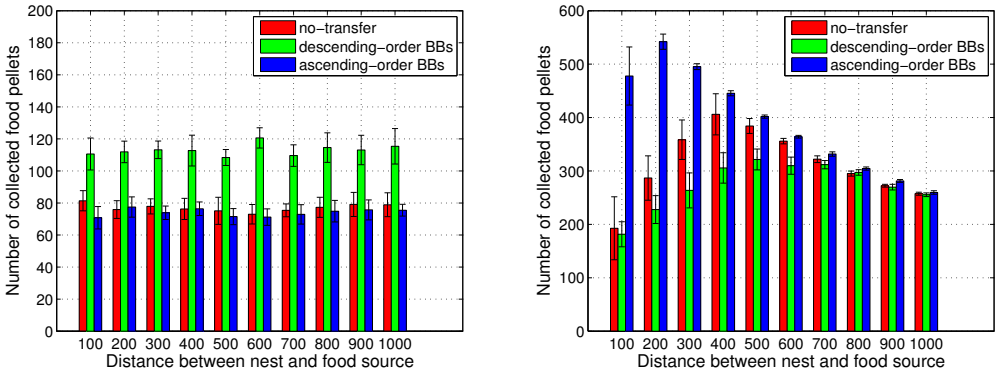
(a) (b)

Figure 5.27: Distribution of slow and fast agents with ascending-order BBs; the object transfer delay is set to 50 time units and the nest entrance has a radius of 20 with four food sources: (a) slow robots; and (b) fast robots.

descending-order BBs, object transfers mostly occur near the nest and agents are also congested near the nest area, irrespective of the number of food sources. This tendency with descending-order BBs aggravates the traffic jam even more than with the non-partitioning group in some cases. In contrast, with ascending-order BBs, object transfer occurrences are distributed uniformly on the trail for multiple food sources, but the occurrence-spreading effect is weak for a single food source.

We can easily see object transfers near the nest in Figure 5.26(b). With ascending-order BBs, fast agents receive an object from slow agents and deliver it to the nest. The object transfers occur almost everywhere on the trail. For ascending-order BBs, collisions near the nest are greatly reduced and collisions are spread almost uniformly across the trail and a little higher density is found near the food source areas, since some fast agents waiting for object transfer come close to the resource area. As a result, the ascending-order BBs improve the foraging performance significantly with more food sources and more robots. With a wide nest entrance, collisions rarely occur near the nest for any case as shown in Figure 5.24(b). The effect of the nest entrance size can be observed in figures. The traffic jam is shifted from the nest area to the food source areas, as the entrance size increases.

Figure 5.27 shows a distribution of slow and fast agents observed during a given time span with ascending-order BBs for four food sources. Slow agents move near the food sources to pick up food pellets, and fast agents tend to shuttle to deliver food between slow



(a) (b)

Figure 5.28: Number of collected objects in relation to the distance between the nest and the food source: (a) a single food source; and (b) four food sources; the nest has a radius of 20 cm, there are 60 robots, and the object-transfer delay is set to 50 time units.

agents and the nest. As a result, fast agents frequently cover most of the trail parts between food sources and slow agents if there is no much collision among robots. This is consistent with the pattern of object transfer occurrences shown in in Figure 5.26. With descending-order BBs, most of agents are crowded near the nest, leading to heavy traffic jam.

Figure 5.28 shows the number of collected objects in relation to the distance between the nest and the food sources. With a single food source, descending-order BBs show the best performance regardless of the distance. The performances with the three methods are not much changed and the same patterns are observed even with long distances. With four food sources, ascending-order BBs show the best performance. As the distance between the nest and the food source becomes shorter, the traffic jam near the nest worsens and ascending-order BBs helps reduce the traffic jam, thus improving the overall foraging performance. The performance of ascending-order BBs is significantly better than the other methods within a limited distance. It is deeply related to whether the delay time due to the traffic jam exceeds the object transfer delay between a pair of agents. With a long distance between the nest and a food source, the traffic jam rarely happens and there is no bucket brigade effect.

We observed the total time spent on food delivery for each robot with task partitioning. The overall performance is influenced by three types of behaviors: food transfer, moving on the trail and collision avoidance. We can easily infer that extra time is needed for food trans-

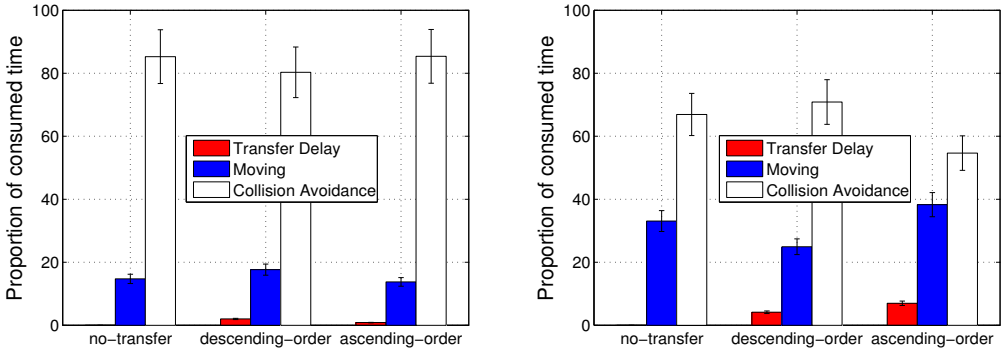
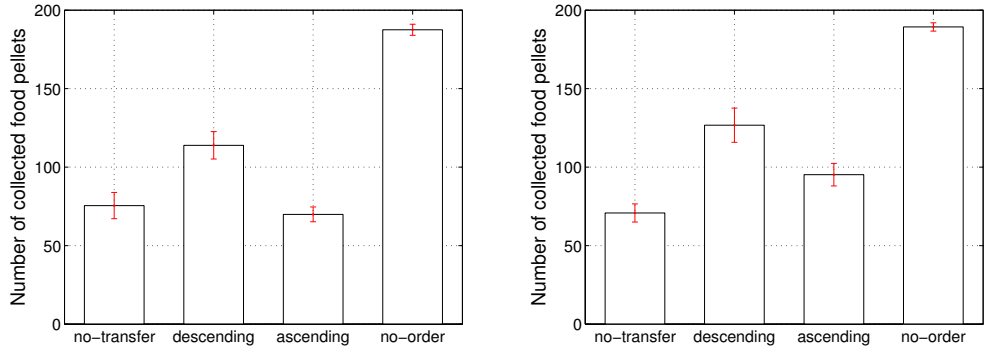


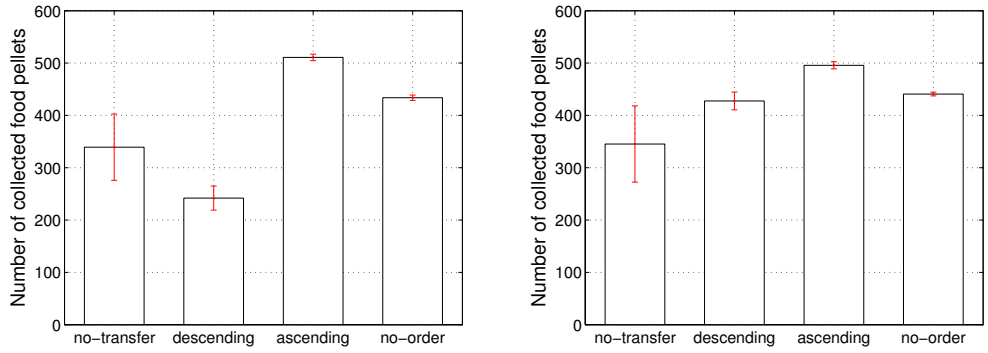
Figure 5.29: Proportion (%) of time spent in three behavior components with (a) one food source and (b) four food sources; the nest has a radius of 20 cm, there are 60 robots, and the object-transfer delay is set to 50 time units.

fer with task partitioning methods. Figure 5.29 shows a distribution of the average amount of time spent on each behavior. The red-colored, blue-colored and white-colored bars represent the consumed time for food transfer, moving and collision avoidance, respectively. Task partitioning groups spend some time on the food transfer behavior, but more time on the moving behavior depending on strategies and environmental conditions. In particular, descending-order BBs have a lengthy moving time in the experiments with one food source as shown in Figure 5.29(a) and ascending-order BBs have a lengthy moving time with four food sources as shown in Figure 5.29(b). The overall foraging performance in terms of how many objects are collected is directly related to the above temporal performance. The efficiency is strongly related to more time being spent on the moving behavior and less time on collision avoidance.

So far we assumed two levels of speed movement for robots, high or low. We investigated whether a greater variety of robot characteristics can affect the BBs performance. That is, robots can have fine-grained levels of speed for their movement; five levels are tested here. The ascending-order BB with two levels of speed (1 or 2 cm per time unit) or five levels of speed (1 cm, 1.25 cm, 1.5 cm, 1.75 cm or 2 cm per time unit) was applied. The BBs with different speed orderings have been tested. In this experiment, another strategy with no-order BBs was tested in task partitioning. For no-order BBs, object transfer occurs unconditionally between a pair of robots. If a laden robot carrying a food pellet meets an



(a)



(b)

Figure 5.30: Comparison results for the non-partitioning method, ascending-order BBs, descending-order BBs and no-order BBs: (a) one food source; and (b) four food sources. Sixty robots were tested and the object transfer time is set to 50 time units. Robots have two-level speeds (left) or five-level speeds (right).

unladen robot, it passes the item being carried regardless of the speed of the robots.

Figure 5.30 shows the comparison results for the non-partitioning method, ascending-order BBs, descending-order BBs and no-order BBs. As shown in Figure 5.30(a), five levels of speed in robots can significantly improve the foraging performance for ascending-order BBs with a single food source. No-order BBs maintain almost the same performance regardless of the speed level. For four food sources, only descending-order BBs significantly improve the foraging performance with five levels of speed in robots (see Figure 5.30(b)). No-order BBs exhibit the best performance for a single source, but not for four food sources. Frequent task changes occur with this approach, and this may degrade the performance due

to the food-transfer delay. For four food sources, ascending-order BBs show the best performance regardless of the speed levels. With this method, more fine-grained speed levels have no effect on the foraging performance.

We note that multi-level task partitioning by speed levels has a positive effect on reducing traffic jams and improves the foraging performance in most of cases. The diverse characteristics of robots can further distribute the robots under traffic congestion throughout the trail with a bucket brigade, and thus a bucket brigade can reduce the traffic congestion. Furthermore, ordering the robots based on the environmental situation greatly influences the foraging performance. Ascending-order BBs help diminish the traffic jam near the nest area.

From the simulation experiments, we can say that if the total amount of food-transfer time between a pair of agents is much smaller than the total collision delay due to a traffic jam, task partitioning can be a good choice in the foraging task. This might explain why several social insects use task partitioning to deliver materials or food for their colony's survival. Generally, many social insect colonies consist of a number of members so that the entrance of a colony nest always suffers from heavy traffic jams. However, for protection against attacks from their enemies, the insects want to maintain a small entrance size as possible. We have seen that task partitioning provides an efficient organization system for the foraging task with a narrow entrance of the nest. Task partitioning plays an effective role in the foraging performance, especially with heavy traffic congestion. The above results may support the hypothesis that task partitioning strategies can improve the foraging performance for the colony's survival.

In spite of the benefit of task partitioning, many ants rarely use BBs (bucket brigades) on their entire pheromone trail. Their trail for food delivery is relatively long and there is not much intense traffic congestion. This is similar to a situation where a small number of agents run to deliver objects in our experiments. In that case, the non-partitioning method is better for the foraging performance than task partitioning methods. With the non-partitioning method, each agent can simply take an object in a resource area and return to the nest, carrying the item without transferring it to another agent. This will prevent any delays in the object transfer between agents. Task partitioning may possibly be needed in the nest cave of ant underground, where the path is very narrow and a large traffic jam is expected.

A traffic jam near the central area is often observed in an environment with multiple resources, several routes converging to the nest, and a narrow nest entrance. In that situation, the foraging performance can be greatly improved by task partitioning based on ascending-

order BBs where the object is transferred from the slower to the faster agent. We found that the proposed approach distributes agents from the overcrowded region near the nest to the trail, thus reducing the traffic jam. Interestingly, it was reported that seed-harvesting ants take bucket brigades similar to ascending-order BBs, that is, workers are sequenced from slower ants (near the food source) to faster ants (near the nest) [185]. Anderson et al. (2002) argued that under the following two assumptions, a sequence of individuals from the smallest (nearest to the source) to the largest (nearest the nest) can be generated as a self-organized process: 1) larger ants are faster than smaller ants and 2) a large ant can take a food item from a smaller ant but not from a larger one. Also, leaf-cutter ants *Atta colombica* demonstrate that post-transfer transportation speeds are significantly faster than pre-transfer [8]. Our experimental results are consistent with biological data found in social insects.

5.5 Summary of Chapter 5

In this chapter, we propose a self-organized method to allocate a swarm of agents to perform sequential subtasks. A complex task can be decomposed into a series of subtasks to be performed sequentially. Each individual agent performs one of multiple tasks with according to simple behavioral rules and local information about neighboring agents. Even for multiple sequential subtasks, we show that the proposed response threshold model can be applied to regulate the proportion of agents in order to meet the task demands for each subtask. Without the help of a history queue, instant information is used. Robot's individual perception is regulated by the relative difference between the number of tasks not completed and the number of robots performing the corresponding subtask. Various experimental results show that the proposed method satisfies the characteristics of swarm intelligence: robustness, scalability, and flexibility.

Sequential tasks are common in nature, and we demonstrate the effect of a task partitioning strategy called bucket brigade that uses the direct transfer of materials or food between a pair of workers. We propose a task partitioning strategy based on agents' moving speeds for the foraging task. It was observed that *M. barbarus* is generally sequenced from slowest (near the food source) to fastest (near the nest) as a bucket brigade arrangement [185]. We test various environmental conditions and compare the performances between task partitioning groups and non-partitioning groups. Our results, we show that task partitioning may not always be the best solution for foraging performance. However, when there

is a transfer bottleneck at a central location, such as at the entrance of the nest, task partitioning can be an effective strategy for reducing the traffic jam and improving the overall foraging performance of a group.

Chapter 6

Conclusions

In this chapter, the contributions of this thesis are presented and suggestions for future work are provided.

6.1 Contribution

The goal of the proposed methods is to determine how the individuals in a swarm are allocated to each subtask in order to maximize the overall system performance. Generally, global information about the surrounding environment is needed. However, in multi-agent systems without a centralized solution, a self-regulated decision-making strategy is needed. Each agent only uses its own effectors and sensors, and determines which task should be performed with only a limited view of the environment. Emergent coordination algorithms that use only local sensing and no direct communication between agents are very attractive because they are robust and scalable. In this dissertation, swarm intelligence has been applied to design a task allocation problem for swarm robotic systems. Each individual robot has equal capability to execute tasks, and the goal is to make each individual robot select an optimal task based on its local information. No communication (or only local communication) between robots is available.

The concept of swarm intelligence has served as an inspiration to collective robotics and several social insects are the source of that inspiration due to their self-organization and self-

regulation capabilities in colonies. We derive task selection function based on the response threshold model. Each robot autonomously decides whether to switch between tasks using the local information available to it individually. In addition, we propose a new response threshold update algorithm, in which an individual perception is regulated by the relative difference between task need and the number of agents performing tasks. The desired task allocation within the group is obtained from the behavior of the individual agents in a self-organized manner. Repetitive and continuous task selection leads to a desired group-level task distribution. Transition rates are regulated adaptively depending on the environment, producing rapid produces fast convergence to the desired state and a little task transition at the equilibrium state. We also show that this process can lead to convergence to the equilibrium of task allocation distribution.

We also demonstrate the effect of a task partitioning strategy, in which objects are transferred directly between agents. In our simulations, we investigated bucket brigade strategies for the foraging task and found that traffic congestion is an important factor that influences foraging performance. Ordering agents with respect to speed level for bucket brigades significantly influences foraging performance. In an environment with multiple resources, several routes converging to the nest, and a narrow nest entrance, the bucket brigade sequenced from the slowest agents (near the food source) to the fastest agents (near the nest) can particularly improve the performance in regions with traffic congestion near the nest. Generally, many social insect colonies consist of a number of members, and the entrances of colony nests always suffer from heavy traffic congestion. Our experimental results support the hypothesis that several social insects use one of these task partitioning strategies based on bucket brigades in their foraging tasks. The main contributions of this thesis are

Swarm intelligence based task allocation algorithm Based on the response threshold model, this thesis presents a task allocation method without a centralized solution.

Enhancement to the algorithm Based on local information about task need and task type assigned to neighboring agents, the results shows that the algorithm performs better than the results only using task need.

Proofs of equilibrium task distribution This thesis presents a mathematical model for the algorithms to improve the theoretical base and to explain convergence behaviors.

Various simulations demonstrate an effect of a task partitioning strategy The results

shed light on why several social insects used various task partitioning method.

6.2 Further work

The proposed method can be extended to more complex problems and further studies could be performed to improve performance.

Changeable task allocation strategy: Leaf-cutting ants use various strategies to handle materials such as food and garbage [83, 39]. When an ant finds a food source, it selects one of three strategies. The first strategy is to simply return to the nest, carrying the food by itself; in this case, the ant has no task partitioning strategy. The second strategy is for the forager ant to directly pass the carried food to another ant. Finally, the third possible strategy is for the ant to drop the carried food in the middle of a pheromone trail so that other moving ants can bring it to the nest through indirect transfer. These ants freely change their strategies, and this could be a possible extension of the work. It would be interesting to investigate an automatic method to improve performance depending on the environmental situation.

Physiology analysis: Task allocation among member in a group is shown well in this dissertation and there are many related studies pertaining to this topic. However, most studies concern how they define an improved task selection function and obtain a proper threshold. There is no consideration about how the threshold value is regulated and a specific task is selected in a physiology aspect. Spiking neural network could be a good solution to this problem [132].

Desynchronized response threshold model: Various simulation results show that a swarm using the proposed method is able to adaptively allocate individuals to sequential subtasks, and we can conclude that the method is adaptive as the robots successfully re-distribute the agents to subtasks even if the environmental conditions change. This model functions well if the threshold value fluctuates; however, if the threshold value is completely same or very close, this may cause oscillatory phenomena that induce a tipping effect. Accounting for oscillatory phenomena could improve performance.

References

[1] Mohammad Amin Adibi and Jamal Shahrabi. A clustering-based modified variable neighborhood search algorithm for a dynamic job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 70(9-12):1955–1961, 2014.

[2] William Agassounon and Alcherio Martinoli. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *The first international joint conference on Autonomous agents and multiagent systems: part 3*, pages 1090–1097. ACM, 2002.

[3] Devanshu Agrawal and Istvan Karsai. The mechanisms of water exchange: the regulatory roles of multiple interactions in social wasps. *PloS one*, 11(1):e0145560, 2016.

[4] Jean-Marc Amé, José Halloy, Colette Rivault, Claire Detrain, and Jean Louis Deneubourg. Collegial decision making based on social amplification leads to optimal group formation. *Proceedings of the National Academy of Sciences*, 103(15):5835–5840, 2006.

[5] Patrick R Amestoy, Iain S Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[6] Christos Ampatzis, Elio Tuci, Vito Trianni, and Marco Dorigo. Evolution of signaling in a multi-robot system: Categorization and communication. *Adaptive Behavior*, 16(1):5–26, 2008.

[7] C Anderson, Jacobus Jan Boomsma, and JJ Bartholdi III. Task partitioning in insect societies: bucket brigades. *Insectes Sociaux*, 49(2):171–180, 2002.

[8] C. Anderson and J.L.V Jadin. The adaptive benefit of leaf transfer in *Atta colombica*. *Insectes Sociaux*, 48:404–405, 2001.

[9] C Anderson and FLW Ratnieks. Task partitioning in insect societies: novel situations. *Insectes sociaux*, 47(2):198–199, 2000.

[10] Ronald C Arkin, Tucker Balch, and Elizabeth Nitz. Communication of behavorial state in multi-agent retrieval tasks. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 588–594. IEEE, 1993.

[11] Gürdal Arslan, Jason R Marden, and Jeff S Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):584–596, 2007.

[12] M Emin Aydin and Ercan Öztemel. Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, 33(2):169–178, 2000.

[13] Gianluca Baldassarre, Stefano Nolfi, and Domenico Parisi. Evolving mobile robots able to display collective behaviors. *Artificial life*, 9(3):255–267, 2003.

[14] Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y Hadaegh. Inhomogeneous markov chain approach to probabilistic swarm guidance algorithm. In *5th Int. Conf. Spacecraft Formation Flying Missions and Technologies*, 2013.

[15] Ralph Beckers, Jean-Louis Deneubourg, Simon Goss, and Jacques M Pasteels. Collective decision making through food recruitment. *Insectes sociaux*, 37(3):258–267, 1990.

[16] Madeleine Beekman, Amy L Gilchrist, Michael Duncan, and David JT Sumpter. What makes a honeybee scout? *Behavioral Ecology and Sociobiology*, 61(7):985–995, 2007.

[17] Randall D Beer and John C Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive behavior*, 1(1):91–122, 1992.

[18] Gerardo Beni. From swarm intelligence to swarm robotics. *Swarm robotics*, pages 1–9, 2005.

[19] Spring Berman, Ádám Halász, M Ani Hsieh, and Vijay Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, 25(4):927–937, 2009.

[20] Samuel N Beshers and Jennifer H Fewell. Models of division of labor in social insects. *Annual review of entomology*, 46(1):413–440, 2001.

[21] SN Beshers, ZY Huang, Y Oono, and GE Robinson. Social inhibition and the regulation of temporal polyethism in honey bees. *Journal of Theoretical Biology*, 213(3):461–479, 2001.

[22] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*, volume 4. Oxford university press New York, 1999.

[23] Eric Bonabeau, Andrej Sobkowski, Guy Theraulaz, and Jean-Louis Deneubourg. Adaptive task allocation inspired by a model of division of labor in social insects. In *BCEC*, pages 36–45, 1997.

[24] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 263(1376):1565–1569, 1996.

[25] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. Fixed response thresholds and the regulation of division of labor in insect societies. *Bulletin of Mathematical Biology*, 60(4):753–807, 1998.

[26] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swram robotics:a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[27] Barry L Brumitt and Anthony Stentz. Dynamic mission planning for multiple mobile robots. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 3, pages 2396–2401. IEEE, 1996.

[28] Arne Brutschy, Giovanni Pini, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous agents and multi-agent systems*, 28(1):101–125, 2014.

[29] Nicholas W Calderone and ROBERT E Page Jr. Temporal polyethism and behavioural canalization in the honey bee, *Apis Mellifera*. *Animal behaviour*, 51(3):631–643, 1996.

[30] Ken Caluwaerts, Mariacarla Staffa, Steve N'Guyen, Christophe Grand, Laurent Dollé, Antoine Favre-Félix, Benoˆıt Girard, and Mehdi Khamassi. A biologically inspired meta-control navigation system for the psikharpax rat robot. *Bioinspiration & biomimetics*, 7(2):025009, 2012.

[31] Scott Camazine. *Self-organization in biological systems*. Princeton University Press, 2003.

[32] Alexandre Campo and Marco Dorigo. Efficient multi-foraging in swarm robotics. In *Advances in Artificial Life*, pages 696–705. Springer, 2007.

[33] Alexandre Campo, Simon Garnier, Olivier Dédriche, Mouhcine Zekkri, and Marco Dorigo. Self-organized discrimination of resources. *PLoS One*, 6(5):e19888, 2011.

[34] Mike Campos, Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, 8(2):83–95, 2000.

[35] Junwei Cao. Self-organizing agents for grid load balancing. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 388–395. IEEE, 2004.

[36] Y Uny Cao, Alex S Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous robots*, 4(1):7–27, 1997.

[37] David Castanon and Cynara Wu. Distributed algorithms for dynamic reassignment. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 1, pages 13–18. IEEE, 2003.

[38] Eduardo Castello, Tomoyuki Yamamoto, Yutaka Nakamura, and Hiroshi Ishiguro. Task allocation for a robotic swarm based on an adaptive response threshold model. In *Control, Automation and Systems (ICCAS), 2013 13th International Conference on*, pages 259–266. IEEE, 2013.

[39] JM Cherrett. The foraging behaviour of atta cephalotes l.(hymenoptera, formicidae). *The Journal of Animal Ecology*, pages 387–403, 1968.

[40] Vincent A Cicirello and Stephen F Smith. Wasp nests for self-configurable factories. In *The fifth international conference on Autonomous agents*, pages 473–480. ACM, 2001.

[41] Vincent A Cicirello and Stephen F Smith. Wasp-like agents for distributed factory coordination. *Autonomous Agents and Multi-agent systems*, 8(3):237–266, 2004.

[42] Colin W Clark and Marc Mangel. The evolutionary advantages of group foraging. *Theoretical population biology*, 30(1):45–75, 1986.

[43] Margaret J Couvillon, Jennifer M Jandt, NHI Duong, and Anna Dornhaus. Ontogeny of worker body size distribution in bumble bee (bombus impatiens) colonies. *Ecological entomology*, 35(4):424–435, 2010.

[44] Iain D Couzin and Nigel R Franks. Self-organized lane formation and optimized traffic flow in army ants. *Proceedings of the Royal Society of London B: Biological Sciences*, 270(1511):139–146, 2003.

[45] Iain D Couzin, Jens Krause, Nigel R Franks, and Simon A Levin. Effective leadership and decision-making in animal groups on the move. *Nature*, 433(7025):513–516, 2005.

[46] Rongxin Cui, Ji Guo, and Bo Gao. Game theory-based negotiation for multiple robots task allocation. *Robotica*, 31(06):923–934, 2013.

[47] Yash Daultani, Sushil Kumar, Omkarprasad S Vaidya, and Manoj K Tiwari. A supply chain network equilibrium model for operational and opportunism risk mitigation. *International Journal of Production Research*, 53(18):5685–5715, 2015.

[48] E De Margerie, S Lumineau, C Houdelier, and MA Richard Yris. Influence of a mobile robot on the spatial behaviour of quail chicks. *Bioinspiration & biomimetics*, 6(3):034001, 2011.

[49] Alan Oliveira de Sá, Nadia Nedjah, and Luiza de Macedo Mourelle. Distributed efficient localization in swarm robotic systems using swarm intelligence algorithms. *Neurocomputing*, 172:322–336, 2016.

[50] Nazlı Demir, Utku Eren, and Behçet Açıkmeşe. Decentralized probabilistic density control of autonomous swarms with safety constraints. *Autonomous Robots*, 39(4):537–554, 2015.

[51] Cl Detrain and JM Pasteels. Caste differences in behavioral thresholds as a basis for polyethism during food recruitment in the ant, pheidole pallidula (nyl.)(hymenoptera: Myrmicinae). *Journal of insect behavior*, 4(2):157–176, 1991.

[52] Gianni Di Caro and Marco Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[53] M Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[54] Marco Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992.

[55] Marco Dorigo, Gianni Di Caro, and Luca Maria Gambardella. Ant algorithms for discrete optimization. *Artificial life*, 5(2):137–172, 1999.

[56] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, et al. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71, 2013.

[57] Frederick Ducatelle, Gianni A Di Caro, Carlo Pinciroli, Francesco Mondada, and Luca Gambardella. Communication assisted navigation in robotic swarms: self-organization and cooperation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4981–4988. IEEE, 2011.

[58] Karthik Elamvazhuthi, Vaibhav Deshmukh, Matthias Kawski, and Spring Berman. Mean-field controllability and decentralized stabilization of markov chains, part i: Global controllability and rational feedbacks. *IEEE Conference on Decision and Control (CDC)*, 2017.

[59] Nourhan Elsayed and Khaled Al-Wahedi. Utilizing task partitioning for self-organized allocation of partially sequential tasks. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pages 3030–3035. IEEE, 2015.

[60] M. Erbas, A. Winfield, and L. Bull. Embodied imitation-enhanced reinforcement learning in multi-agent systems. *Adaptive Behavior*, 22(1):31–50, 2013.

[61] Maria Pia Fanti, Agostino Marcello Mangini, and Walter Ukovich. A quantized consensus algorithm for distributed task assignment. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 2040–2045. IEEE, 2012.

[62] Eliseo Ferrante, Ali Emre Turgut, Edgar Duéñez-Guzmán, Marco Dorigo, and Tom Wenseleers. Evolution of self-organized task specialization in robot swarms. *PLoS computational biology*, 11(8):e1004273, 2015.

[63] Eliseo Ferrante, Ali Emre Turgut, Edgar Duenez-Guzman, Marco Dorigo, and Tom Wenseleers. Evolution of self-organized task specialization in robot swarms. *PLos Computational Biology*, 11(8):e1004273, 2015.

[64] Jennifer H Fewell and Susan M Bertram. Division of labor in a dynamic environment: response by honeybees (apis mellifera) to graded changes in colony pollen stores. *Behavioral ecology and sociobiology*, 46(3):171–179, 1999.

[65] TD Fitzgerald and SC Peterson. Cooperative foraging and communication in caterpillars. *BioScience*, pages 20–25, 1988.

[66] Harold G Fowler and SW Robinson. Foraging by atta sexdens (formicidae: Attini): seasonal patterns, caste and efficiency. *Ecological Entomology*, 4(3):239–247, 1979.

[67] Simon Garnier, Christian Jost, Raphaël Jeanson, Jacques Gautrais, Masoud Asadpour, Gilles Caprari, and Guy Theraulaz. Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots. In *ECAL*, volume 3630, pages 169–178. Springer, 2005.

[68] Veysel Gazi and Kevin M Passino. A class of attractions/repulsion functions for stable swarm aggregations. *International Journal of Control*, 77(18):1567–1579, 2004.

[69] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[70] A. Giagkos and Myra S Wilson. Swarm intelligence to wireless ad hoc networks: adaptive honeybee foraging during communication sessions. *Adaptive Behavior*, 21(6):501–515, 2013.

[71] Fatma Pinar Goksal, Ismail Karaoglan, and Fulya Altiparmak. A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering*, 65(1):39–53, 2013.

[72] Dani Goldberg and Maja J Mataric. Design and evaluation of robust behavior-based controllers for distributed multi-robot collection tasks. In *Robot teams: From diversity to polymorphism*. Citeseer, 2001.

[73] David E Goldberg. Genetic algorithms in search, optimization, and machine learning, 1989. *Reading: Addison-Wesley*, 1989.

[74] Harry Goldingay and Jort van Mourik. Distributed sequential task allocation in foraging swarms. In *Self-Adaptive and Self-Organizing Systems (SASO), 2013 IEEE 7th International Conference on*, pages 149–158. IEEE, 2013.

[75] Deborah M Gordon. The organization of work in social insect colonies. *Nature*, 380(6570):121–124, 1996.

[76] Roderich Groß, Shervin Nouyan, Michael Bonani, Francesco Mondada, and Marco Dorigo. Division of labour in self-organised groups. In *From Animals to Animals 10: Proceedings of the Fourth International Conference of Simulation of Adaptive Behavior*, pages 426–436. Springer, 2008.

[77] Qinglin Guo and Ming Zhang. A novel approach for multi-agent-based intelligent manufacturing system. *Information Sciences*, 179(18):3079–3090, 2009.

[78] Martin T Hagan, Howard B Demuth, Mark H Beale, et al. *Neural network design*. Pws Pub. Boston, 1996.

[79] Adám Halász, M Ani Hsieh, Spring Berman, and Vijay Kumar. Dynamic redistribution of a swarm of robots among multiple sites. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2320–2325. IEEE, 2007.

[80] Heiko Hamann. Towards swarm calculus: Urn models of collective decisions and universal properties of swarm performance. *Swarm Intelligence*, 7(2-3):145–172, 2013.

[81] Heiko Hamann, Istvan Karsai, and Thomas Schmickl. Time delay implies cost on task switching: A model to investigate the efficiency of task partitioning. *Bulletin of mathematical biology*, 75(7):1181–1206, 2013.

[82] Julia Handl and Bernd Meyer. Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95–113, 2007.

[83] Adam G Hart and Francis LW Ratnieks. Task partitioning, division of labour and nest compartmentalisation collectively isolate hazardous waste in the leafcutting ant atta cephalotes. *Behavioral Ecology and Sociobiology*, 49(5):387–392, 2001.

[84] Manfred Hartbauer and Heiner Römer. A novel distributed swarm control strategy based on coupled signal oscillators. *Bioinspiration & biomimetics*, 2(3):42, 2007.

[85] D Hernández, H Trejo, and E Ordoñez. Development of an exploration land robot using low-cost and open source platforms for educational purposes. In *Journal of Physics: Conference Series*, volume 582, page 012007. IOP Publishing, 2015.

[86] DJ Hoare, Jens Krause, Nina Peuhkuri, and J-GJ Godin. Body size and shoaling in fish. *Journal of Fish Biology*, 57(6):1351–1366, 2000.

[87] Bert Hölldobler and Edward O Wilson. *The ants*. Harvard University Press, 1990.

[88] Bert Hölldobler and Edward O Wilson. *The superorganism: the beauty, elegance, and strangeness of insect societies*. WW Norton & Company, 2009.

[89] M Ani Hsieh, Ádám Halász, Spring Berman, and Vijay Kumar. Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence*, 2(2):121–141, 2008.

[90] Jinwen Hu, Lihua Xie, Jun Xu, and Zhao Xu. Multi-agent cooperative target search. *Sensors*, 14(6):9408–9428, 2014.

[91] Zhi-Yong Huang and Gene E Robinson. Regulation of honey bee division of labor by colony age demography. *Behavioral Ecology and Sociobiology*, 39(3):147–158, 1996.

[92] Yusuke Ikemoto, Toru Miura, and Hajime Asama. Adaptive division of labor control for robot group. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2409–2414. IEEE, 2009.

[93] Aleksandar Jevtić and Alvaro Gutiérrez. Distributed bees algorithm parameters optimization for a cost efficient target allocation in swarms of robots. *Sensors*, 11(11):10880–10893, 2011.

[94] Long Jin and Shuai Li. Distributed task allocation of multiple robots: A control perspective. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016.

[95] C. Jones and M.J. Mataric. Adaptive division of labor in large-scale minimalist multi-robot systems. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1969–1974. IEEE, 2003.

[96] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[97] Nidhi Kalra and Alcherio Martinoli. Comparative study of market-based and threshold-based task allocation. In *Distributed autonomous robotic systems 7*, pages 91–101. Springer, 2006.

[98] Anshul Kanakia, John Klingner, and Nikolaus Correll. A response threshold sigmoid function model for swarm robot collaboration. In *Distributed Autonomous Robotic Systems*, pages 193–206. Springer, 2016.

[99] Peter Karlson and Martin Lüscher. 'pheromones': a new term for a class of biologically active substances. 1959.

[100] Istvan Karsai and Gabor Balazsi. Organization of work via a natural substance: regulation of nest construction in social wasps. *Journal of Theoretical Biology*, 218(4):549–565, 2002.

[101] Istvan Karsai and Andrew Runciman. The 'common stomach'as information source for the regulation of construction behaviour of the swarm. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):13–24, 2012.

[102] Istvan Karsai and Thomas Schmickl. Regulation of task partitioning by a "common stomach": a model of nest construction in social wasps. *Behavioral Ecology*, 22(4):819–830, 2011.

[103] István Karsai and John W Wenzel. Productivity, individual-level and colony-level flexibility, and organization of work as consequences of colony size. *Proceedings of the National Academy of Sciences*, 95(15):8665–8669, 1998.

[104] Istvan Karsai and John W Wenzel. Organization and regulation of nest construction behavior in metapolybia wasps. *Journal of Insect Behavior*, 13(1):111–140, 2000.

[105] Akshay Kashyap, Tamer Başar, and Ramakrishnan Srikant. Quantized consensus. *Automatica*, 43(7):1192–1203, 2007.

[106] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

[107] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[108] Ehsan Khosrowshahi-Asl, Majid Noorhosseini, and Atieh Saberi Pirouz. A dynamic ant colony based routing algorithm for mobile ad-hoc networks. *Journal of Information Science and Engineering*, 27(5):1581–1596, 2011.

[109] Jungyun Kim, Seong Youb Chung, and Hyun Joong Yoon. Multi-agent-based scheduling methods for hybrid cellular production lines in semiconductor industry. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 228(12):1701–1712, 2014.

[110] Min-Hyuk Kim, Hyeoncheol Baik, and Seokcheon Lee. Response threshold model based uav search planning and task allocation. *Journal of Intelligent & Robotic Systems*, 75(3-4):625–640, 2014.

[111] Oran Kittithreerapronchai and Carl Anderson. Do ants paint trucks better than chickens? markets versus response thresholds for distributed dynamic scheduling. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 2, pages 1431–1439. IEEE, 2003.

[112] Jens Krause and Graeme D Ruxton. *Living in groups*. Oxford University Press, 2002.

[113] Michael JB Krieger and Jean-Bernard Billeter. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30(1):65–84, 2000.

[114] Michael JB Krieger, Jean-Bernard Billeter, and Laurent Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995, 2000.

[115] Thomas H Labella, Marco Dorigo, and Jean-Louis Deneubourg. Division of labor in a group of robots inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(1):4–25, 2006.

[116] Michail G Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton J Kleywegt, Sven Koenig, Craig A Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, volume 5, page 343C350. Rome, Italy, 2005.

[117] James S Langer. Instabilities and pattern formation in crystal growth. *Reviews of Modern Physics*, 52(1):1, 1980.

[118] Wonki Lee and DaeEun Kim. Adaptive division of labor in multi-robot system using visual information. In *International Symposium on Distributed Autonomous Robotic Systems*, pages 234–235. Springer, 2014.

[119] Wonki Lee and DaeEun Kim. Adaptive division of labor in multi-robot system with minimum task switching. In *The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 750–756. MIT Press, 2014.

[120] Wonki Lee and DaeEun Kim. Dynamic scheduling for job shop problem based on wasps colony algorithm. In *International Symposium on Distributed Autonomous Robotic Systems*, pages 194–195. Springer, 2014.

[121] Wonki Lee and DaeEun Kim. Task partitioning based on response threshold model in robot harvesting task. In *The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 759–760. MIT Press, 2014.

[122] Wonki Lee and DaeEun Kim. Desynchronization based response threshold model for task allocation in multi-agent systems. In *The First International Symposium on Swarm Behaviors and Bio-Inspired Robotics*, pages 153–154, 2015.

[123] Wonki Lee and DaeEun Kim. Desynchronization-based task allocation in multi-agent systems. In *International Technical Conference on Circuits Systems, Computers and Communications*, pages 508–509, 2015.

[124] Wonki Lee and DaeEun Kim. Dynamic task allocation using a pheromone-based approach in factory domain applications. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2015 IEEE/WIC/ACM International Conference on*, volume 2, pages 174–177. IEEE, 2015.

[125] Wonki Lee and DaeEun Kim. Local interaction of agents for division of labor in multi-agent systems. In *International Conference on Simulation of Adaptive Behavior*, pages 46–54. Springer, 2016.

[126] Wonki Lee and DaeEun Kim. Adaptive stochastic strategies to regulate division of labor in multi-agent systems (submitted). *Swarm Evolutionary Computation*, 2017.

[127] Wonki Lee and DaeEun Kim. Autonomous shepherding behaviors of multiple target steering robots. *Sensors*, 17(12):2729, 2017.

[128] Wonki Lee and DaeEun Kim. Dynamic scheduling using a pheromone-based approcah in multi-agent systems (submitted). *Applied Soft Computing*, 2017.

[129] Wonki Lee and DaeEun Kim. Handling interference effects on foraging with bucket brigades. *Bioinspiration & Biomimetics*, 12(6):066001, 2017.

[130] Wonki Lee and DaeEun Kim. History-based response threshold model for division of labor in multi-agent systems. *Sensors*, 17(6):1232, 2017.

[131] Wonki Lee and DaeEun Kim. Personal biometric identification using multi-cycle ecg waveform pattern (submitted). *Sensors*, 2017.

[132] Wonki Lee and DaeEun Kim. Spike response threshold model for task allocation in multi-agent systems. In *Robot and Human Interactive Communication (RO-MAN), 2017 26th IEEE International Symposium on*, pages 1165–1168. IEEE, 2017.

[133] Wonki Lee and DaeEun Kim. Task allocation in multi-stage sequential task using swarm robotics (preparing). *IEEE access*, 2017.

[134] Wonki Lee and DaeEun Kim. Task partitioning to sequential interdependent tasks in swarm of robots (submitted). *Swarm Intelligence*, 2017.

[135] K. Lerman. A model of adaptation in collaborative multi-agent systems. *Adaptive Behavior*, 12(3-4):187–197, 2004.

[136] Kristina Lerman and Aram Galstyan. Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13(2):127–141, 2002.

[137] Kristina Lerman and Aram Galstyan. Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13(2):127–141, 2002.

[138] Kristina Lerman, Chris Jones, Aram Galstyan, and Maja J Matarić. Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, 25(3):225–241, 2006.

[139] Reinhard H Leuthold, O Bruinsma, and A van Huis. Optical and pheromonal orientation and memory for homing distance in the harvester termite *Hodotermes mossambicus (Hagen). Behavioral ecology and sociobiology*, 1(2):127–139, 1976.

[140] Stuart S Levine, Ian FG King, and Robert E Kingston. Division of labor in polycomb group repression. *Trends in biochemical sciences*, 29(9):478–485, 2004.

[141] Paweł Lichocki, Danesh Tarapore, Laurent Keller, and Dario Floreano. Neural networks as mechanisms to regulate division of labor. *The American Naturalist*, 179(3):391–400, 2012.

[142] Wenguo Liu, Alan FT Winfield, Jin Sa, Jie Chen, and Lihua Dou. Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive behavior*, 15(3):289–305, 2007.

[143] Yanfei Liu and Kevin M Passino. Stable social foraging swarms in a noisy environment. *IEEE Transactions on automatic control*, 49(1):30–44, 2004.

[144] Steven KC Lo. A collaborative multi-agent message transmission mechanism in intelligent transportation system–a smart freeway example. *Information Sciences*, 184(1):246–265, 2012.

[145] F Lopez, C Agbogba, and I Ndiaye. Prey chain transfer behaviour in the african stink ant *Pachycondyla tarsata Fabr. Insectes Sociaux*, 47(4):337–342, 2000.

[146] Siriluck Lorpunmanee, Mohd Noor Sap, Abdul Hanan Abdullah, and Chai Chompoo-inwai. An ant colony optimization for dynamic job scheduling in grid environment. *International Journal of Computer and Information Science and Engineering*, 1(4):207–214, 2007.

[147] Elise L Mansfield, Frini Karayanidis, Sharna Jamadar, Andrew Heathcote, and Birte U Forstmann. Adjustments of response threshold during task switching: a model-based functional magnetic resonance imaging study. *The Journal of neuroscience*, 31(41):14688–14692, 2011.

[148] Magdalene Marinaki and Yannis Marinakis. A glowworm swarm optimization algorithm for the vehicle routing problem with stochastic demands. *Expert Systems with Applications*, 46:145–163, 2016.

[149] Alcherio Martinoli, Kjerstin Easton, and William Agassounon. Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *The International Journal of Robotics Research*, 23(4-5):415–436, 2004.

[150] Alcherio Martinoli, Auke Jan Ijspeert, and Francesco Mondada. Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, 29(1):51–63, 1999.

[151] Maja J Matarić. Learning social behavior. *Robotics and Autonomous Systems*, 20(2):191–204, 1997.

[152] Maja J Mataric. Using communication to reduce locality in distributed multiagent learning. *Journal of experimental & theoretical artificial intelligence*, 10(3):357–369, 1998.

[153] T William Mather and M Ani Hsieh. Macroscopic modeling of stochastic deployment policies with time delays for robot ensembles. *The International Journal of Robotics Research*, page 0278364911401442, 2011.

[154] T William Mather and M Ani Hsieh. Synthesis and analysis of distributed ensemble control strategies for allocation to multiple tasks. *Robotica*, 32(2):177–192, 2014.

[155] Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. A survey of swarm intelligence for dynamic optimization: algorithms and applications. *Swarm and Evolutionary Computation*, 2017.

[156] Hans Meinhardt. Models of biological pattern formation. 1982.

[157] Papoutsidakis Michail, Piromalis Dimitrios, Neri Filippo, and Camilleri Michel. Intelligent algorithms based on data processing for modular robotic vehicles control. *WSEAS Transactions on systems*, 13:242–251, 2014.

[158] Marvin L Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.

[159] Francesco Mondada, Edoardo Franzi, and Paolo Ienne. Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Proceedings of the Third International Symposium on Experimental Robotics*, volume 200, pages 501–513. Springer, 1994.

[160] László Monostori, József Váncza, and Soundar RT Kumara. Agent-based systems for manufacturing. *CIRP Annals-Manufacturing Technology*, 55(2):697–720, 2006.

[161] Sérgio Monteiro and Estela Bicho. Attractor dynamics approach to formation control: theory and application. *Autonomous Robots*, 29(3):331–355, 2010.

[162] RE Morley and C Schelberg. An analysis of a plant-specific dynamic scheduler. In *NSF workshop on dynamic scheduling*, pages 115–122, 1993.

[163] Richard Morley. Painting trucks at general motors: The effectiveness of a complexity-based approach. *Embracing Complexity: Exploring the application of complex adaptive systems to business*, pages 53–58, 1996.

[164] Mehdi Moussaïd, Dirk Helbing, and Guy Theraulaz. How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences*, 108(17):6884–6888, 2011.

[165] Peter B Cech Moyle and J Joseph. *Fishes: an introduction to ichthyology*. Number 597 MOY. 2004.

[166] Satyasai Jagannath Nanda and Ganapati Panda. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary computation*, 16:1–18, 2014.

[167] Ian Newton. *The migration ecology of birds*. Academic press, 2010.

[168] Stefano Nolfi and Dario Floreano. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000.

[169] Ernesto Nunes and Maria L Gini. Multi-robot auctions for allocation of tasks with temporal constraints. In *AAAI*, pages 2110–2116, 2015.

[170] Rehan O'Grady, Anders Lyhne Christensen, and Marco Dorigo. Swarmorph: multirobot morphogenesis using directional self-assembly. *IEEE Transactions on Robotics*, 25(3):738–743, 2009.

[171] Esben H Østergaard, Gaurav S Sukhatme, and Maja J Mataric. Emergent bucket brigading: a simple mechanism for improving performance in multi-robot constrained-space foraging tasks. In *Proceedings of the fifth international conference on Autonomous agents*, pages 29–30. ACM, 2001.

[172] Rehan O'Grady, Carlo Pinciroli, Anders Lyhne Christensen, and Marco Dorigo. Supervised group size regulation in a heterogeneous robotic swarm. *Proceedings of ROBOTICA*, pages 113–119, 2009.

[173] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

[174] Lynne E Parker. Alliance: An architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on*, 14(2):220–240, 1998.

[175] Marie-Hélène Pillot, Jacques Gautrais, Patrick Arrufat, Iain D Couzin, Richard Bon, and Jean-Louis Deneubourg. Scalable rules for coherent group motion in a gregarious vertebrate. *PloS one*, 6(1):e14487, 2011.

[176] Carlo Pinciroli. The swarmanoid simulator. *Bruxelles: UniversitéLibre de Bruxelles*, 2007.

[177] Giovanni Pini, Arne Brutschy, Mauro Birattari, and Marco Dorigo. Interference reduction through task partitioning in a robotic swarm. In *Proceedings of the sixth International Conference on Informatics in Control, Automation and Robotics–ICINCO*, pages 52–59, 2009.

[178] Giovanni Pini, Arne Brutschy, Marco Frison, Andrea Roli, Marco Dorigo, and Mauro Birattari. Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, 5(3-4):283–304, 2011.

[179] Giovanni Pini, Arne Brutschy, Carlo Pinciroli, Marco Dorigo, and Mauro Birattari. Autonomous task partitioning in robot foraging: an approach based on cost estimation. *Adaptive behavior*, 21(2):118–136, 2013.

[180] Giovanni Pini, Arne Brutschy, Carlo Pinciroli, Marco Dorigo, and Mauro Birattari. Autonomous task partitioning in robot foraging: an approach based on cost estimation. *Adaptive behavior*, 21(2):118–136, 2013.

[181] Giovanni Pini, Arne Brutschy, Alexander Scheidler, Marco Dorigo, and Mauro Birattari. Task partitioning in a robot swarm: Object retrieval as a sequence of subtasks with direct object transfer. *Artificial Life*, 20(3):291–317, 2014.

[182] G Polverino, N Abaid, V Kopman, S Macrì, and M Porfiri. Zebrafish response to robotic fish: preference experiments on isolated individuals and small shoals. *Bioinspiration & biomimetics*, 7(3):036019, 2012.

[183] Francis LW Ratnieks and C Anderson. Task partitioning in insect societies. *Insectes sociaux*, 46(2):95–108, 1999.

[184] John H Reif and Hongyan Wang. Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, 27(3):171–194, 1999.

[185] J.L. Reyes and F. Fernández-Haegar. Sequential co-operative load transport in the seed-harvesting ant *Messor barbarus*. *Insectes Sociaux*, 46:119–125, 1999.

[186] Jeanne Robert L. The evolution of the organization of work in social insects. *Monitore Zoologico Italiano-Italian Journal of Zoology*, 20(2):119–133, 1986.

[187] ANJA Robinson. Dietary supplements for reproductive conditioning of crassostrea gigas kumamoto (thunberg). i. effects on gonadal development, quality of ova and larvae through metamorphosis. *Journal of Shellfish Research*, 11:437–437, 1992.

[188] J Röschard and F Roces. Cutters, carriers and transport chains: distance-dependent foraging strategies in the grass-cutting ant atta vollenweideri. *Insectes Sociaux*, 50(3):237–244, 2003.

[189] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer, 2004.

[190] Mohsen Amini Salehi, Hossein Deldari, and Bahare Mokarram Dorri. Balancing load in a computational grid applying adaptive, intelligent colonies of ants. *Informatica*, 33(2), 2009.

[191] Thomas Schmickl and Heiko Hamann. Beeclust: A swarm algorithm derived from honeybees. *Bio-inspired Computing and Communication Networks*, pages 95–137, 2011.

[192] Thomas Schmickl and Istvan Karsai. Sting, carry and stock: How corpse availability can regulate de-centralized task allocation in a ponerine ant colony. *PloS one*, 9(12):e114611, 2014.

[193] Thomas Schmickl and Istvan Karsai. How regulation based on a common stomach leads to economic optimization of honeybee foraging. *Journal of theoretical biology*, 389:274–286, 2016.

[194] Thomas Schmickl, Ronald Thenius, and Karl Crailsheim. Swarm-intelligent foraging in honeybees: benefits and costs of task-partitioning and environmental fluctuations. *Neural Computing and Applications*, 21(2):251–268, 2012.

[195] Miguel Schneider-Fontán and Maja J Mataric. A study of territoriality: The role of critical mass in adaptive task division. In *From Animals to Animals 4: Proceedings of the Fourth International Conference of Simulation of Adaptive Behavior*, pages 553–561. MIT Press, 1996.

[196] Thomas D Seeley. Division of labor between scouts and recruits in honeybee foraging. *Behavioral Ecology and Sociobiology*, 12(3):253–259, 1983.

[197] Thomas D Seeley. The wisdom of the hive: the social physiology ofhoney bee colonies. *Cambridge, MA: Harvard UniversityPress*, 1995.

[198] Cheng Shao and Dimitrios Hristu-Varsakelis. Cooperative optimal control: broadening the reach of bio-inspiration. *Bioinspiration & biomimetics*, 1(1):1, 2006.

[199] Dylan A Shell and Maja J Mataric. On foraging strategies for large-scale multi-robot systems. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2717–2723. IEEE, 2006.

[200] Weiming Shen and Douglas H Norrie. Agent-based systems for intelligent manufacturing: a state-of-the-art survey. *Knowledge and information systems*, 1(2):129–156, 1999.

[201] Xiao-Ning Shen and Xin Yao. Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Information Sciences*, 298:198–224, 2015.

[202] Stephen L Smith and Francesco Bullo. Monotonic target assignment for robotic networks. *Automatic Control, IEEE Transactions on*, 54(9):2042–2057, 2009.

[203] Andrei P Sommer, Dan Zhu, Tim Scharnweber, and Hans-Joerg Fecht. On the social behaviour of cells. *Journal of Bionic Engineering*, 7(1):1–5, 2010.

[204] William M Spears, Diana F Spears, Jerry C Hamann, and Rodney Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2):137–162, 2004.

[205] Valerio Sperati, Vito Trianni, and Stefano Nolfi. Self-organised path formation in a swarm of robots. *Swarm Intelligence*, 5(2):97–119, 2011.

[206] Anton Stabentheiner, Helmut Kovac, and Robert Brodschneider. Honeybee colony thermoregulation–regulatory mechanisms and contribution of individuals in dependence on age, location and thermal stress. *PLoS One*, 5(1):e8967, 2010.

[207] Jan-Philipp Steghofer, Jörg Denzinger, Holger Kasinger, and Bernhard Bauer. Improving the efficiency of self-organizing emergent systems by an advisor. In *Engineering of Autonomic and Autonomous Systems (EASe), 2010 Seventh IEEE International Conference and Workshops on*, pages 63–72. IEEE, 2010.

[208] Liselotte Sundstrom. Sex allocation and colony maintenance in monogyne and polygyne colonies of formica truncorum (hymenoptera: Formicidae): the impact of kinship and mating structure. *The American Naturalist*, 146(2):182–201, 1995.

[209] V Suresh and Dipak Chaudhuri. Dynamic scheduling—a survey of research. *International Journal of Production Economics*, 32(1):53–63, 1993.

[210] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[211] Ronald Thenius, Thomas Schmickl, and Karl Crailsheim. Optimisation of a honeybee-colony's energetics via social learning based on queuing delays. *Connection Science*, 20(2-3):193–210, 2008.

[212] Guy Theraulaz, Eric Bonabeau, and JN Denuebourg. Response threshold reinforcements and division of labour in insect societies. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1393):327–332, 1998.

[213] Daxin Tian, Junjie Hu, Zhengguo Sheng, Yunpeng Wang, Jianming Ma, and Jian Wang. Swarm intelligence algorithm inspired by route choice behavior. *Journal of Bionic Engineering*, 13(4):669–678, 2016.

[214] Chris Tofts. Algorithms for task allocation in ants.(a study of temporal polyethism: theory). *Bulletin of Mathematical Biology*, 55(5):891–918, 1993.

[215] John Toner and Yuhai Tu. Flocks, herds, and schools: A quantitative theory of flocking. *Physical review E*, 58(4):4828, 1998.

[216] Sahar Trigui, Anis Koubaa, Omar Cheikhrouhou, Habib Youssef, Hachemi Bennaceur, Mohamed-Foued Sriti, and Yasir Javed. A distributed market-based algorithm for the multi-robot assignment problem. *Procedia Computer Science*, 32:1108–1114, 2014.

[217] Richard T Vaughan, Brian P Gerkey, and Andrew Howard. On device abstractions for portable, reusable robot code. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2421–2427. IEEE, 2003.

[218] Kirk P Visscher and Reuven Dukas. Honey bees recognize development of nestmates' ovaries. *Animal Behaviour*, 49(2):542–544, 1995.

[219] Markus Waibel, Dario Floreano, Stéphane Magnenat, and Laurent Keller. Division of labour and colony efficiency in social insects: effects of interactions between genetic architecture, colony kin structure and rate of perturbations. *Proceedings of the Royal Society of London B: Biological Sciences*, 273(1595):1815–1823, 2006.

[220] Markus Waibel, Laurent Keller, and Dario Floreano. Genetic team composition and level of selection in the evolution of cooperation. *IEEE Transactions on Evolutionary Computation*, 13(3):648–660, 2009.

[221] Joanne H Walker and Myra S Wilson. Task allocation for robots using inspiration from hormones. *Adaptive Behavior*, 19(3):208–224, 2011.

[222] Akira Watanabe and Kimihisa Takeda. The change of discharge frequency by ac stimulus in a weak electric fish. *Journal of Experimental Biology*, 40(1):57–66, 1963.

[223] Wilson. *The insect societies*. Cambridge, Massachusetts, USA, Harvard University Press, 1971.

[224] Edward O Wilson. The relation between caste ratios and division of labor in the ant genus pheidole (hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, 16(1):89–98, 1984.

[225] Mark L Winston. *The biology of the honey bee*. Harvard University Press, 1991.

[226] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399, 2013.

[227] Jianyi Yang, Ruifeng Ding, Yuan Zhang, Maoqin Cong, Fei Wang, and Guoan Tang. An improved ant colony optimization (i-aco) method for the quasi travelling salesman problem (quasi-tsp). *International Journal of Geographical Information Science*, (ahead-of-print):1–18, 2015.

[228] Jin-hui Yang, Liang Sun, Heow Pueh Lee, Yun Qian, and Yan-chun Liang. Clonal selection based memetic algorithm for job shop scheduling problems. *Journal of Bionic Engineering*, 5(2):111–119, 2008.

[229] Yongming Yang, Changjiu Zhou, and Yantao Tian. Swarm robots task allocation based on response threshold model. In *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*, pages 171–176. IEEE, 2009.

[230] Payam Zahadat and Thomas Schmickl. Division of labor in a swarm of autonomous underwater robots by improved partitioning social inhibition. *Adaptive Behavior*, 24(2):87–101, 2016.

[231] Ouarda Zedadra, Nicolas Jouandeau, Hamid Seridi, and Giancarlo Fortino. Multi-agent foraging: state-of-the-art and research challenges. *Complex Adaptive Systems Modeling*, 5(1):3, 2017.

[232] Dandan Zhang, Guangming Xie, Junzhi Yu, and Long Wang. Adaptive task assignment for multiple mobile robots via swarm intelligence approach. *Robotics and Autonomous Systems*, 55(7):572–588, 2007.

[233] Rui Zhang and Cheng Wu. Bottleneck machine identification method based on constraint transformation for job shop scheduling with genetic algorithm. *Information Sciences*, 188:236–252, 2012.

[234] Yingfeng Zhang, George Q Huang, Shudong Sun, and Teng Yang. Multi-agent based real-time production scheduling method for radio frequency identification enabled ubiquitous shopfloor environment. *Computers & Industrial Engineering*, 76:89–97, 2014.

국 문 요 약

군집 지능을 이용한 다중 개체 시스템에서의 적응적 임무 할당

자연계에서 군집 현상은 오래 전부터 관측되어 왔으며, 다수의 개체들이 협동과 분업화 과정으로 생존에 더욱 유리하도록 진화되어 왔다는 것은 이미 널리 알려진 사실이다. 개체 간의 많은 상호 작용은 복잡한 네트워크와 같지만, 단일 개체가 수행하기 어렵거나 불가능한 임무를 제한된 능력을 가진 개체들이 몇 가지 행동 규칙을 사용하여 조직적으로 협동하여 수행합니다. 각 개체가 전문화된 임무를 동시에 여러 장소에서 개별적으로 수행함으로써, 단일개체에 의존한 시스템보다 작업의 효율성과 유연성을 확보할 수 있으며, 한 개체의 고장이나 오류가 전체 임무 수행에 큰 영향을 미치지 않아 안정적인 목표 달성을 기대할 수 있다.

본 연구에서는 군집지능 기반 응답 임계 모델을 이용하여 주위 환경의 제한적인 정보를 이용하여 개별 개체들이 주어진 임무의 요구량에 비례하여 각 임무에 능동적으로 적응하는 모델을 제안한다. 각 개체는 모든 임무에 대해 응답 임계값을 가지고 있으며, 각

임무의 요구 정도와 임계값을 고려하여 임무 선택 함수의 결과가 높은 임무를 수행하게 된다. 임계값을 잘 조절하는 것이 중요하며, 임무에 대한 정보뿐만 아니라 이를 수행하는 주변 개체의 임무 정보를 활용하여 각 개체는 특정 임무에 대한 전문화 경향을 가지게 된다. 이러한 전문화 경향을 통해 임무 전환을 최소화하면서 원하는 작업 분업 비율을 달성하며 수학적인 모델링을 통해 수렴성도 증명하였다.

다양한 실험 결과를 통해 집단 구성원의 비율 또는 작업 요구의 변화와 같은 환경 변화에서의 적응 능력을 입증하였으며, 임무 변환에 부가적인 시간이나 에너지를 필요로 하는 경우, 응답 임계 모델을 활용하면 성능을 높일 수 있을 것으로 기대된다. 또한, 순차적 작업 할당에서의 다양한 실험을 통해 공통 구역에 병목 현상이있는 경우 작업 분할이 교통 체증을 줄이고, 전체 시스템의 인 성능을 향상시키는 데 효과적인 전략이 될 수 있음을 보여준다. 본 연구의 결과는 많은 사회 곤충들이 생존의 확률을 높이기 위해 다양한 임무 할당 전략을 사용한다는 가설을 뒷받침한다.

핵심되는 말 : 임무 할당, 다중개체 시스템, 군집 지능, 응답 임계 모델, 수렴성, 전문성