DISSERTATION

Collective behavior generation and analysis for an evolutionary swarm robotics system

（進化的スワームロボティクスシステムにおける

群れ行動の生成と解析）

余　天

Graduate School of Engineering
Hiroshima University
March 2016

Acknowledgements

The research included in this dissertation could not have been performed if not for the assistance, patience, and support of many individuals. I am deeply indebted to my supervisor, Prof. Ohkura, K. for his fundamental role in my PhD candidate life, for the support of my doctoral course study and research, for his expertise, motivation, and immense knowledge. Besides my advisor, I also would like to thank the rest of my thesis committee: Prof. Yamada, K., associate Prof. Iwamoto, T. and associate Prof. Matsumura, Y., for their revising thesis, and insightful comments.

Thanks to the Yahata Memorial Ikuei Scholarship for offering me the scholarship during my doctoral course.

Thanks to assistant Prof. Yasuda, for advising me both in research and life. I want to thank associate Pro. Goka, M., for the good discussion that we had in the second year of my master course. Thanks to the current and former members of Manufacturing Systems A Laboratory at the Graduate School of Engineering: Hai Shan, Kadota, adachi, wei, murakawa, morikawa, abo, sun, nakashima, nakatani, hagimori, okamura, kataoka, suenaga, hiraga, takashi.

Thanks to the Global Career Design Center (Young Researchers' Training Division) for advising me about my career design. I would also like to thank all of my friends who supported me in writing, and encouraged me to strive towards my goal.

Special thanks to my love Shevin Zhang, for her moral support, encouragement, quiet patience and unwavering love were undeniably the bedrock upon which the past five years of my life have been built.

Finally I would like to extend my deepest gratitude to my parents Yu Jie and He Mingyin, without whose love, support and understanding I could never have completed this doctoral degree. Words cannot express how grateful I am to my mother and father for all of the sacrifices that you have made on my behalf. Your prayer for me was what sustained me thus far.

Tian Yu
Hiroshima, Japan
March 2016

Contents

Chapter 1

Introduction

1.1 Background

Swarm robotics (SR) (Şahin, 2005) is a novel approach inspired by the observation of social insects, such as ants and wasps. These examples of social insects show that simple individuals can successfully accomplish difficult tasks when they coordinate as a group. Recently, swarm robotics gathered much attention in the research community. By drawing inspiration from social insects and other self-organizing systems, it focuses on large robot groups featuring distributed control, adaptation, high robustness and flexibility. Various reasons lay behind this interest in similar multi-robot systems.

Above all, inspiration comes from the observation of social activities, which are based on concepts like division of labour, cooperation and communication. If societies are organized in such a way in order to be more efficient, then also robotic groups could benefit from similar paradigms.

Constructing tools from a collection of individuals is not a novel endeavor for man. A chain is a collection of links, a rake is a collection of tines, and a broom is a collection of bristles. Sweeping the sidewalk would certainly be difficult with a single or even a few bristles. Thus there must exist tasks that are easier to accomplish using a collection of robots, rather than just one.

A multi-robot approach can have many advantages over a single-robot system. First, a monolithic robot that could accomplish various tasks in varying environmental conditions

(a) A group of ants build a bridge between two leaves. (www.independent.co.uk)

(b) A group of red harvester ants scurrying to carry a large worm back to their nest. (en.wikipedia.org)

Figure 1.1: Cooperative collective behavior of ants

is difficult to design. Moreover, the single-robot approach suffers from the problem that even small failures of the robotic unit may prevent the accomplishment of the whole task. On the contrary, a multi-robot approach can benefit from the parallelism of operation to be more efficient, from the versatility of its multiple, possibly heterogeneous, units and from the inherent redundancy given by the usage of multiple agents.

Swarm robotics pushes the cooperative approach to its extreme. It represents a theoretical and methodological approach to the design of ”intelligent” multi-robot systems inspired by the efficiency and robustness observed in social insects in performing collective tasks [8]. Collective motion in fish, birds and mammals, as well as collective decisions, synchronization and social differentiation are examples of collective responses observed in natural swarms. In all these examples, the individual behavior is relatively simple, but the global system behavior presents complex features that result from the multiple interactions of the system components. Similarly, in a swarm robotics system, the complexity of the group behavior should not reside in the individual controller, but in the interactions among the individuals. Thus, the main challenge in designing a swarm robotics system is represented by the need to identify suitable interaction rules among the individual robots. In other words, the challenge is designing the individual control rules that can lead to the desired global behavior.

The expression *swarm intelligence* was first conceived by Beni to denote a class of cellular robotic systems in 1980s. These works used many simple agents occupy one-or two-dimensional environment to generate patterns and self-organize there nearest-neighbor interactions. At that time, the definition *swarm intelligence* only marginally covers works on cellular robotic systems, which does not take the inspiration from social insect behavior. Recently, the expression "swarm intelligence" moved on to cover a wide range of researches from optimization to social insect studies, losing its robotics context in the meantime. Nowadays, the term SR has started to be used as the application of swarm intelligence to multi-robot systems. This concern was first explicitly started by Sahin in 2005.

As previously mentioned before, an SRS must have three functional properties at the system level that are observed in natural swarms:

Robustness is the ability to operate despite disturbances resulting from the malfunctioning of its individuals. For instance, lost individuals can be immediately replaced by others, so that the operation will continuing smoothly. This is seen as the key advantage of the SRS approach (Şahin and Winfield, 2008) (Şahin et al., 2008).

Flexibility is the ability of an SRS to generate modularized solutions to various tasks, meaning that an SRS must be able to adapt their behaviors to different environments.

Scalability is the ability of an SRS to operate with a wide range of group sizes and support a large number of individuals.

The concept of swarm engineering was introduced by Kazadi (Kazadi, 2000) in 2000 and the first formal introduction of swarm engineering was released by Winfield et al. in 2004 (Winfield et al., 2004). Researchers indicated that finding a predictable and controllable design methodology for swarms is the core research direction of swarm engineering (Bonabeau et al., 1999) (Beni, 2005). Today, although swarm engineering is still in a very early stage, core topics of swarm intelligence, the design, and analysis have already received attention from SR researchers. The notable swarm-bots project (Dorigo et al., 2005) was began in 2001 and terminated in 2005, and was followed by the swarmanoid projrect in 2006 (Dorigo et al., 2013). New approaches to the design and implementation of self-organizing and the self-assembling problem of autonomous robots were studied in the project (Soysal and Şahin, 2007).

1.2 Goal of the Thesis

As mentioned in the background section, there are two fundamental problems: the hot topic *design problem* and the undeveloped area *analysis problem*. These two problems consist of defining the appropriate individual rules that will lead to a certain global pattern and analyzing the generated collective behavior.

In this thesis, we focused on an automatic technique called evolutionary robotics approach for generating robust controllers for a robotic swarm, based on artificial evolution (Fogel et al., 1966; Holland,1975; Schwefel, 1981; Goldberg, 1989). As benchmark problems of SR, the cooperative transport and the cooperative food foraging problem are investigated. Specifically, we focused on the covariance matrix adaptation evolution strategy (CMA-ES) with an artificial neural network to develop an efficience approach, CMA-NeuroES, for an SRS to solve cooperative food foraging problems (Martinoli et al., 2004).

However, when an evolutionary approach meets a complex task like food foraging problem, it is typical that a simple ER strategy will face a situation that all individuals of the first generation are scored with the same null value; moreover, the selection process cannot operate as expected. This is called *bootstrap problem* which very often occurs to difficult tasks. To avoid this kind of failure, an incremental evolution approach with staged evolution and environmental complexification for a cooperative food foraging task is adopted.

In addition, an SRS can work dynamically as the individual robots are deployed respectively. This kind of decentralized control ensures that SRS has no common failure point. The failure of individual robots will hardly affect SRS performance. The resulting high-level robustness contrasts with the high engineering cost of fault tolerance in conventional robotics systems, and this is free as a basic property of an SRS. Consequently, we compare the best controllers evolved by CMA-NeuroES with two other evolutionary algorithms, fast evolution strategies (FESs) (Back et al., 1991) (B ack and Schwefel, 1993) and real-coded genetic algorithms (GAs), (Goldberg, 1989) through random breakdown tests in computer simulations. As an intrinsic property of SRS, we want to find out whether the loss of some individuals can be compensated for by others or not and also whether the destruction of a particular part of the swarm is will stop its operation or not.

Furthermore, an analysis method inspired from the field of complex networks is applied to a network associated with a SRS which is drawn by assuming that nodes are robots and links are informational connections with two nearest robots after we generating the collective behavior of a cooperative food foraging problem. This analysis method makes it possible for us to extract the subgroups in a robotic swarm. Applying this method with the concept of behavioral sequence inspired from ethology, we are possibly to observe task allocation of a cooperative food foraging problem.

1.3 Structure of the Thesis

The structure of the thesis are organized as follows.

The research field of swarm robotics system are introduced in chapter 2, which we talk about the definitions of SRS and introduce some typical works done by the pioneers researchers around the world from both the design methodology and the analysis methodology. This chapter also introduces the benchmarks of SRS and we will explain why researchers take the cooperative food foraging task as the most difficult benchmark problems in the field of SRS.

Chapter 3 investigates the concept of evolutionary robotics and artificial neural networks, which we applied ER approach to design the controller of SRS. In this chapter, we introduce both the concept and the advantage of ER. The related works of ER also mentioned here and at last we explain the reason why we focused ER approach to apply solve benchmarks of SRS.

Chapter 4 explains about the details of complex networks, as we proposed an analysis method from the field of complex networks, this chapter gives an overview of complex networks. Although most of the related works in this field are about social networks analysis, we find out that the SRS can be regard as a large network and it is possible to apply complex network approaches to analyze the behavior of SRS.

In chapter 5, we propose our design method called the neuroevolution based covariance matrix adaptation evolution strategy (CMA-ES) to a cooperative transport problem and two cooperative food foraging problem. As we mentioned above, the bootstrap problem occurs when our proposed method meets a specific cooperative food foraging problem. As a result, we apply incremental evolution to extended our ANN controller so that we get the most robust controller ever of our research group. The results are compared with other evolutionary algorithms like, genetic algorithm, evolutionary strategies, and diffierencial evolution.

The proposed analysis method was applied to the cooperative food foraging task in chapter 6. A clustering method inspired from the field of complex networks is applied to a network associated with a SRS helps us to find some subgroups of robots. Applying this method with the concept of behavioral sequence inspired from ethology, we are possibly to observe task allocation of a cooperative food foraging problem with a SRS.

Finally, conclusions for this thesis are given in chapter 7.

Chapter 2

Swarm Robotic Systems

In this chapter, the definition of swarm robotic systems is introduced at first. And then, the properties of swarm robotic system, the history of this research field and the benchmark of swarm robotic systems are described.

2.1 Introduction

Social insects stand as fascinating examples of how collectively intelligent system can be generated from a large number of individuals. Despite noise in the environment, errors in the processing information and in performing tasks, and the lack of global communication system, social insects can coordinate their actions to accomplish tasks that are beyond the capabilities of a single individual: termites build large and complex mounds, army ants organize impressive foraging raid, ants can collectively carry large prey.

In the last decade, swarm robotics gathered much attention in the research community. By drawing inspiration from social insects and other self-organizing systems, it focuses on large robot groups featuring distributed control, adaptation, high robustness and flexibility. Various reasons lay behind this interest in similar multi-robot systems. Above all, inspiration comes from the observation of social activities, which are based on concepts like division of labour, cooperation and communication. If societies are organized in such a way in order to be more efficient, then also robotic groups could benefit from similar paradigms.

A multi-robot approach can have many advantages over a single-robot system. First, a monolithic robot that could accomplish various tasks in varying environmental conditions is difficult to design. Moreover, the single-robot approach suffers from the problem that even small failures of the robotic unit may prevent the accomplishment of the whole task. On the contrary, a multi-robot approach can benefit from the parallelism of operation to be more efficient, from the versatility of its multiple, possibly heterogeneous, units and from the inherent redundancy given by the usage of multiple agents (Jones and Mataric, 2006) .

Swarm robotics pushes the cooperative approach to its extreme. It represents a theoretical and methodological approach to the design of *intelligent* multi-robot systems inspired by the efficiency and robustness observed in social insects in performing collective tasks. Collective motion in fish, birds and mammals, as well as collective decisions, synchronization and social differentiation are examples of collective responses observed in natural swarms (for some recent reviews, see (Camazine et al., 2001) (Franks et al., 2002) (Couzin, 2007) (Couzin and Krause, 2007) (Sumpter, 2006). In all these examples, the individual behavior is relatively simple, but the global system behavior presents complex features that result from the multiple interactions of the system components. Similarly, in a swarm robotics system, the complexity of the group behavior should not reside in the individual controller, but in the interactions among the individuals. Thus, the main challenge in designing a swarm robotics system is represented by the need to identify suitable interaction rules among the individual robots. In other words, the challenge is designing the individual control rules that can lead to the desired global behavior.

In the above perspective, self-organization is the mechanism that can explain how complex collective behaviors can be obtained in a swarm robotics system from simple individual rules. In this context, a complex collective behavior should be intended as some spatio-temporal organization in a system that is brought forth through the interactions among the system components. Not every collective behavior is self-organized, though [9]. The presence of a leader in the group, the presence of blueprints or recipes to be followed by the individual system components clashes with the concept of self-organization, at least at the level of description in which leader or blueprints are involved. Another condition in which a collective behavior cannot be considered self-organizing is when environmental cues or heterogeneities are exploited to support the group organization. For instance, animals that aggregate in a warm part of the environment following a temperature gradient do not self-organize. But animals that aggregate to stay warm, and therefore create and support a temperature gradient in the environment, do self-organize. In both cases, the observer may recognize the presence of some structure (the aggregate) that correlates with the presence of an environmental heterogeneity (the temperature gradient). However, the two examples are radically different from the organizational point of view. Similar natural examples can be easily given also for the presence of leader or blueprints, to show that not every collective behavior is self-organizing (Camazine et al., 2001). Both the leader or the blueprint can be recognized as the place where the behavioral complexity of the group is centralized.

In other words, the complexity of the group behavior does not result from the multiple interactions among the individual behaviors. Rather, the group behavior results from a fixed pattern of interactions among the system components that is either decided beforehand (in the case of a blueprint) or is centrally and/or continuously re-planned (in the case of a leader). In both cases, there is limited room for adaptiveness to unknown, unpredictable situations resulting from a highly dynamical environment, both physical and social.

2.2 The Origin of SRS

In 1980s, people are concentrated on the so called *Cellar Robots* (Fukuda and Nakagawa, 1989) that the designer just using very simple rules to generate the collective behavior of robots, especially for its insects-like assemble behaviors.

At that time, the word ”swarm” was first used to the public. After that, ”Swarm Optimization” like the Ant Colony Optimization (ACO)(Dorigo and Gambardella, 1997), Particle Swarm Optimization(PSO)(Kennedy and Eberhart, 1995) were released to the public in the research field of optimization. Biologists also tried to analysis the swarm of termites, ants and wasps to figure it out how these kinds of insects work together to build its nest or to forage for food. Researchers also tried to simulate the model of these swarm(Theraulaz and Bonabeau, 1995). After this type of research, Dr. Beni found that the Cellular Robots follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local, and to a certain degree random, interactions between such robots lead to the emergence of ”intelligent” global behavior, unknown to the individual agents, which was pointed out as a new research field: Swarm Intelligence (SI) (Beni and Wang, 1989). The expression was introduced by Gerardo Beni and Jing Wang in 1989, in the context of cellular robotic systems.

Recently, the research field Swarm Robotic Systems (SRS) is regard as an application of the Swarm Intelligence (SI) to the Multi-Robot Systems (MRS).

2.3 System-level properties

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 the agents and between the agents and the environment. The system-level operation of a swarm robotic system should exhibit three functional properties that are observed in natural swarms and remain as desirable properties of multi-robot systems.

- Robustness

  The swarm robotic system should be able to operate despite disturbances from the environment or the malfunction of its individuals. A number of factors can be observed in social insects behind the robustness of their operation. First, swarms are inherently redundant systems; the loss of an individual can be immediately compensated by another one. Second, coordination is decentralized and therefore the destruction of a particular part of the swarm is unlikely to stop its operation. Third, the individuals continue the swarm are relatively simple, making them less prone to failure. Fourth, sensing is distributed, hence the system is robust against the local perturbations in the environment (Fig. 2.1).

- Scalability

  The swarm should be able to operate under a wide range of group sizes and support large number of individuals without impacting performance considerably (Fig. 2.3). That is, the coordination mechanisms and strategies to be developed for swarm robotic systems should ensure that the operation of the swarm under varying swarm sizes.

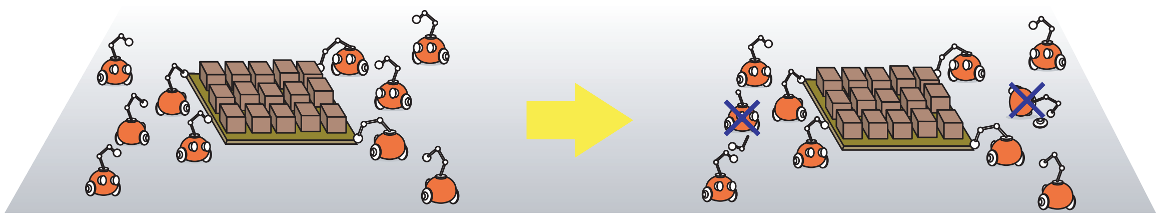


Figure 2.1: *Robustness* (It requires that the swarm robotic system should be able to continue to operate, although at a lower performance, despite failures in the individuals, or disturbances in the environment.)

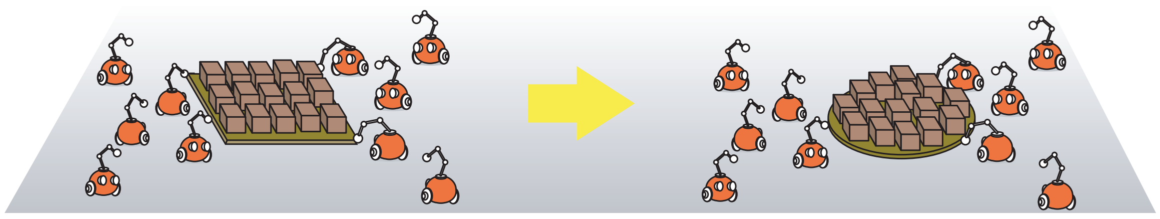


Figure 2.2: *Flexibility* (It requires the swarm robotic system to have the ability to generate modularized solutions to different tasks. Swarm robotic systems should also have the flexibility to offer solutions to the tasks at hand by utilizing different coordination strategies in response to the changes in the environment.)

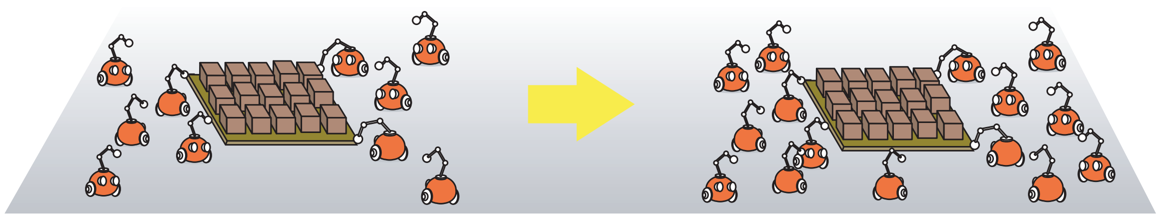


Figure 2.3: *Scalabilityrequires* (Swarm robotic system should be able to operate under a wide range of group sizes. That is, the coordination mechanisms that ensure the operation of the swarm should be relatively undisturbed by changes in the group sizes.)

- Flexibility

  The individuals of a swarm should be able to coordinate their behaviors to tackle tasks of different nature (Fig. 2.2). For instance, the individuals in an ant colony can collectively find the shortest path to a food source or carry a large prey through the utilization of different coordination strategies.

2.4 Definitions of SRS

We pointed out that the SRS is an application of MRS, however it is not enough to make people know what exactly SRS is. There are also some different branches of MRS, For example, collective robotics (Kube and Zhang, 1997), distributed robotics (Yim et al., 2001) and robot colonies (Arkin and Bekey, 1997). The Distinguishing characteristics of SRS are showed as the follows:

- Physical embodiment

  That means every single robot in the SRS should be able to physically interact with their environment. The main emphasis in swarm robotics is the interaction between the robots and its environment. Sensing, signaling and communications are the basic interaction here (Akyildiz et al., 2002). The robots get signals through the sensors and the physical interaction between robots or obstacles. Most of the time, a robot system can not be called as SRS without sensor network (Akyildiz et al., 2002), however, researchers see the metamorphic robot system (Chirikjian, 1994) (Yim et al., 2001) as a kind of SRS even the robots interact without centralized control.

- Large size

  SRS should be scalable for a wide range of swarm size. That means not only one or two robots, but more then ten robots to cooperate at the same swarm. Large numbers of robots lead to the properties of scalability. The SRS can work with all of the robots and can also work even some of the robots in the swarm have some problems. The researchers also use a large number of robots to archive complex tasks and to analyze the collective behavior of its swarm.

- Homogeneity

  That is, the individuals that makes up the swarm should be rather identical, at the level of interactions. Coordination strategies developed for heterogeneous multi-robot systems, which consist of individuals that differ in their interactions due to their physical embodiment or their behavioral control, fall outside of the swarm robotics approach (Şahin and Winfield, 2008).

- Simple robots

  The definition of simplicity does not refer to the hardware or software, but rather meant to emphasize the limitations in their individual capabilities relative to the task. A robot in the SRS cannot complete the task by itself without other robots' cooperation. The cooperation of a group of robots should be essential and the deployment of a group of robots should improve the performance or robustness of the handling of task.

- Local interaction

  The members in a SRS should have basic local interaction abilities. This constraint ensures that the coordination between the robots is distributed, and the coordination abilities are adaptive with the size of the swarm. Mechanisms that rely on global interaction capabilities is likely to be bounded by the bandwidth and the range of communication channel and may develop unscalable coordination mechanisms.

2.5 Coordination Mechanisms

Researchers of biology show that there are a number of coordination mechanisms in natural systems, in which two main coordination mechanisms are known as: self-organization and stigmergy.

Self-organization is common in natural systems. It is defined as GA process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the systemh (Camazine et al., 2001). Studies of self-organization in natural systems show that an interplay of positive and negative feedback of local interactions among the individuals is essential (Bonabeau et al., 1999). In these systems, positive feedback is typically generated through autocatalytic behaviors; that is the change occurs in the swarm-environment system by the execution of a behavior increases the triggering of the very same behavior. In addition to these mechanisms, self-organization also depends on the existence of randomness within the system and multiple interactions.

Studies of self-organization in natural systems often develop models that are built with simplified interactions in the environment and abstract behavioral mechanisms in individuals. The self-organization models of social insects and animals have already been used as inspiration sources since, in a sense, swarm robotics can be considered as the engineering and utilization of self-organization in physically embodied swarms.

Stigmergy was first proposed by Grasse (Grasse, 1959) to explain the coordination mechanisms behind the building of nests in termites. Stigmergic communication is quite common between social insects. Grasse showed that a particular configuration of a termite colony's environment can triggered a termite to modify its environment (drop a mud in a particular place for building or maintaining the nest). The modification in turn stimulates the original or other termites in colony to further transform its environment. Grasse made a general definition of stigmergy as: the stimulation of the workers by the very performances they have achieved. The stigmergy process has been observed in termites, ants, bees, and wasps in a wide range of activities. As the SRS provides a local, distributed scalable communication mechanism, it is the same to swarm robotics.

2.6 Research directions

2.6.1 Controller Design of SRS

The main problem of swarm robotic system can be stated as follows: How should one design individuals, both in terms of their physical embodiment as well as their behavioral control, such that a desired swarm-level behavior emerges from the interactions among the individuals as well as between the individuals and the environment. This goal, which can also be considered as the engineering of self-organization in multi-robot systems, is a challenging task that is difficult, if not impossible, to solve in general terms. The studies within this category can be grouped into two as: ad-hoc and principled approaches.

Although SRSs may be controlled by many different ways, they are categorized as either ad hoc or principled. In the ad hoc approach, almost all robot behavior is preprogramed by a human designer (Kube and Zhang, 1997) (Mataric, 1997) . Although the ad hoc approach is much more widely used in SR research, we believe that there may be a physical limit reachable for the human ability in programming all robot behavior when a task is highly complex or a swarm is large enough. In this approach, the behaviors used as inspiration sources are observed at a certain abstraction level that captures the essential parameters of these behavior needed to adapt to robots and yet reproduce similar swarm-level behaviors.

The principled approach introduces a set of principles for generating robot behavior. In principled approaches, instead of designing a specific swarm-level behavior, a general methodology through which desired swarm-level behaviors can be used to build necessary individual behaviors, is proposed or utilized. One of such approach is the use of artificial evolution. In most of these studies, simple feedforward or recurrent multi-layer perceptrons were used to encode the behaviors and that the evolved behaviors in the simulation environment were later successfully transferred to the physical robot system. (Dorigo et al., 2005). Relatively few such methods have been developed so far, but one widely used principled approach is Evolutionary Robotics (ER), in which robot controllers are represented by Evolving Artificial Neural Networks (EANNs) (Yao, 1999) (Floreano et al., 2008). Although this is a minor approach, with few reports available, we feel that it has great potential in freeing human beings from programming robot behavior.

ER performance clearly depends on the evolvability of EANNs. Classical EANNs are defined as having a fixed topological structure for all generations and a fixed number of evolving synaptic weights (Eiben et al., 2007) (Tuci et al., 2008). This approach matches the classical methodology of evolutionary algorithms in the sense that conventional artificial evolution works under the condition of a fixed-length genotype. Given that the topological structure depends on evolution performance, a drawback exists in that human programmers must provide an appropriate topological structure beforehand based on intuition and experience. Another approach, called Topology and Weight Evolving Artificial Neural

Networks (TWEANNs), has therefore been investigated. Among the several approaches proposed thus far are GNARL (Angeline et al., 1994), EPNet (Yao and Liu, 1997), ESP (Gomez and Miikkulainen, 1999), NEAT (Stanley and Miikkulainen, 2002), and EANT (Siebel and Sommer, 2007). Our research group has proposed a TWEANN method called the MBEANN (Mutation-Based Evolving Artificial Neural Network) (Ohkura et al., 2007). MBEANN has been confirmed to yield better solutions to the cooperative package-pushing problem with 10 robots (Ohkura et al., 2008), although the swarm size is considered the minimum for an SRS. An SRS is generally expected to consist of a large number of robots so that it can utilize its high redundancy to exhibit different strategies in different situations to complete a given task. It is thus assumed that many situations exist in which only some of the available robots are contributing directly. Accordingly, we frequently observe the other robots doing something else other than helping the hard-working robots.

2.6.2 Analysis methods of SRS

Robotic swarms are highly redundant systems and controlled by a way of emergent synthesis, to the best of our knowledge, no methods are available for grasping the collective behavior of robotic swarm in a macroscopic manner so far. However, analyzing collective behavior can be useful to develop more sophisticated swarm robotic systems. From the viewpoint that biological swarms perform task allocation suitable for situations, we have developed a technique for extracting autonomous specialization developed in a robotic swarm (Harvey et al., 1997). This technique can extract subgroups consisting of robots that play the same role for a given situation. A robotic swarm is regarded as a network according to the interaction between robots, and divided into subgroups by using a clustering method.

2.7 Benchmark Problems

During the last ten years, most of the SRS researches are still at the computer simulation level to complete benchmark tasks or examine the properties of SRS. This subsection would introduce some basic tasks of SRS.

- Aggregation

Aggregation is a common behavior observed in organisms ranging from bacteria to social insects and mammals. That means the grouping of individuals of a swarm into a cluster without using any environmental clues. In SRS, this is a precursor to other behaviors such as flocking and self-assembly. Aggregation behaviors were developed for myopic robots (Dorigo et al., 2005) (Soysal and Şahin, 2007), robots perceive only a small part of the environment, confined in a large arena using evolutionary approaches the same as controllers inspired from social insects.

- Dispersion

  This kind of tasks are regard as the opposite side of aggregation and it often of interest in surveillance scenarios. Researchers challenged to obtain uniform spreading of a swarm of robots into a space maximizing the area covered which remaining connected through some form of communication channel. (Payton et al., 2005) (Payton et al., 2001) (Christensen et al., 2007)

- Foraging

  This problem is inspired from the behavior of ants which search for food sources distributed around their nest and bring them back to the nest. In this problem, the difficulty is to find the optimum search strategies that let the robots have enough time or space to return food sources. different foraging strategies were explored and analyzed in terms of performance (Sugawara and Sano, 1997) (Krieger et al., 2000) (Liu et al., 2006) and the modelling of foragaing were developed. (Lerman et al., 2002) (Hamann and Worn, 2006)

- Self-assembly

  This behavior is also observed in ants, where they form chains through connecting to each other to build bridges or float-like structures to stay above water. The task of self-assembly means to create the structures through the formation of physical connections among a swarm of individual robots. Self-assembly has been studied in physical robots (Christensen et al., 2007) (Grady et al., 2007) such that a desired self-assembled structure is formed.

- Connected movement

  This problem can be described as follows: How can a swarm of mobile robots, physically connected to each other, coordinate their movement such that the group moves smoothly in an environment and avoids environmental obstacle, such as holes, in a coordinated way. This problem has been studied by Trianni et al. (Dorigo et al., 2005) within the Swarm-bots project. In this study, evolutionary approach was used to develop behaviors that can control a number of connected robots to avoid holes within the environment. The robots, which are physically connected to each other through their grippers, were able to sense the forces acting on their bodies through traction sensors and were able to feel holes underneath them.

- Cooperative transport

  Ants are known to transport large preys to their nest through coordinating their pushing and pulling actions. Such a coordination ability is obviously valuable for swarm robotic systems since it allows individuals to join forces generating a combined force large enough to pull a heavy object. This problem is partially related to the connected movement, with the difference that it includes a passive object that needs to be transported. In (Dorigo et al., 2004) a recurrent neural network controller is evolved to obtain solitary and group transport behaviors in a physics based simulator. The robots used the angular position of the goal (marked with a light source), the distance and angular position of prey and a connection sensor which indicates whether the robot is connected to other robots or not, were used to control the motors of the robot.

- Pattern formation

  This is a rather generic term for the problem of how a desired geometrical pattern can be obtained and maintained by a swarm of robots without any centralized coordination. The problem can be categorized into two; namely geometric and functional pattern formation. In geometric pattern formation problem, the challenge is to develop behaviors such that individuals of a swarm form a desired geometrical pattern, similar to the formation of crystals. In this task, the environment is assumed to be uniform and that the focus is on the use of inter-robot interactions to create such patterns. In functional pattern formation, the pattern to be formed is dictated by the environment. In natural swarms, encircling of a prey by a group of predators or the formation of pulling chains by weaver ants can be considered as examples of functional pattern formation, where the geometrical shape or size of the patterns formed are partially determined by the task at hand.

- Self-organized construction

  This problem can be formulated as follows: How can a number of passive objects, randomly distributed in an environment, can be clustered together by a swarm of robots. This problem, sometimes also being referred as aggregation has been one of first problems studied. Beckers et al. (Beckers et al., 1994) studied how a swarm of physical robots can cluster frisbees spread in an environment, and showed that despite the lack of communication and signaling among robots, frisbee clusters can be obtained.

2.8 Summary

In this chapter, we introduced the concept of swarm robotic systems by describe the origin of SRS and what properties should a swarm robotic system contain. We also provided a brief review of swarm robotics as a new approach to the control and coordination of multi-robot systems. We stated the inspirations behind this approach, the desirable properties, and the requirements to clarify the defining characteristics of this approach in relation to other existing studies.

Chapter 3

Evolutionary Robotics

3.1 Introduction

The topic of this chapter is evolutionary robotics (ER) and evolutionary algorithms (EAs) which EAs are used for generating and optimizing the artificial brains of robots. Evolutionary Robotics is the application of artificial evolution to robotic systems with a sensory-motor interface to the world. EAs are methods for search and optimization based on darwinian evolution, which will be described further in the later section.

3.2 The Origin of Evolutionary Robotics

Evolutionary robotics is a new technique for the automatic creation of autonomous robots. Inspired by the Darwinian principle of selective reproduction of the fittest, it views robots as autonomous artificial organisms that develop their own skills in close interaction with the environment and without human intervention. Drawing heavily on biology and ethology, it uses the tools of neural networks, genetic algorithms, dynamic systems, and biomorphic engineering. The resulting robots share with simple biological systems the characteristics of robustness, simplicity, small size, flexibility, and modularity.

The term evolutionary robotics has been introduced only quite recently (Cliff, Harvey and Husband) (Cliff et al., 1993), but the idea of this kind of control system of a robot as an artificial chromosome subject to the laws of genetics and of natural selection dates back to the end of the 1980's when the first simulated artificial organisms with a sensory motor system began evolving on computer screens. At that time, however, real robots were still machines that requires accurate programming efforts and careful manipulation, Toward the end of that period, a few engineers began questioning some of the basic principles of robot design and came up with a new generation of robots that shard important characteristics with simple biological systems. Above all, these robots were designed so that they could be programmed and controlled by people with different backgrounds and levels of technical skills. In the year 1992 and 1993, the first experiments on artificial evolution of autonomous robots were reported by the team at the Swiss Federal institute of Technology in Lausanne, by a team at the University of Sussex at Brighton, and by a team at the University of Southern California. The success and potentials of these researches triggered a whole new activity in evolutionary robotics in labs across Europe, Japan, and the United States.

In very last few years ER has gathered the interest of a large community of researchers with different research interests and backgrounds. Continuous investment, growth, and progress in ER has caused a substantial maturation of the methodology and of the issues involved, and at the same time has generated a diversification of the basic methodology.

3.3 Related Works

Important work on an evolutionary approach to agent control using neural networks has been done by Beer and Gallagher. They explore the evolution of continuous time recurrent neural networks as a mechanism for adaptive agent control using as example tasks chemotaxis and locomotion control for a six legged insect like agent. The networks are based on the continuous. Hoped model but allow arbitrary recurrent connections. They used a standard genetic algorithm (GA) to determine neuron time constants and thresholds and connection weights. Axed number of network parameters are encoded in a straightforward way on bit string genotypes. They report success in their objectives in the case of locomotion control controllers were evolved that in practice generated a tripod gait front and back legs on one side in phase with the middle leg on the opposite side. This was achieved both with and without the use of sensors which measured the angular position of each leg.

Beer develops a dynamical systems perspective on control systems for autonomous agents in fenced by early work in Cybernetics. In further developments of their evolutionary approach. Yamauchi and Beer evolve networks which can control autonomous agents in tasks requiring sequential and learning behavior. The prime focus of Beers use of artificial evolution is as a means of developing models of simple nervous systems in order to test theories of how real nervous systems may work.

Colombetti and Dorigo use Classier Systems (CSs) for robot control. In this work the leeches implementation is used to build a hierarchical architecture of CSs one for each desired behavior plus a coordinating CS. Results are reported which have been generated in simulations and then transferred to a real robot.

Floreano and Mondada were able to run a GA on a real robot in real time rather than a simulation they used the Khepera robot developed at Lausanne. The GA set the weights and thresholds in a simple recurrent network where every sensory input was connected to both motor outputs. The task was to traverse a circular corridor while avoiding obstacles and this work demonstrates that with well designed equipment it is possible to avoid the problems associated with simulations.

In related work with Nol similar experiments were performed to compare evolving in simulation with using a real Khepera robot. The problem of transferring control systems evolved in simulation over to the real robot was discussed. One approach advocated was that of continuing the evolution for some time further on the real robot to compensate for any inadequacies in the simulation. In contrast to Beers primarily scientific motivation this work primarily emphasizes the practical engineering problems.

Koza used the technique of Genetic Programming to develop subsumption architectures for simulated robots engaged in wall following and box moving tasks. Craig, Reynolds also uses Genetic Programming (GP) to create control programs which enable a simple simulated moving vehicle to avoid collisions. He comments that these solutions are brittle vulnerable to any slight changes or to noise. In further work where the testing includes noise he reports that the brittleness problem is overcome and only compact robust solutions survive.

3.4 Characteristics of Evolutionary Robotics

Evolutionary robotics shares many of characteristics with other approaches, such as behavior-based robotics, robot learning, and artificial life.

- Behavior-Based Robotics

  The behavior-based robotics approach is based upon the idea of providing the robot with a collection of simple basic behaviors. The global behavior of the robot emerges through the interaction between those basic behaviors and the environment in which the robot finds itself (Brooks 1986,1999; Arkin 1998).(R.A.Brooks, 1986) (R.A.Brooks, 1999) (Arkin, 1998)Basic behaviors are implemented in separate subparts of the control system and a coordination mechanism is responsible for determining the relative strength of each behavior in a particular moment. Coordination may be accomplished by means of competitive or cooperative methods. In competitive methods only one behavior affects the motor output of the robot in a particular moment. In cooperative methods different behaviors may contribute to a single motor action although with different strength.(Arkin, 1998)

  In this approach, as in evolutionary robotics, the environment plays a central role by determining the role of each basic behavior at any given time. Moreover, these systems are usually designed through a trial and error process in which the designer modifies the current behaviors and progressively increase the number of basic behaviors while testing the resulting global behavior in the environment. However, evolutionary robotics, by relying on an automatic evaluation process, usually makes a larger use of the trial and error process described above. Moreover, while in the behavior-based approach the break down of the desired behavior into simpler basic behaviors is accomplished intuitively by the designer, in evolutionary robotics this is often the result of a self-organizing process. Indeed, the entire organization of the evolving system, including its organization into subcomponents, is the result of an adaptation process that usually involves a large number of evaluations of the interactions between the system and the environment.

- Robot Learning

Robot learning is based on the idea that a control system can be trained using incomplete data and then allowed to rely on its ability to generalize the acquired knowledge to novel circumstances. The general motivation behind this approach is that it may produce better results than approaches based on explicit design, given the well known difficulties of engineering behavioral systems. In some cases the neural control system learns to perform a mapping between sensory inputs and motor states while in other cases learning is used to develop subsystems of the controller. Different learning algorithms can be used for this purpose: back-propagation learning (Rumelhart et al. 1986); reinforcement learning (Barto et al. 1995); classifier systems (Booker et al. 1989); self-organized maps (Kohonen 1982), etc. These algorithms impose different constraints on the type of architecture that can be used and on the quantity and quality of the supervision required from the designer. Evolutionary robotics shares with these approaches the emphasis on self-organization. Indeed, artificial evolution may be described as a form of learning. However, evolutionary robotics differs from robot learning in two respects. The amount of supervision required by evolution is generally much lower and the evolutionary method in principle does not introduce any constraint on what can be part of the self-organization process.

- Artificial Life

Artificial life represents an attempt to understand all life phenomena through their reproduction in artificial systems (typically through their simulation on a computer). More specifically, artificial life provides a unique framework for studying how entities at different levels of organization interact among themselves (Parisi 1997) although at the cost of introducing crude simplifications. To attain this ambitious goal, artificial life relies on the theory of complex dynamical systems and, from an experimental point of view, on the power of computers. A complex dynamical system is a system that can be described at different levels, in which global properties at one level emerge from the interaction of a number of simple elements at lower levels. Global properties are emergent in the sense that, even if they result from nothing else but local interactions among the elements, they cannot be predicted or inferred from a knowledge of the elements or of the rules by which the elements locally interact, given the high nonlinearity of these interactions. Evolutionary robotics shares most of these characteristics with artificial life, but it also stresses the important of using physical robots instead of simulated agents.

By using real robots, several additional factors due to the physical properties of the robot and of the environment must be taken into account (Brooks 1992). Moreover, only types of sensors and actuators can be used. Similarly, the sensory inputs and the motor out puts should necessarily correspond to physical measures or forces; that is, they are grounded representations (Harnad 1990) and can not include any abstract information truly available in the environment can be used for training.

3.5 Artificial Evolution in Swarm Robotic Systems

Evolutionary computational methods are inspired by the natural evolution. In nature, a population of animals struggle to survive and produce the next generation. The principle of survival of the fittest applies: individuals that are fitter within their environments are more likely to survive and also more likely to produce offsprings, transferring their genetic material to the next generation. By this way, nature eliminates weak individuals and the population gets more adapted to the environmental conditions generation by generation.

Early studies on evolving behaviors for swarm robotic systems reported limited success and expressed pessimistic conclusions. One of the earliest studies, Zaera et al. (Zaera et al., 1996) used artificial evolution to develop behaviors for dispersal, aggregation, and schooling in fish. Although they had evolved controllers for dispersal and aggregation successfully; the performance of the evolved behaviors for schooling was considered not very good, and they concluded that for complex actions like schooling, manual design of a controller would require less time and effort than evolving one, mainly due to the difficulty of determining a useful evaluation function for the specific task.

Mataric et al. (Mataric and Clifi, 1996) have made a comprehensive review of the studies until 1996 on evolving controllers to be used in physical robots and they have discussed the key challenges. They addressed approaches and problems such as evolving morphology or controller, evolving in simulation or with real robots, fitness function design, co-evolution, and genetic encodings. They emphasized that for an evolved controller to be beneficial, the effort to produce it in evolution should be less than the effort needed to manually design a controller for the same robotic task. They stated that it has not been the case, yet; but when the challenges and problems are handled, they may become a practical alternative to controllers designed by hand.

Lund et al. used evolution to develop minimal controllers for exploration and homing task in (Lund and Hallam, 1997). They evolved controllers for the Khepera robot (K-Team, Switzerland) for the task considered where the robot was desired to leave a light source, i.e., home, explore the surrounding for some time, and then return back home where it is virtually recharged. To obtain this periodic behavior, they used sampled sensory input and a minimal network architecture without recurrent connections, which can be used to obtain the notion of return period. Instead their evolution exploited the geometrical shape as perceived by robot and produced a suitable controller.

In contrast to some of these pessimistic conclusions, during the recent years optimistic results are being reported on the evolution of swarm behaviors. In the Swarm-bots Project (Dorigo et al., 2005), Baldassarre et al. successfully evolved controllers for a swarm of robots to aggregate and move toward a light source in a clustered formation. Moreover, for this specific task, several distinct movement types emerged: constant formation, amoeba (extending and sliding), and rose (circling around each other). In (Dorigo et al., 2005), Trianni et al. also evolved successful controllers for a swarm of robots that can grip each other, called a swarm-bot, to fulfill tasks such as aggregation, coordinated motion in a common direction, cooperative transport of heavy loads (as in ants), and all-terrain navigation to avoid holes (connected in swarm-bot formation). Their evolved controllers made use of sound sensors, traction sensors, and flexible links. Trianni et al. has also identified two types aggregation behaviors emerged from evolution: a dynamic and a static clustering behavior. In static clustering, robots move in circles until they are attracted to a sound source. Then they bounce against each other until an aggregate is formed. The clusters are tight and static with the robots involved turning on the spot, whereas in dynamic aggregation, the clusters are loose and they flock around. This study is a good example of evolution of different strategies, or behaviors, for a specific task. Furthermore, in (Dorigo et al., 2005) Dorigo et al. evolved aggregation behaviors for a swarm robotic system. They analyzed two of the evolved behaviors and showed that evolution was able to discover rather scalable behaviors.

Despite these studies, the use of artificial evolution to generate swarm robotic behaviors for a desired task is a rather unexplored field of study. The effort in using evolutionary methods can be reduced by suggestions on choosing parameters required in applying artificial evolution to swarm robotics. To the best of our knowledge, no systematic study has been made to investigate effects of parameters to help such choices. In this study, we addressed this lack of systematic analysis studies to deduce some rules of thumb on the choice of some parameters used in evolution of swarm robotic behaviors.

3.6 Controller Design

The control operate of evolutionary robotics is as follows:

1. An initial population of different artificial chromosomes, each encoding the control system of a robot, are random create and put in the environment.

2. Each robot is then let free to act (move, look around, manipulate) according to a genetically specified controller while its performance on various tasks is automatically evaluated.

3. The fittest robots are allowed to reproduce by generating copies of their genotypes with the addition of changes introduced by some genetic operators (mutations, crossover, duplication).

4. Repeat step 2 for a number of generations until an individual is born which satisfies the performance criterion (fitness function) .

In this chapter, we thesis, we aimed at applying artificial neural networks to evolutionary robotics.

3.7 Bootstrap Problem

Evolutionary algorithms have been widely used to design controllers, and especially neurocontrollers, for robots (Murphy et al., 2000) (Şahin, 2005) (Şahin and Winfield, 2008). Compared to other learning methods, one of their main strength is their ability to evolve both the structure and the parameters of complex architectures using a high-level reward

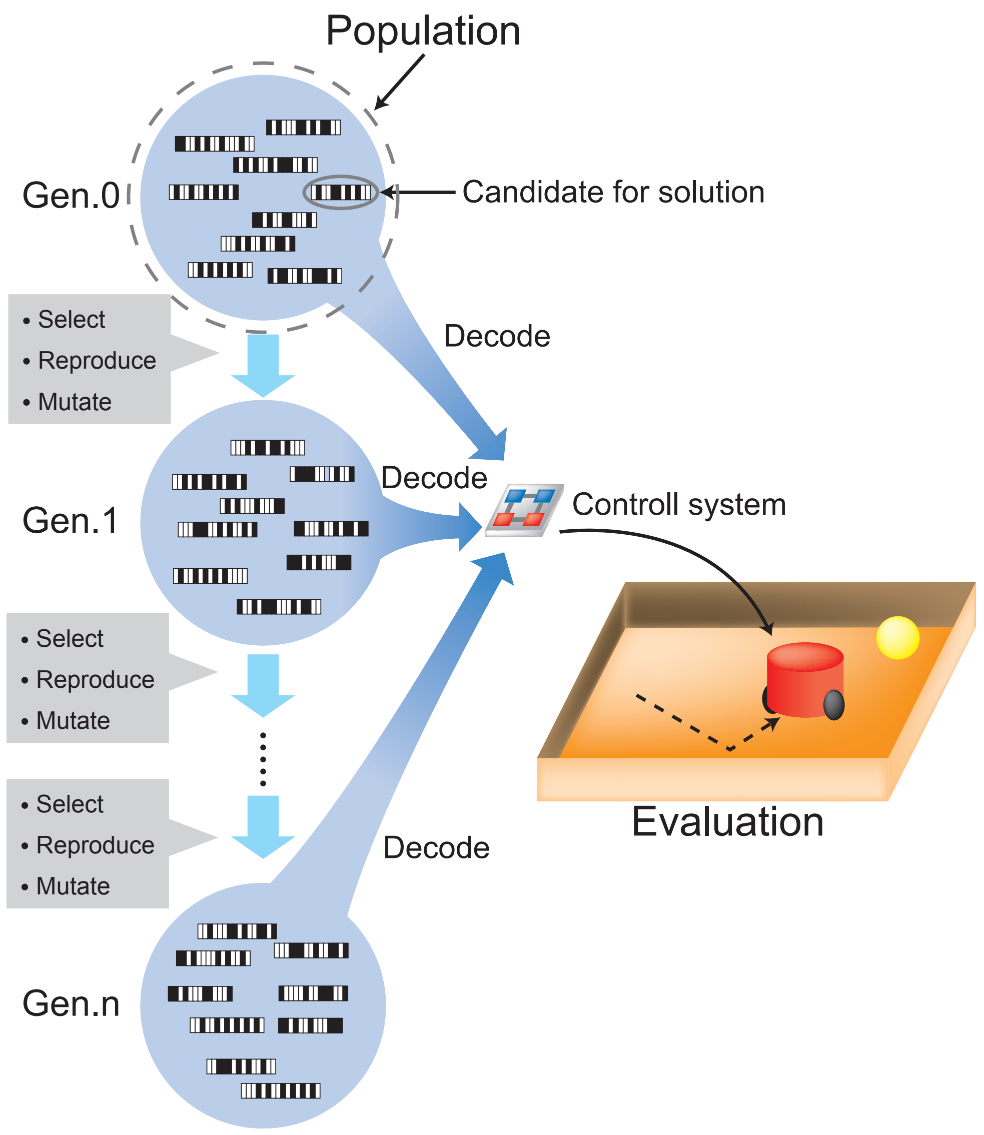


Figure 3.1: Basic Evolutionary Robotics methodology

function. However, this huge amount of work hides many unsuccessful attempts to evolve complex behaviors by only rewarding the performance of the global behavior. The bootstrap problem is often viewed as the main cause of this difficulty, and consequently as one of the main challenges of evolutionary robotics: if the objective is so hard that all the individuals in the first generation perform equally poorly, evolution cannot start and no functioning controllers will be found. For instance, it has been found hard to evolve a light-seeking behavior for a robot in a complex arena without having first evolved an obstacle-avoidance reflex (Şahin et al., 2008).

In consequence, many researchers added some kind of rewards for intermediate steps, leading to successful incremental evolution processes (Şahin and Winfield, 2008) (Kube and Zhang, 1997). Nonetheless, these evolutionary approaches rely on some assumptions that require an accurate knowledge of the problem to solve and can lead the evolutionary algorithm to a local extremum. Most of them, for instance, require to precisely order the different subtasks or to determine when to switch from a sub-task to another one. These biases prevent them from scaling up well to more complex or more open tasks; remarkably, most of them have been tried with only two or three sub-tasks.

Another idea to bootstrap an evolutionary process is to efficiently explore the neighborhood of current candidate solutions to find one with a non-minimal fitness. This concept is close to diversity-preserving mechanisms, widely investigated in evolutionary computation (Dorigo et al., 2005), which typically rely on a distance between the genotypes or the phenotypes. However, such distances are conceptually and computationally difficult to employ with complex structures as neural networks.

Drawing inspiration from these diversity-preserving mechanisms and adapting them to evolutionary robotics, we introduce in this paper the behavioral diversity, an original method to maintain the diversity in a population that relies on a distance between behaviors and a multi-objective evolutionary algorithm (MOEA, (Crespi et al., 2008)). We then show how such a diversity preservation mechanism can efficiently overcome the bootstrap problem in an evolutionary robotics experiment in which a simulated robot that has to successively reach some lights in an arena. As a reference point, we compare the results obtained using this new approach with the ones obtained with multi-subgoal evolution (Beni and Wang, 1989), a recently published incremental method based on multi-objective evolution.

3.8 Summary

This chapter described that the evolutionary robotics is the application of artificial evolution to robotic systems. This approach is fit for robotic systems with a sensory-motor interface to the world. The use of artificial evolution to generate swarm robotic behaviors for a desired task is a unexplored field so far. The next chapter is the introduction of artificial neural networks, which implement simplified models of their biological counterparts and biological neural networks

Chapter 4

Complex Networks

4.1 Introduction

A network is, in its simplest form, a collection of points joined together A in pairs by lines. In the jargon of the field the points are referred to as vertices' or nodes and the lines are referred to as edges. Many objects of interest in the physical, biological, and social sciences can be thought of as networks and thinking of them in this way can often lead to new and useful insights.

4.2 Example of Networks

One of the best known and most widely studied examples of a network is the Internet, the computer data network in which the vertices are computers and the edges are physical data connections between them, such as optical fiber cables or telephone lines. Figure 1.1 shows a picture of the structure of the Internet, a snapshot of the network as it was in 2003, reconstructed by observing the paths taken across the network by a large number of Internet data packets traveling between different sources and destinations. It is a curious fact that although the Internet is a man-made and carefully engineered network we don't know exactly what its structure is, since it was built by many different groups of people with only limited knowledge of each other's actions and little centralized control. Our best current data on its structure are derived from experimental studies, such as the one that produced this figure, rather than from any central repository of knowledge or coordinating authority.

There are a number of excellent practical reasons why we might want to study the network structure of the Internet. The function of the Internet is to transport data between computers (and other devices) in different parts of the world, which it does by dividing the data into pieces or packets and shipping them from vertex to vertex across the network until they reach their intended destination. Certainly the structure of the network will affect how efficiently it accomplishes this function and if we know the network structure we can address many questions of practical relevance. How should we choose the route by which data are transported? Is the shortest route always necessarily the fastest? If not, then what is, and how can we find it? How can we avoid bottlenecks in the traffic flow that might slow things down? What happens when a vertex or an edge fails (which they do with some regularity)? How can we devise schemes to route around such failures? If we have the opportunity to add new capacity to the network, where should it be added?

4.3 Modularity and Community Structure in Networks

Many systems of scientific interest can be represented as networks-sets of nodes or vertices joined in pairs by lines or edges. Examples include the Internet and the worldwide web, metabolic networks, food webs, neural networks, communication and distribution networks, and social networks. The study of networked systems has a history stretching back several centuries, but it has experienced a particular surge of interest in the last decade, especially in the mathematical sciences, partly as a result of the increasing availability of large-scale accurate data describing the topology of networks in the real world. Statistical analyses of these data have revealed some unexpected structural features, such as high network transitivity, power-law degree distributions, and the existence of repeated local motifs.

Past work on methods for discovering groups in networks divides into two principal lines of research, both with long histories. The first, which goes by the name of graph partitioning, has been pursued particularly in computer science and related fields, with applications in parallel computing and VLSI design, among other areas (Yim et al., 2001).

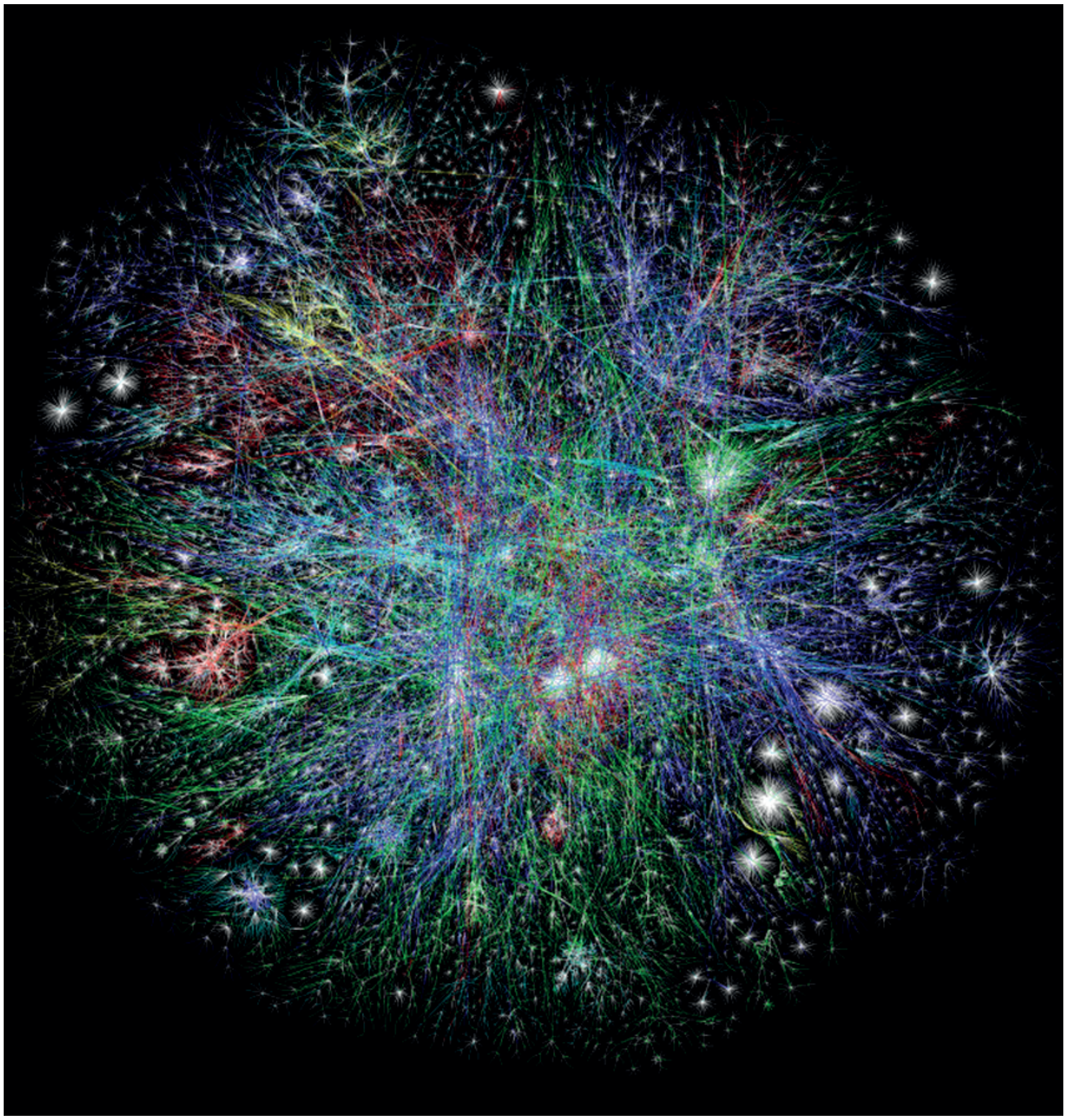


Figure 4.1: The network structure of the Internet. The vertices in this representation of the Internet are ”class C subnets”-groups of computers with similar Internet addresses that are usually under the management of a single organization-and the connections between them represent the routes taken by Internet data packets as they hop between subnets. The geometric positions of the vertices in the picture have no special meaning; they are chosen simply to give a pleasing layout and are not related, for instance, to geographic position of the vertices. Figure created by the Opte Project (www.opte.org).

The second, identified by names such as block: The vertices in many networks fall naturally into groups or communities, sets of vertices (shaded) within which there are many edges, with only a smaller number of edges between vertices of different groups. Modeling, hierarchical clustering, or community structure detection, has been pursued by sociologists and more recently also by physicists and applied mathematicians, with applications especially to social and biological networks.

Community structure detection, by contrast, is per 2 haps best thought of as a data analysis technique used to shed light on the structure of large-scale network datasets, such as social networks, Internet and web data, or biochemical networks. Community structure methods normally assume that the network of interest divides naturally into subgroups and the experimenter's job is to find those groups. The number and size of the groups is thus determined by the network itself and not by the experimenter. Moreover, community structure methods may explicitly admit the possibility that no good division of the network exists, an outcome that is itself considered to be of interest for the light it sheds on the topology of the network.

4.4 Modularity Optimization

Suppose then that we are given, or discover, the structure of some network and that we wish to determine whether there exists any natural division of its vertices into nonover lapping groups or communities, where these communities may be of any size.

Let us approach this question in stages and focus initially on the problem of whether any good division of the network exists into just two communities. Perhaps the most obvious way to tackle this problem is to look for divisions of the vertices into two groups so as to minimize the number of edges running between the groups. This *minimum cut* approach is the approach adopted, virtually without exception, in the algorithms studied in the graph partitioning literature. However, as discussed above, the community structure problem differs crucially from graph partitioning in that the sizes of the communities are not normally known in advance. If community sizes are unconstrained then we are, for instance, at liberty to select the trivial division of the network that puts all the vertices in one of our two groups and none in the other, which guarantees we will have zero intergroup edges. This division is, in a sense, optimal, but clearly it does not tell us anything of any worth. We can, if we wish, artificially forbid this solution, but then a division that puts just one vertex in one group and the rest in the other will often be optimal, and so forth.

The problem is that simply counting edges is not a good way to quantify the intuitive concept of community structure. A good division of a network into communities is not merely one in which there are few edges between communities; it is one in which there are fewer than expected edges between communities. If the number of edges between two groups is only what one would expect on the basis of random chance, then few thoughtful observers would claim this constitutes evidence of meaningful community structure. On the other hand, if the number of edges between groups is significantly less than we expect by chance-or equivalently if the number within groups is significantly more-then it is reasonable to conclude that something interesting is going on. This idea, that true community structure in a network corresponds to a statistically surprising arrangement of edges, can be quantified using the measure known as modularity (Bonabeau et al., 1999). The modularity is, up to a multi-placative constant, the number of edges falling within groups minus the expected number in an equivalent network with edges placed at random. (A precise mathematical formulation is given below.) The modularity can be either positive or negative, with positive values indicating the possible presence of community structure. Thus, one can search for community structure precisely by looking for the divisions of a network that have positive, and preferably large, values of the modularity (Şahin, 2005).

The evidence so far suggests that this is a highly effective way to tackle the problem. For instance, Guimer‘a and Amaral (Guimera and Amaral, 2005) and later Danon et al. (Danon et al., 2005) optimized modularity over possible partitions of computer generated test networks using simulated annealing. In direct comparisons using standard measures, Danon et al. found that this method outperformed all other methods for community detection of which they were aware, in most cases by an impressive margin. On the basis of considerations such as these we consider maximization of the modularity to be perhaps the definitive current method of community detection, being at the same time based on sensible statistical principles and highly effective in practice.

Unfortunately, optimization by simulated annealing is not a workable approach for the large network problems facing today's scientists, because it demands too much computational effort. A number of alternative heuristic methods have been investigated, such as greedy algorithms (Newman, 2004) and extremal optimization (Duch and Arenas, 2005). Here we take a different approach based on a reformulation of the modularity in terms of the spectral properties of the network of interest.

Chapter 5

Neuroevolution Based on Covariance Matrix Adaptation Evolution Strategy

5.1 CMA-NeuroES

CMA-ES is evolutionary function optimization proven efficient for a variety of test functions and benchmark problems (Hansen and Ostemeier, 1996). CMA-ES optimization is based on mutation. Each solution point $x_k^{(g)}$ at generation g in this algorithm presents a n-dimension real-valued decision variable vector. These variables are altered by recombination and mutation, which correspond to calculating the mean value of μ solution points selected from offspring λ. In this algorithm, mutation is used to add a normally distributed random vector with zero mean and the covariance matrix are updated during evolution to improve searching. Formally, solution point $x_k^{(g+1)}$ of offspring $k = 1, \dots , \lambda$ created in generation g are calculated by

$$x_k^{(g+1)} = m^{(g)} + N_k^{(g)}(0, \sigma^{(g)^2} C^{(g)}) \tag{5.1}$$

This is realized by adding a zero-mean random vector drawn from multivariate normal distribution specified with step size $\sigma^{(g)}$ and covariance matrix $C^{(g)}$. $m^{(g)}$ denotes the mean value of the population in generation g and $N_k^{(g)}(0, \sigma^{(g)^2} C^{(g)})$ is a multivariate normal distribution with zero mean and covariance matrix C in the g-th generation.

CMA-ES efficiency is provided by self-adaptation of C and σ. This allows CMA-ES to search efficiently in highly correlated search space. For details, see reference (Hansen and Ostemeier, 1996).

CMA-NeuroES is a weight evolving artificial neural network that applies CMA-ES to weight optimization. Weight optimization of neural networks has highly correlated search space. Since the adaptation of C allows efficient searching in the existence of correlation between parameters, we expect that CMA-NeuroES would show good performance on the optimization of the synaptic weights for our robot controller.

Every robot in our SRS have the same type of CMA-NeuroES controller. Each robot gets 16 input information from the environment. Our robots act independently after calculating different input data. Swarm behavior fitness is calculated from a fitness table to evaluate a robotic swarm and then update mean value m, covariance matrix C and global step-size σ. This is called a one-generation loop. CMA-NeuroES operates in five steps:

Algorithm 1 CMA-ES

```
 1: procedure CMA–ES
 2:     Initialize:
 3:     $\mathbf{x}_k^{(g+1)} \leftarrow \mathbf{0}, \sigma \leftarrow \sigma_{init}, \mathbf{C} \leftarrow \mathbf{I}, g=0$
 4:     while Stop condition is not satisfied do
 5:         for $k$=1 to $\lambda$ do
 6:             $x_k^{(g+1)} = m^{(g)} + N_k^{(g)}(0, \sigma^{(g)^2} C^{(g)})$
 7:         end for
 8:         select $\mu$ solution points from offspring $\lambda$
 9:         adapt mean value $m^{(g)}$ accordingly
10:         adapt step size $\sigma$ accordingly
11:         adapt covariance matrix $\mathbf{C}$ accordingly
12:     end while
13: end procedure
```

Step 1: Set all synaptic neural network weights randomly at initial generation. If it is not the first generation, create offspring from (1).

Step 2: Start simulation, then evaluate the fitness of λ offspring by using the fitness table.

Step 3: Send fitness and μ parents to CMA-NeuroES to create new offspring, and update all synaptic weights.

Step 4: Choose synaptic weights with higher fitness as parents for the next generation.

Step 5: Repeat step 1 to start a new generation until the terminal condition is met.

5.2 Simulation Experiment with 10 robots

5.2.1 Cooperative Package Pushing Problem

Experimental Setup

The purpose of the cooperative package pushing problem is that robots cooperate in pushing all three packages to the right side of the goal line within 2,000 time steps, which means the center of a package should pass the goal line. The SRS is evaluated by points based on rules in Table 5.1. A swarm gets 100 points when a robot touches a package. For each package, a swarm also gets its moving distance of x axis' s toward the goal line as a moving points. When a package reaches a goal, a swarm gets 1,000 bonus points and the distance that its x axis is to be moved.

The objective of the food foraging problem is to collect all food and bring it back to the nest within 3,000 time steps. In a preliminary experiment, the swarm could not generate foraging behavior because the swarm has got the same point no matter robots touched food once then left or touched the food all the time. One of the key points in the foraging task thus becomes how to increase chances for touch between robots with food. As a result, the fitness function was changed as shown in Table 5.2. A swarm gets 0.1 points in each time step when a robot touches a package. In the 3,000th time step, a swarm gets the same point as the distance that packages are pushed toward the goal line. When food is collected, a swarm gets 1,000 bonus points and the moving distance of its x axis. If the swarm solves the food foraging problem successfully, it gets the same number of points as the remaining

Table 5.1: Fitness table for cooperative package pushing problem

robots touched packages	+100
package move towards goal line	+ moving distance of x axis
packages goals	+ moving distance of x axis
bonus when a package goals	+ 1000

Table 5.2: Fitness table for food foraging problem

Robots touch packages	+0.1/time step
Package moved towards nest	+ moving distance of *x* axis
Packages goals	+ moving distance of *x* axis
Bonus when goals	+ 1000
Bonus when problem complete	+ Left time steps

Table 5.3: 10-robot cooperative package pushing problem

Strategy	Success Rate	Max Fitness Point	First Success Generation	Average Evaluation Times
FES	7/10	12023	122	2100
FESplus	7/10	12056	48	1400
DE/rand/1/bin	7/10	11912	84	4900
CMA-NeuroES	10/10	12058	27	1820

time steps as a bonus.

To compare CMA-NeuroES with others approaches, we usedthree evolutionary algorithms: (μ, λ)-FastES (FES) (Yao and Liu, 1997), (μ+λ)-FastES (FESplus) and differential evolution (DE) (Storn and Price, 1997) are adopted. Three patterns in our cooperative package pushing experiment decrease the number of robots from ten to seven in simulation as a preliminary experiment.

The parameter setting of the four evolutionary algorithms is as follows. FES parameter setting: FES and FESplus have the same number of parents μ= 10, offspring λ= 70, strategy parameter: [1e-8, 3.0], start from 1.0. DE parameter setting: we use the best-performing DE parameter from our preliminary experiment. We tested DE by changing its crossover constant (CR) from 0.0 to 1.0 and the select weighting factor (F) from 0.5 to 1.0 in the ten-robot cooperative transport problem. As a result, the best-performing DE/rand/1/bin with population NP=70, scale factor F =0.2 and crossover rate CR = 0.9 were chosen. CMA-NeuroES parameters are set at the number of offspring λ= 70, initial standard deviation 0.2 and initial covariance matrix C= I.

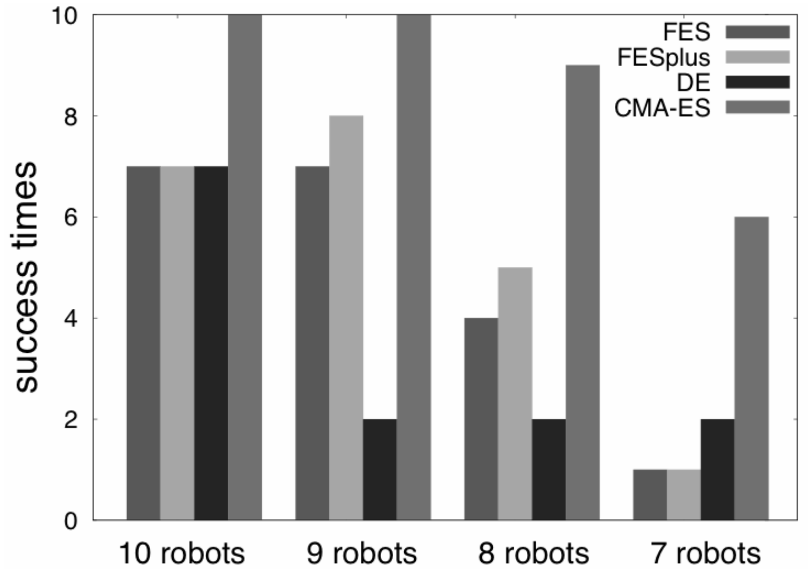


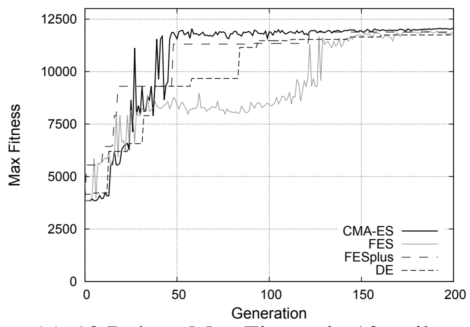
Figure 5.1: Success rate of cooperative package pushing

5.2.2 Results

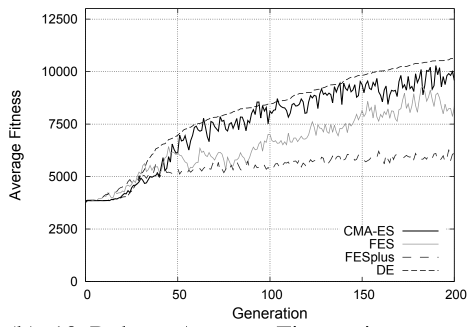
Fig. 5.1 shows success times for the cooperative package pushing problem. CMA-NeuroES succeeds in all trials for the 10-robot and 9-robot experiments. When the success rate of other algorithms decreased sharply in 8-robot and 7-robot experiments, CMA-NeuroES still has a high success rate of 90% and 60%, respectively.

Results for the 10-robot cooperative package pushing problem are given in Table 5.3. We get maximum fitness point information from Table 5.3 showing that the CMA-NeuroES has the highest max fitness value. First success generation is the generation number that the problem was solved at the first time in 10 trials and the fastest CMA-NeuroES takes only 27 generations to succeed. The average evaluation time here refers to the average search speed of the algorithm. In this pattern, FESplus is fastest. Fig. 5.2 shows the learning history of four evolutionary algorithms' maximize and average fitness. Typical behavior observed in one simulation trial for a robot swarm with a CMA-NeuroES controller is shown in Fig. 7. Robots pushed a package located nearest to the start line smoothly (Fig. 5.3(a)), then robots started to separate into three subgroups to push all packages over the goal line at the same time (Fig. 5.3(b)-5.3(f)).

Results for the 9-robot cooperative package pushing problem are shown in Table 5.4. Max fitness of CMA-NeuroES is higher than that for other algorithms. FES is faster than

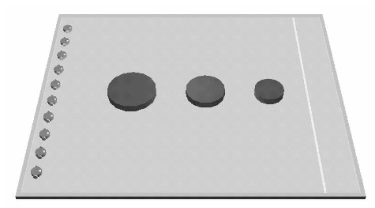


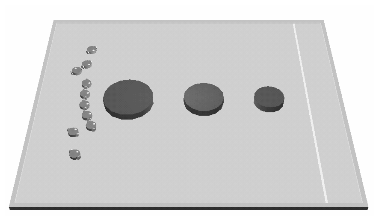
(a) 10 Robots Max Fitness in 10 trails

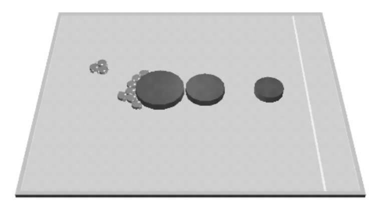


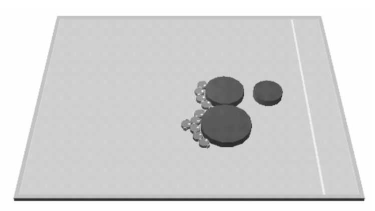
(b) 10 Robots Average Fitness in success trails

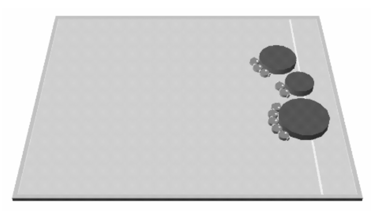
Figure 5.2: Experiment results (10 robots)


(a) Snapshot 1


(b) Snapshot 2


(c) Snapshot 3


(d) Snapshot 4


(e) Snapshot 5

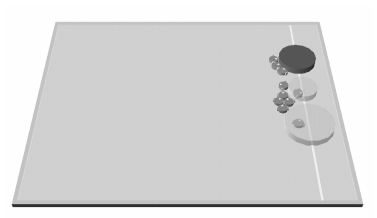
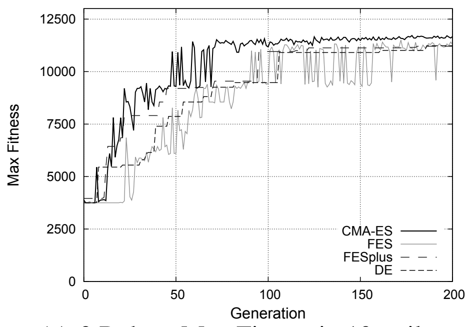
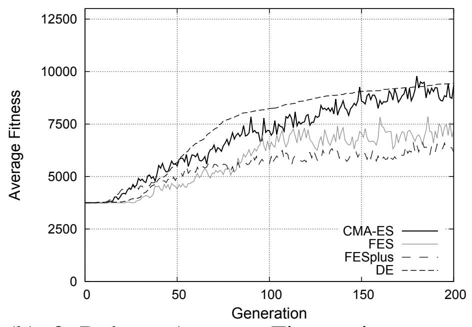

(f) Snapshot 6

Figure 5.3: Snapshot of cooperative behavior (10 robots)

FESplus in succeeding, but CMA-NeuroES still performs the best. FESplus again gets a low average evaluation time for its fast search ability. Fig. 5.4 shows the learning history of max and the average fitness for the four algorithms with 9 robots. By decreasing the number of robots, the SRS must take more generations to succeed. Typical behavior observed in one simulation trial for a SRS with a CMA-NeuroES controller is shown in Fig. 5.5. Robots push a package located nearest to the start line (Fig. 5.5(b)). In this pattern, it becomes impossible for robots to push three packages over the goal line at the same time due to the shortage of robots. Snapshots in Fig . 5.5 shows that our SRS pushed the two small packages first (Fig. 5.5(e)), and then robots turned back to cooperate to push the heaviest package over the goal line (Fig. 5.5(f)).

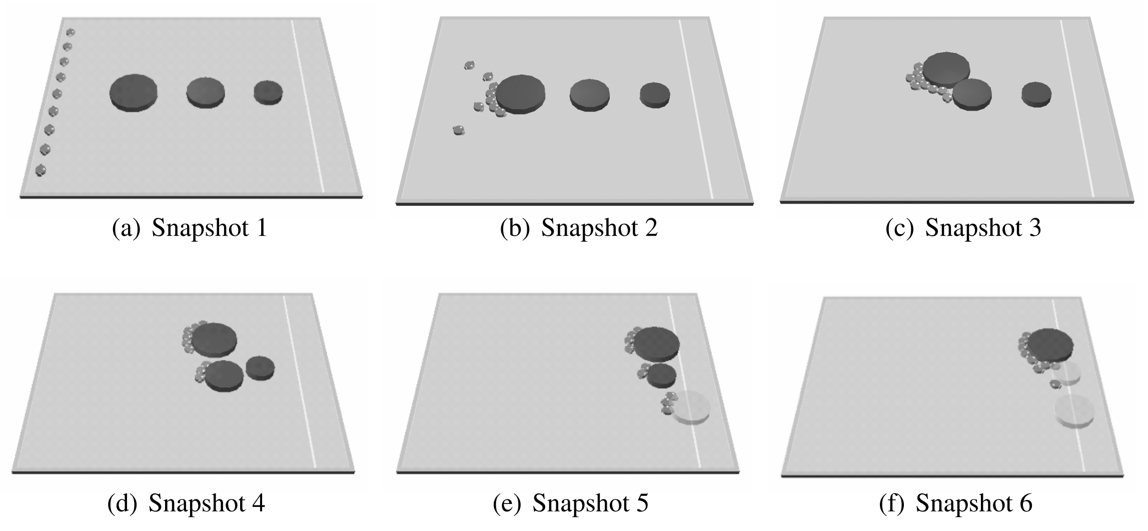


(a) 9 Robots Max Fitness in 10 trails



(b) 9 Robots Average Fitness in success trails

Figure 5.4: Experiment results (9 robots)



(a) Snapshot 1 (b) Snapshot 2 (c) Snapshot 3

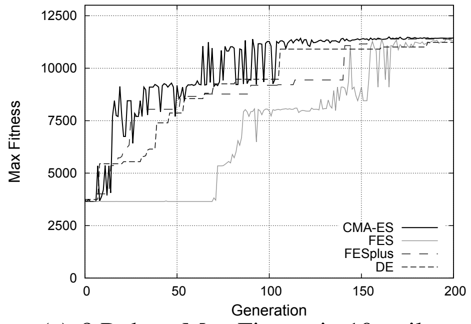
(d) Snapshot 4 (e) Snapshot 5 (f) Snapshot 6

Figure 5.5: Snapshot of cooperative behavior (9 robots)

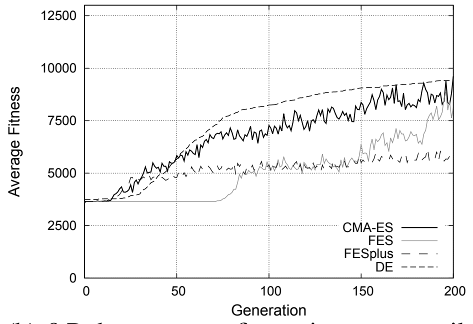
Results for the 8-robot cooperative package pushing problem are shown in Table 5.5. Max fitness for CMA-Neuro ES is higher than that of others. The first success generation is also as we expected that CMA-NeuroES has the over all fastest speed. FESplus with the low success rate gets lower average evaluation time. Fig. 5.6 shows the learning history of max and the average fitness for the 4 evolutionary algorithms with 8 robots. Some of these robots have problem choosing which package to push at first as the time limitation lapses. Typical behavior in one simulation trial for a robot swarm with a CMA-NeuroES controller is shown in Fig. 5.7. This time, the robots starting by pushing the heavier packages first (Fig . 5.7(b)), then pushing the two heavier to the goal line (Fig. 5.7(e)). After that, robots

Table 5.4: 9-robot cooperative package pushing problem

Strategy	Success Rate	Max Fitness Point	First Success Generation	Average Evaluation Times
FES	7/10	11704	91	12054
FESplus	8/10	11590	95	9044
DE/rand/1/bin	2/10	11231	106	13342
CMA-NeuroES	10/10	11782	48	6776



(a) 8 Robots Max Fitness in 10 trails

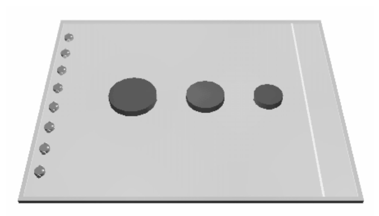


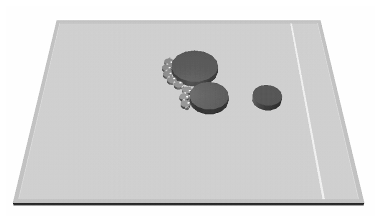
(b) 8 Robots average fitness in success trails

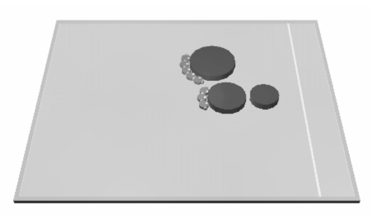
Figure 5.6: Experiment results (8 robots)

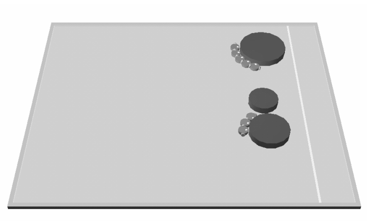
turn back and push the small package to the goal line (Fig. 5.7(f)). This may be recognized as smart behavior coordinated by CMA-NeuroES.

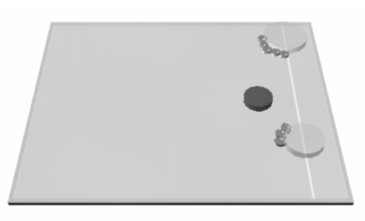
A typical failure in the cooperate package pushing problem is shown in Fig. 5.7. At the start, all robots move toward the first package, which requires five or more robots to move it, and start to push it from the start position (Fig. 5.7(a)- Fig.5.7(b)). Robots then separate into two groups to push two packages in front of them Fig. 5.7(c). 5 robots push the largest packages toward the down side of the field while the other 3 robots push the second package toward the goal line (Fig. 5.7(c)). After the 5 robots pushed the first package to the down side, they left it and searched for a new package (Fig. 5.7(d)), with 4 of the 5 finding the second package and one leaving the group and moving toward the goal line itself. The robots separate into two groups and a single robot, with 5 robots trying to push the second package to the goal line and 2 robots pushing the last package. The single robot passed the goal line at that time (Fig. 5.7(e)). Two packages are successfully moved over the goal line and the first package is still left in the field, when the time limitation is reached, meaning that robots cannot behave so as to return to other packages (Fig. 5.7(f)). This failure infers that to complete the problem, the robot controller should make full use of time and decide


(a) Snapshot 1


(b) Snapshot 2


(c) Snapshot 3


(d) Snapshot 4


(e) Snapshot 5

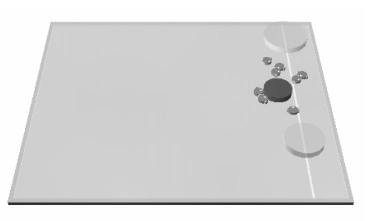

(f) Snapshot 6

Figure 5.7: Snapshot of cooperative behavior (8 robots)

Table 5.5: 8-robot cooperative package pushing problem

Strategy	Success Rate	Max Fitness Point	First Success Generation	Average Evaluation Times
FES	4/10	11420	144	13573
FESplus	5/10	11390	141	11221
DE/rand/1/bin	2/10	11231	106	13265
CMA-NeuroES	9/10	11494	64	11935

which package to push first.

As a result, we made the following hypotheses: the difficulty of this cooperate package pushing problem is in letting robots learn to turn back and push the rest of the packages because the shortage of robots make it impossible for them to goal at the same time. If we continue decreasing the number of robots, we can see much more clearly that CMA-NeuroES performs better in our problem. To test these hypotheses, we conducted a 7-robot experiment as followed.

We decreased the robot number to 7 and increased time steps into 2,700 to avoid time limitation failure. Result in Table 5.6 show that FES (1 success) and FESplus (1 success) almost failed this time. DE (2 successes) performed better and CMA-NeuroES (6 successes) had the best performance. We also infer that CMA-NeuroES max fitness still

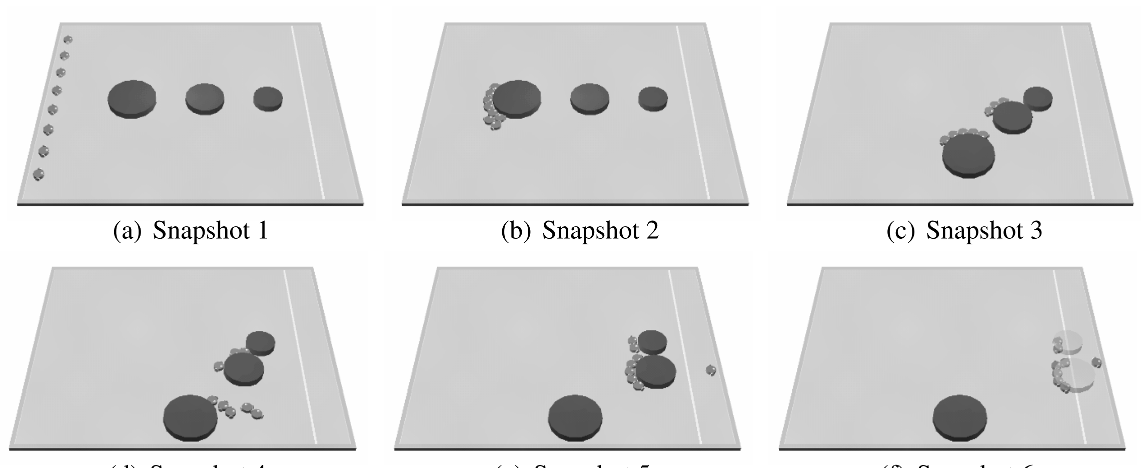

(a) Snapshot 1 (b) Snapshot 2 (c) Snapshot 3
(d) Snapshot 4 (e) Snapshot 5 (f) Snapshot 6

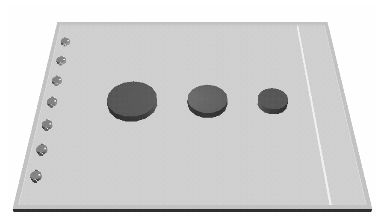
Figure 5.8: Snapshot of failed cooperative trial behavior (8 robots)

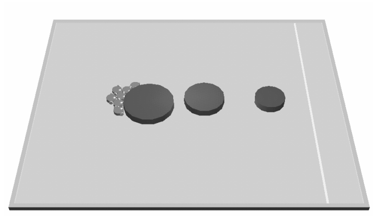
Table 5.6: 7-robot cooperative package pushing problem

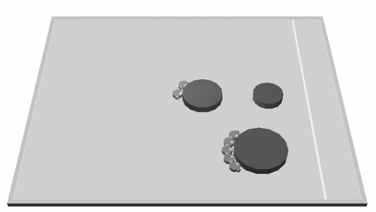
Strategy	Success Rate	Max Fitness Point	First Success Generation	Average Evaluation Times
FES	1/10	11420	156	19890
FESplus	1/10	11390	106	18660
DE/rand/1/bin	2/10	11231	106	18950
CMA-NeuroES	6/10	10985	79	15830

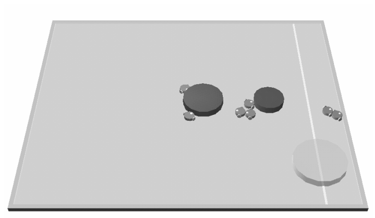
increased very sharply and started to converge after the 130th generation, compared to the other three.

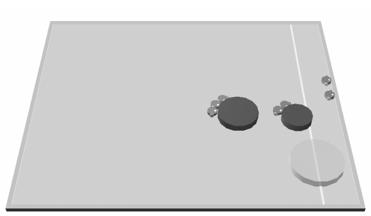
Typical behavior observed in one simulation trial for a 7-robot swarm with a CMA-NeuroES controller is shown in Fig. 5.8. Robots push three packages located nearest to the start line (Fig. 5.8(a)-13(b)), pushing the heaviest package to the goal first (Fig. 5.8(d)) and then turning back and separating into two groups to keep pushing the two small packages to the goal (Fig. 5.8(d)- 5.8(f)). Based on this behavior, we concluded that robots with a CMA-NeuroES controller do not just move forward to push the packages to the goal, but also have learned to turn around and even separate into two subgroups and cooperate to get packages to the goal.


(a) Snapshot 1


(b) Snapshot 2


(c) Snapshot 3


(d) Snapshot 4


(e) Snapshot 5

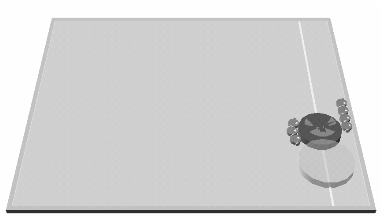
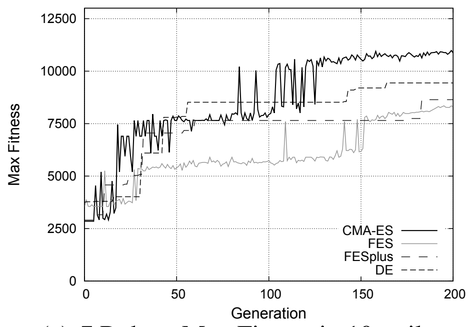
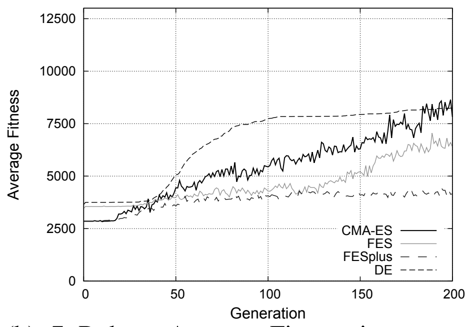

(f) Snapshot 6

Figure 5.9: Snapshot of cooperative behavior (7 robots)



(a) 7 Robots Max Fitness in 10 trails



(b) 7 Robots Average Fitness in success trails

Figure 5.10: Experiment results (7 robots)

5.2.3 Cooperative Food Foraging Problem

5.2.4 Results

Results for the food foraging problem are summarized in Table 5.7. CMA-NeuroES succeeds 10 times in 10 trials again, with the success rate of the other algorithms 60% (FES), 70% (FESplus) and 20% (DE). We also get max fitness value information from Table 5.7 showing that CMA-NeuroES has the highest max fitness. The first success generation shows the fastest succeed generation in 10 trials, the fastest CMA-NeuroES takes only 81

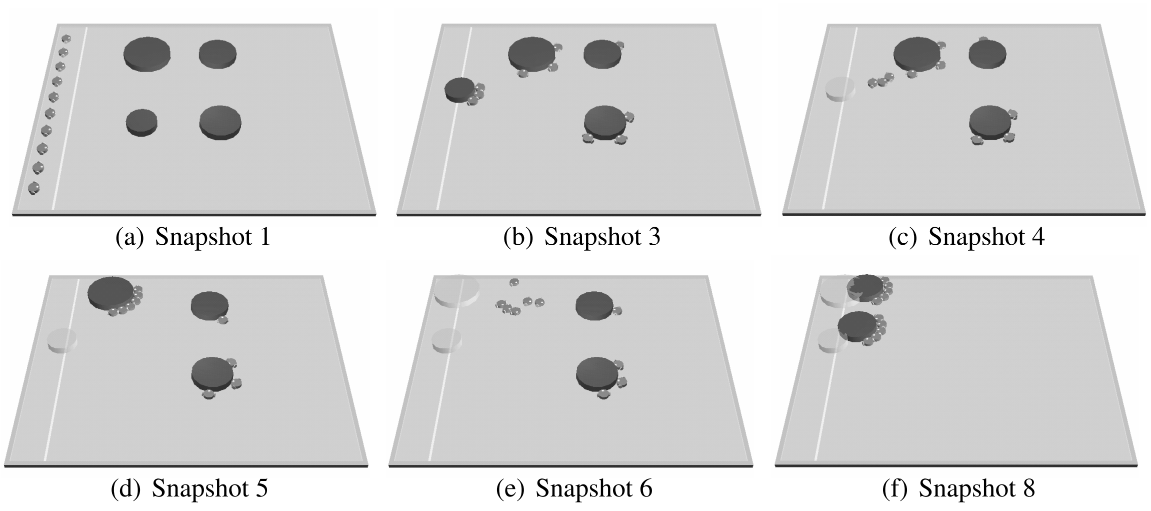
Table 5.7: Food foraging problem

Strategy	Success Rate	Max Fitness Point	First Success Generation	Average Success Generation
FES	6/10	15448	116	180
FESplus	7/10	15542	120	160
DE/rand/1/bin	2/10	15025	461	461
CMA-NeuroES	10/10	15613	81	98

generations to succeed. The average evaluation succeed generation means the average generation an algorithm requires to complete this problem. CMA-NeuroES only required 98 generations to overcome this problem, where as FES required 160 and FESplus required 180. DE performed the worst.

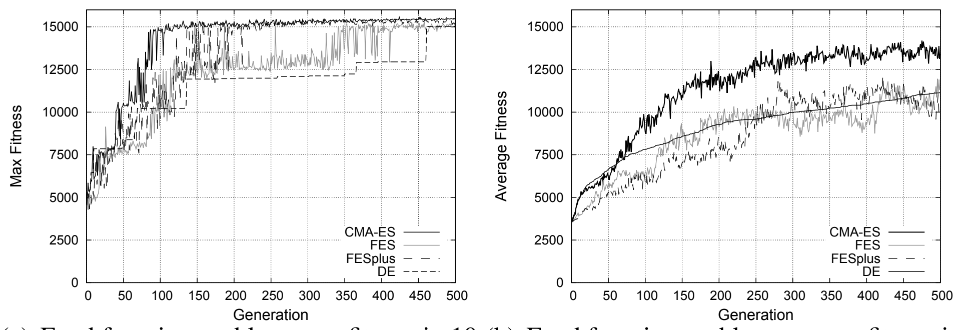
Typical behavior observed in one simulation trial for a SRS with a CMA-NeuroES controller is shown in Fig. 5.11. Robots exit from the nest then all try to touch food (Fig. 5.11(b)). Three of them then start to bring the smallest food back to the nest (Fig. 5.11(c)), while other robots are still contacting three food items still left in the field. As soon as three robots complete collection of their first food, they turn back and work for the nearest food, which requires five robots to move it (Fig. 5.11(d)-5.11(f)). The other two food items have the same weight and require four or more robots to move them. Robots finally separate into two groups and bring the last two food items back to the nest at the same time. Fig. 5.12 shows the learning history of the maximum and average fitness of the four algorithms in the food foraging problem. As we increased the difficulty of the problem, CMA-NeuroES showed fast evolution and a high success rate again. In the food foraging problem, even the average fitness of CMA-NeuroES increased faster than that of the other evolutionary algorithms.

Fig. 5.13 shows the history of step-size σ in 10 trials that represent the CMA-NeuroES evolution path. From this figure, we infer that in the first 125 generations, most of the trials involve searching for the global best optimize solutions by increasing searching distribution, the covariance matrix of CMA-NeuroES. All step-size σ started to decrease after 200 generations, which means CMA-NeuroES found the direction of the optimize solution area that contains the best optimize solution. After that, all σ start to become convergent and are seen to converge.


(a) Snapshot 1 (b) Snapshot 3 (c) Snapshot 4

(d) Snapshot 5 (e) Snapshot 6 (f) Snapshot 8

Figure 5.11: Snapshot of food foraging problem



(a) Food foraging problem max fitness in 10 trails (b) Food foraging problem average fitness in success trails

Figure 5.12: Experiment results (Food foraging problem)

5.3 Simulation Experiment with 100 robots

5.3.1 Cooperative Food Foraging Problem

The cooperative food foraging problem was inspired by the behavior of ants searching for food sources and bringing the food to the nest. The task is to find better search strategies that maximize the ratio of bringing food to the nest in a specified environment (Sugawara and Sano, 1997) (Yu et al., 2013). Figure 6.1 shows the food foraging problem we investigate in this thesis. The field is a 5,000×5,000 length square unit. The nest, a 1,000×1,000 square unit goal area, is located at the center of the field. One hundred autonomous mobile robots are randomly placed in the nest as the initial condition. Three food

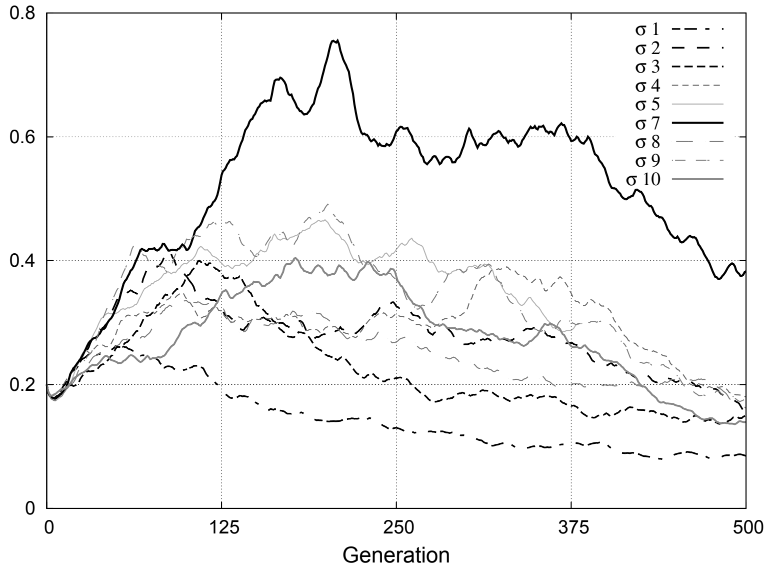


Figure 5.13: Learning history of step-size σ

sources, F, are randomly placed in the field. Every robot is set to be able to move a food source up to a five-unit weight. However, all of the food sources are 24-unit weight, which means that one food source requires at least five robots to move it cooperatively in a specific direction. A new food source appears soon after one food source is collected during 5,000 time steps. Three obstacles are fixed in the field at a given point that we set in the field. The large static friction we set for each obstacle are impossible for robots to move it, which means the SRS must avoid these obstacles and maximize the ratio of bringing food sources to the nest. The goal of cooperative food foraging task is that SRS should collect as many food sources as possible.

5.3.2 Robot Setup

The SRS in this thesis is assumed to be homogenous, i.e., all the robots in the system are assumed to have the same specifications, as shown in Figure 6.2. Each robot is 50 length units in diameter and has two types of sensors: eight infrared (IR) sensors and an omni-Vision camera. The eight IR sensors are arranged around a robot. 4 IR sensors equally distributed in the front of the robot, and 2 IR sensors are equally distributed in the back of

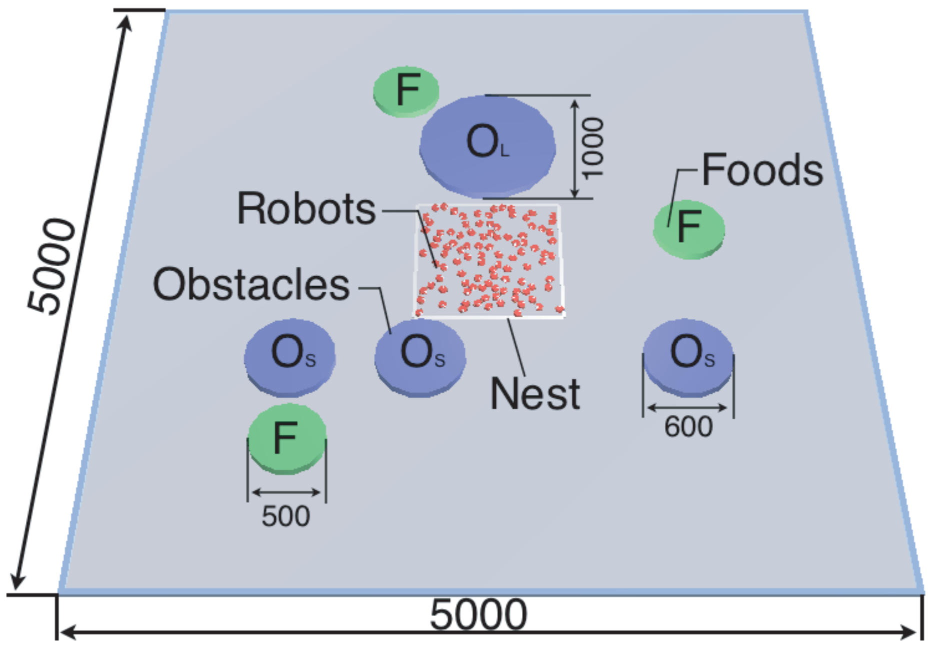


Figure 5.14: The cooperative food foraging problem

robot. The other 2 IR sensors are set at two sides of the robot, separately. Each IR sensor provides a value that is inversely proportional to the distance to an object, which might be a food source, an obstacle, a wall, or other robots within the sensor range of 64 length units. The values are normalized between zero and one. The omni-Vision camera is located at the center of each robot.

The robot's sensor abilities are summarized as follows:

- Distance from an IR sensor to objects: $O_i (i = 0, 1, \cdots, 7)$.
- Distance and direction to the nearest robot: r_{R1}, $\sin\theta_{R1}$ and $\cos\theta_{R1}$.
- Distance and direction to the second nearest robot: r_{R2}, $\sin\theta_{R2}$ and $\cos\theta_{R2}$.
- Distance and direction to the nearest food source: r_{F1}, $\sin\theta_{F1}$ and $\cos\theta_{F1}$.
- Distance and direction to the second nearest food source: r_{F2}, $\sin\theta_{F2}$ and $\cos\theta_{F2}$.
- Direction to the nest: $\sin\theta_N$ and $\cos\theta_N$.

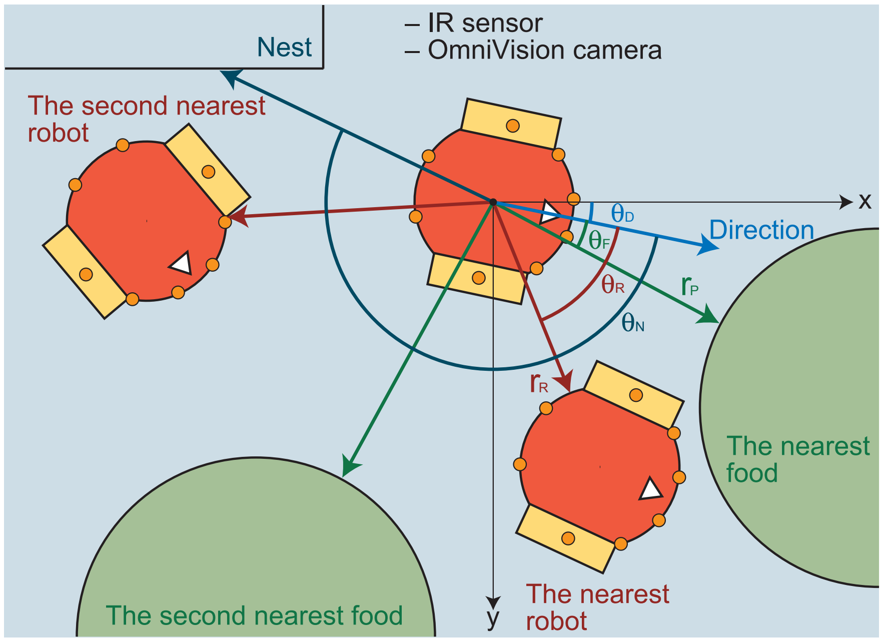


Figure 5.15: Robot information

- Global direction of the robot: $\sin\theta_D$ and $\cos\theta_D$.

Information obtained by the two types of sensor forms an input layer of a robot controller comprising 24 inputs connected to a motor on the right and another motor on the left that controls two differential driven wheels, enabling robot to move forward or to turn left or right using the rotational difference between wheels. In addition, each input neuron receives Gaussian noise, whose mean and standard deviation (SD) are 0 and 0.03, respectively. Four fully inter-connected hidden layers are adopted from our preliminary experiments for computer simulation. Because the two motor wheels are controlled by EANN output, the output layer consists of two neurons. The neurons of recurrent artificial neural networks (RANN) are connected as shown in Figure 6.3, as in our previous study (Yu et al., 2013). Therefore, the number of synaptic connections is 162. All robots are assumed to have the same RANN controller.

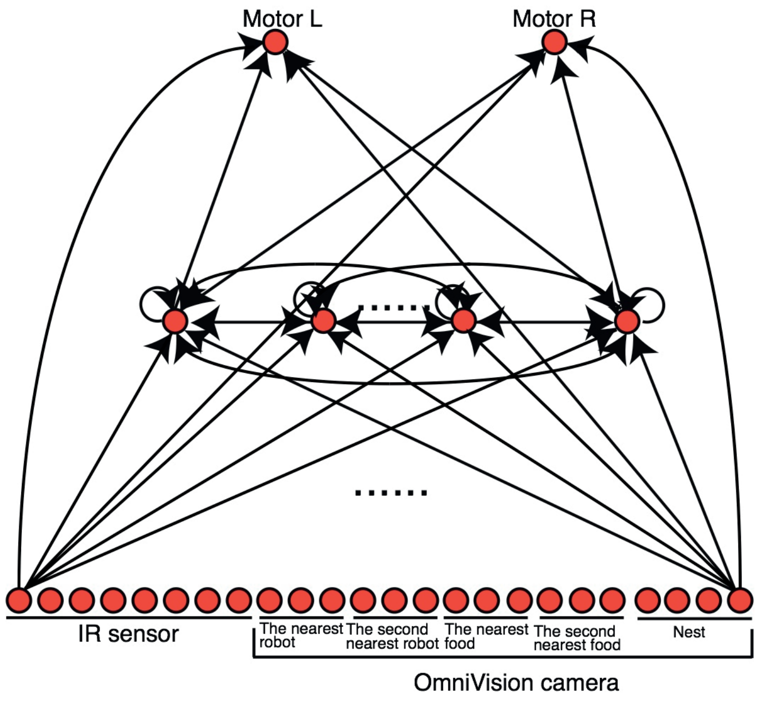


Figure 5.16: Artificial neural networks for robot controller

5.3.3 Experimental Setup

CMA-ES is a stochastic, iterative method for difficult nonlinear and nonconvex optimization problems. It has been proven to be a powerful evolutionary optimization algorithm for a variety of test functions, and benchmark problems, and it performs especially well in searching landscapes with discontinuities, noise, and local optima (Hansen, 2011) (Igel, 2003).

CMA-ES was introduced by Hansen Ostermeier in 1996 (Hansen and Ostemeier, 1996), and its use of covariance matrix adaptation made this evolution strategy a highly elaborate optimization algorithm. After weighted recombination was introduced to CMA-ES in 2001 (Hansen and Ostemeier, 2001), the so-called rank-μ-update greatly reduced time complexity (Hansen et al., 2003) in 2003. The performance of CMA-ES was improved after researchers found that increasing the population size can enhance global search characteristics (Hansen and Kem, 2004). In 2008, Raymond and Hansen presented a new approach that reduces evaluation time and space complexity for CMA-ES (Ros and Hansen, 2008).

In CMA-ES, the offspring for the next generation $(g + 1)$ are generated by sampling a multivariate normal distribution with mean $m \in \mathbb{R}^n$ and covariance $C \in \mathbb{R}^{n \times n}$ (Hansen et al., 2010). Each solution point $x_k^{(g+1)}$ at generation $(g + 1)$ in this algorithm presents an n-dimensional real-valued decision variable vector. These variables are altered by recombination and mutation, which correspond to the calculation of the mean value of μ solution points selected from offspring λ. In this algorithm, mutation is used to add a normally distributed random vector with zero mean, and the covariance matrix is updated during evolution to improve searching. Formally, solution points $x_k^{(g+1)}$ of offspring $k = 1, \dots, \lambda$ created in generation g are calculated as Algorithm.

This is realized by adding a zero-mean random vector drawn from a multivariate normal distribution specified with step size $\sigma^{(g)}$ and covariance matrix $C^{(g)}$. $m^{(g)}$ is the mean value of the population in generation g, and $N_k^{(g)}(0, {\sigma^{(g)}}^2 C^{(g)})$ is a multivariate normal distribution with zero mean and covariance matrix C in the *g-th* generation.

CMA-ES efficiency is provided by self-adaptation of C and σ. This allows CMA-ES to search efficiently in a highly correlated search space. For details, see reference (Hansen et al., 1995) (Hansen and Ostemeier, 1996) (Moriguchi and Honiden, 2012).

CMA-NeuroES is a weight-evolving artificial neural network that applies CMA-ES to weight optimization. Since the adaptation of C allows efficient searching in the existence of correlation between parameters, we expect that CMA-NeuroES will show good performance on the optimization of the synaptic weights for our robot controller (Hansen, 2011) (Hansen and Ostemeier, 1997) (Floreano et al., 2008).

Every robot in our SRS has the same type of CMA-NeuroES controller. Each robot receives 16-input information from the environment. Swarm behavior fitness is calculated from a fitness table to evaluate a robotic swarm and update the mean value m, covariance matrix C and global step-size σ. CMA-NeuroES operates in five steps:

$Step$ 1: Set all synaptic neural network weights randomly at the initial generation. If it is not the first generation, create offspring from (1).

$Step$ 2: Start the simulation, then evaluate the fitness of λ offspring by using the fitness table.

$Step$ 3: Send fitness and μ parents to CMA-NeuroES to create new offspring, and update all synaptic weights.

Table 5.8: Evaluation of SRS behavior

f_1	Touching a food source	+0.0015 × [time steps]
f_2	A food source reaches the nest	+3000
f_3	A food source is moved toward the nest	$+1500 \times (1 - d_{rem}/d_{init})$
f_4	All foods reach the nest	+1.0 × [remaining time steps]

$Step$ 4: Choose synaptic weights with higher fitness as parents for the next generation.

$Step$ 5: Repeat Step 1 to start a new generation until the terminal condition is met.

5.3.4 Apply Incremental Evolution to CMA-NeuroES

In evolutionary robotics approaches, the situation where no initial search pressures exist can occur when solving highly complex tasks. The result of our experiment on CMA-NeuroES with conventional evolution for the cooperative food foraging problem shows that there are three runs wherein the SRS collected nothing. This situation, the bootstrap problem in ER, occurs when all of the individuals in the initial generation are scored with null fitness prohibiting the progress of evolution. Overcoming the bootstrap problem is one of the difficulties in the ER approach.

Incremental evolution is an approach for solving bootstrap problems in highly complex tasks with evolutionary approaches. Mouret and Doncieux (Mouret and Donciex, 2008) categorized incremental evolution into four main approaches: staged evolution, environmental complexification, behavioral decomposition, and fitness shaping.

Staged evolution is an approach in which an objective task is divided into ordered sub-tasks, with every sub-task having a corresponding fitness function. A navigation task performed with staged evolution was presented by Bajaj and Ang Jr. (Bajaj and Ang, 2000). A mobile robot was placed in a simple environment wherein only one obstacle existed. The fitness value was calculated using a straight navigation component and an avoiding obstacles component. At a later stage, the robot was placed in a more complex environment, in which closer walls and sharp turns had been added to the environment. The fitness value was calculated in the same manner as that in the first stage. In the third and final stages, the fitness value was calculated as the product of the value calculated in the previous stage and the wall-following factor. The final result was that the robot acquired the wall-following behavior.

Environmental complexification works on a fitness value calculation in which the task complexity can be continuously modified by operating on certain parameters. A typical example was presented by Gomez and Miikkulainen (Gomez and Miikkulainen, 1997) in 1997. The task was for a predator whose behavior was controlled by an evolving artificial neural network to capture a prey within a fixed number of time steps.

Behavioral decomposition is an approach in which the robot controller is divided into sub-controllers. Every robot controller is evolved separately to solve a sub-task. Nardi et al. (Nardi et al., 2006) evolved a position controller for an autonomous helicopter with three phases of incremental evolution. In the first phase, a simple yaw controller was evolved. In the next phase, the rest of the controller, comprising three modules, specifically, guidance, pitch, and role modules, was evolved independently. In the final phase, these modular controllers were simultaneously evolved to enable them to adapt to each other.

Fitness shaping uses a weighted sum of multiple evaluation criteria to create a fitness gradient for artificial evolution to follow. Nolfi and Parisi (Nolfi et al., 1995) evolved an autonomous robot that picks up objects. They used a fitness formula with five components, which correspond to the following scenarios: the robot is approaching the target object, the target object is in front of the robot, the robot tries to pick up the object, the robot has the object in its grasp, and the robot releases the object outside the area.

To improve the performance of the SRS in solving a complex cooperative food foraging problem, we proposed staged evolution with environmental complexification using the CMA-NeuroES approach. In our cooperative food foraging task, we assume that three basic behaviors, (1) food-exploration, (2) food-transportation, and (3) obstacle avoidance, are required to solve our problem. Therefore, three-stage incremental evolution was provided, as shown in Figure 6.6.

Sub-Task 1 is a very simple problem, in which all three food sources are placed in the field without any obstacles in the environment. Every food source in Sub-task 1 requires at least three robots to move it (The dynamical friction for every food sources is 14 power units). The expectation is that SRS will acquire the basic behavior of food-exploration and food-transportation to the nest, i.e., collect three food sources. When the SRS solved Sub-Task 1, Sub-Task 2, in which two obstacles are added to the field and the positions of food are changed is given to the SRS. The third basic behavior of obstacle avoidance will be acquired after Sub-Task 2. At that time, every food sources needs at least four robots to move it (We increased the dynamical friction to 19 power units).

Our simulation will then randomly add a source after the first food source is collected. When the SRS has solved Sub-Task 2, a final task, Goal Task, in which two new obstacles are placed into the field with a narrow path between them is posed. In that case, food sources are too large to be moved through the narrow path. This trap makes our cooperative food foraging task much more difficult. The SRS learns more advanced food-transportation through obstacle avoidance behavior. In Goal Task, new food sources are randomly created after each food source has been collected. Every food source in Goal Task requires at least five robots to move it. In our cooperative food foraging task, task-transitions to the next sub-task occur only when the SRS has solved the current sub-task continuously for ten generations. The number of generations for task-transition has been optimized in our preliminary experiment.

The performance of SRS was also evaluated using the four components shown in Table 5.8. In the case of Sub-Task 1, the fitness value f of the SRS is calculated as $f_1 + f_2 + f_3 + f_4$. In the case of Sub-Task 2, the fitness value f is calculated as $f_2 + f_3 + f_4$. The f_1 is omitted, because the SRS has already learned the aggregation behavior for food sources through Sub-Task 1. In the case of Goal Task, the simulation will run 5,000 time steps to see how many food sources the SRS can collected. The fitness value of Goal Task f is calculated as f_2+f_3, because the SRS has already learned to touch food sources, and food sources will be added to the field continually during the 5,000-time step simulation.

5.3.5 Results

The performance of SRS is depicted by four components shown in Table 5.8. The SRS collects 0.0015 at each robot and each time step when a robot touches one of the food sources. The sum of the points is set at the f_1 component. The SRS collects a bonus point each time the swarm successfully returns to the nest with a food source. The sum of the points is set at the f_2 component. However, since there can be cases wherein they cannot finish bringing the food source to the nest within the time limit, partial evaluation for moving a food source is considered. For each food source, the points awarded are calculated as 1,500 $\times$ (1 - d_{rem}/d_{init}) at the end of the run, where d_{rem} and d_{init}, the remaining distance to the nest and the initial distance from the nest, respectively, are produced as points. The sum of these points is set as the f_3 component. When all the food sources have been moved to the nest, the f_4 component is calculated as 1.0 $\times$ [remaining time steps] when the task is achieved.Otherwise, f_4 is evaluated as zero. The CMA-NeuroES parameter setting is as follows. The offspring λ are set at 100, and the initial SDis set at 0.2, with the initial covariance matrix $C= I$. The computer simulations' last generation is set at 500, and 10 independent experimental runs are conducted.

In our compearation computer simulation, (μ, λ)-FastES (FES) (Rechenberg, 1973) (Yao and Liu, 1997) and a real-coded GA (Eshleman and Schaer, 1993) (Holland, 1973) (Koza, 1992) (Koza, 1989) were also used to evolve the synaptic connection weights of the artificial neural network that generated the robots' actions. To make the experiment comparable, four approaches are proposed to solve the cooperative food foraging problem: CMA-NeuroES with conventional evolution, CMA-NeuroES, FES, and real-coded GA with incremental evolution. The parameter settings of the other evolutionary algorithms are as follows. The real-coded GA's population size is also set at 100. Tournament selection with size two and elite preservation with size one are adopted. The mutation rate is set at 1.0. This means that all the synaptic connections are mutated for each generation by adding Gaussian noise, whose mean and SDs are 0 and 0.05, respectively. No crossover was used. These parameter tunings had been performed in our preliminary experiments. All the last generation of the artificial evolution are set at 500, and ten independent experimental runs were conducted.

5.3.6 Robustness Test

The robustness of the best robot controllers with each approach was measured by conducting a breakdown test with the Goal Task (Figure 4 (c)). In this test, the robots with the best controllers of each approach were selected. The fact that an SRS can work dynamically as individual robots are deployed has the advantage that the failure of individual robots will

Table 5.9: average number of generations in which the swarm succeeded in solving sub-tasks for incremental evolution

		Average	SD
Sub-Task 1	FES	31.8	31.41
	Real Coded GA	41.7	15.2
	CMA-NeuroES	23.9	11.5
Sub-Task 2	FES	149.4	66.3
	Real Coded GA	136.6	46.8
	CMA-NeuroES	115.4	38.8

hardly affect the performance of an evolved SRS. In our test, the SRS continued its collective behavior for searching food sources and returning food sources to the nest, even after a few robots had stopped working.

Every robots is tested to determine whether it is broken at every time step. The breakdown coefficient (B_c) is calculated as follows:

$$B_c = \frac{S_r}{R_s N} \tag{5.2}$$

In this equation, S_r is the stop rate, the R_s are the random steps from 0 to 5,000, and N is the number of robots. The test system will decide if any robot is broken by comparing B_c with a uniformly distributed double value between 0.0 and 1.0 from a random number generator's through at every time step. If B_c is larger than the random number, then the robot will stop. All the broken robots remain in the field and can be detected by robot sensors. Stop counter S_c will count one after a robot is stopped to control the number of broken robots.

Our breakdown simulation runs 5,000 time steps for ten iterations. Moreover, we consider the limitation of the stop counter by 10, 20, 30, 50, meaning that 10, 20, 30, 50 robots in the SRS, respectively, will be stopped randomly during the simulation.

Figure 5.18 shows the result of the CMA-NeuroES controller for the cooperative food foraging problem. Our SRS successfully collected food sources in seven of ten runs and the maximum number of collected food sources was five. However, three runs performed very poorly; in them the SRS collected nothing at all.

Figure 6 shows the fitness transitions of the best individuals and the averages of each

Table 5.10: number of brought back food sources while the robots Breakdown

	Conventional Evolution				FES				Real Coded GA				CMA-NeuroES			
Breakdown	10	20	30	50	10	20	30	50	10	20	30	50	10	20	30	50
Average	2.3	1.0	1.3	1	3.5	2.8	3.3	2.8	4.4	4.2	3.9	3.7	5.1	4.5	4.9	4.3
SD	1.1	0.9	0.6	0.7	1.3	1.0	1.4	0.9	1.8	1.2	0.9	0.8	1.0	0.5	0.5	0.8

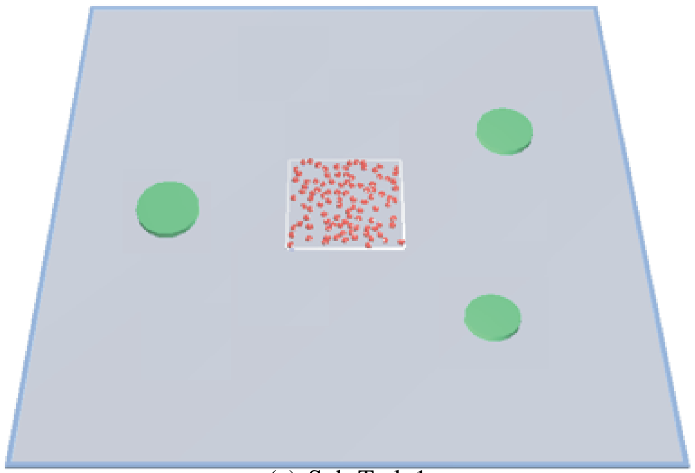
individual in the best run. Figure 5.19 shows that the incremental evolution approaches with evolutionary algorithms collected at least two food sources, indicating that they were successful in solving the Sub-Tasks for all the runs. The best run of CMA-NeuroES collected eight food sources in Goal Task, whereas conventional evolution had three runs that collected nothing. It is clear that not only the maximum fitness but also the average fitness of CMA-NeuroES with incremental learning is higher than those of others. Table. 5.9 shows the average number of generations and corresponding SDs required by the swarm to succeed in solving the sub-tasks. The incremental evolution approach with CMA-NeuroES required approximately 23 generations to solve Sub-Task 1 and approximately 115 additional generations to solve Sub-Task 2. Conversely, the FES and real-coded GA required approximately 31 generations and 41 generations to solve Sub-Task 1, respectively, and approximately 149 generations and 136 generations to solve Sub-Task 2, respectively. Therefore, incremental evolution with CMA-NeuroES exhibits better search ability to find better solutions.

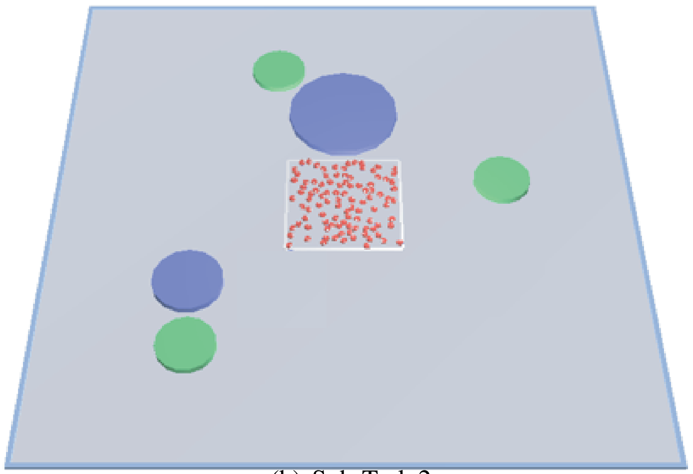
Table 5.10 shows the results of the average returned food source numbers, and the SD of four approaches for ten iterations. As a result, we see that conventional evolution performs poorly. When 50 robots stopped during our simulation, only one food source could be returned to the nest. In the incremental approach, the number of returned food sources of FES decreased to two, when the stopped robots number increased from 10 to 50. Real-coded GA shows its robustness, because the returned food source numbers decreased from four to three and the SD shows its stability. However, CMA-NeuroES with incremental evolution performed best overall. When ten robots in the SRS stopped, the SRS could still return at least five food sources to the nest. The robust of CMA-NeuroES enables it to return four food sources even when half of the robots in the SRS are stopped.

Typical behavior observed in robustness tests for an SRS with a CMA-NeuroES controller is shown in Figure 8, wherein 50 robots are stopped during the simulation. Some robots immediately find a food source (Marked 2) after leaving the nest (Figure 8(a)-8(b)). At the same time, some robots are stopped at the beginning of our robustness tests and becomes obstacles in the field. In Figure 8(c), another food source (Marked 1) is found by another group of robots, when the first group of robots is trying to return the food sources to the nest. The food source (Marked 1) is collected after our SRS successfully bypassed the narrow pass in the field (Figure 8(c)-8(e)) while two food sources (Marked 2, 3) are already near the nest. An additional food source (Marked 4) is added to the field as soon as the first food source is collected (Figure 8(f)). Nearly half of the robots are stopped at this moment, after the food sources (Marked 4 and 5) are collected ((Figure 8(h)-8(i))). At the end of the simulation, 50 robots in our SRS have been stopped. Two groups of robots are still trying to collect the food sources (Marked 7 and 8) as the simulation ends (Figure 8(j)).

5.4 Summary

In this chapter, We firstly has combined an artificial neural network approach with CMA-ES, called CMA-NeuroES, with a swarm robotics system through a cooperative transport problem. Results shows that the evolutionary robotic approach is successfully applied to cooperative transport problems by adapting CMA-NeuroES. Swarm behavior emerged as a result of CMA-NeuroES. Using a recurrent artificial neural network and CMA-ES to adapt weights led to better performance for the SRS. CMA-NeuroES evolves faster and has a higher success rate. Unlike some of the evolutionary algorithms, CMA-NeuroES does not require tedious parameter tuning such as DE. The contribution of this paper is (*i*) that we are the first one to apply CMA-NeuroES to swarm robotic systems and successfully generated cooperative behavior, and (*ii*) we compared the result with other evolution strategies and experimentally showed that CMA-NeuroES is a better approach when applied to SRSs. In addition, we successfully applied the ER approach to a specific complex cooperative food foraging problem with a large SRS including one hundred homogenous robots. Swarm behavior emerged as a result of CMA-NeuroES with incremental evolution. The incremental evolution approach of staged evolution and environmental complexification helps ER avoid the bootstrap problem. The result of incremental evolution outperforms the conventional evolution approach for cooperative food foraging. In addition, a robustness test confirmed that the incremental evolution approach with CMA-NeuroES is robust, because it can solve the same cooperative food foraging problem, even when half of the robots are stopped. We expect that our proposed method and robustness test will also hold for other SRS benchmarks.


(a) Sub-Task 1


(b) Sub-Task 2

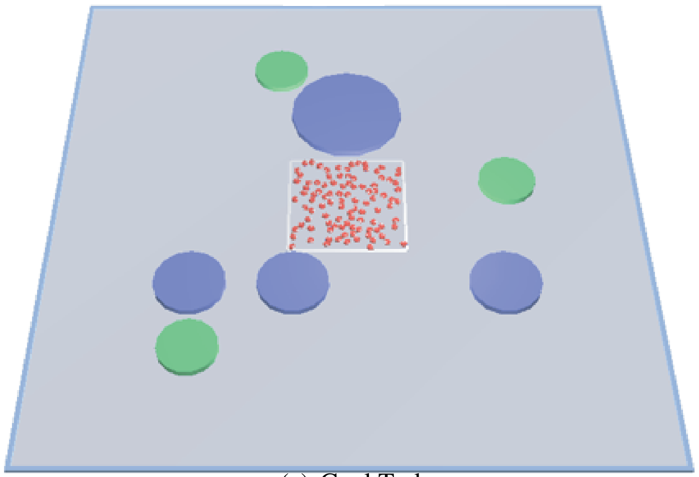

(c) Goal Task

Figure 5.17: Three-stage incremental evolution for CFFT

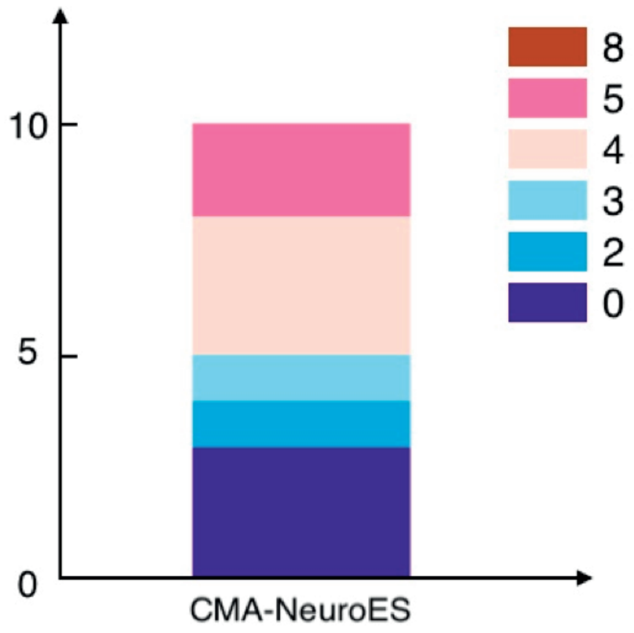


Figure 5.18: Food sources that SRS collected

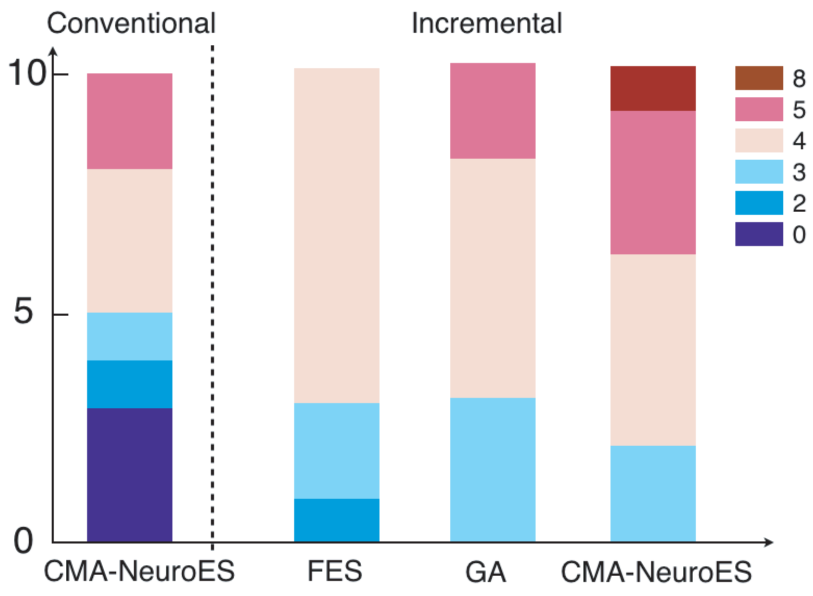
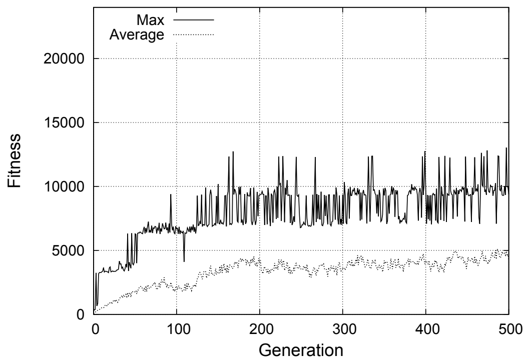
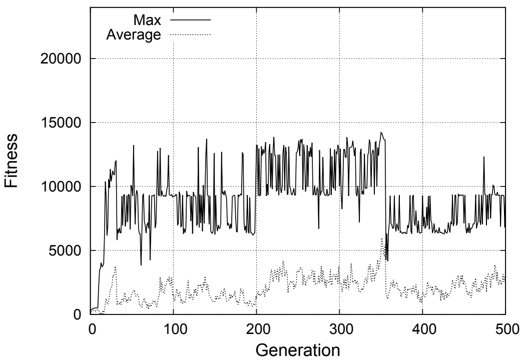


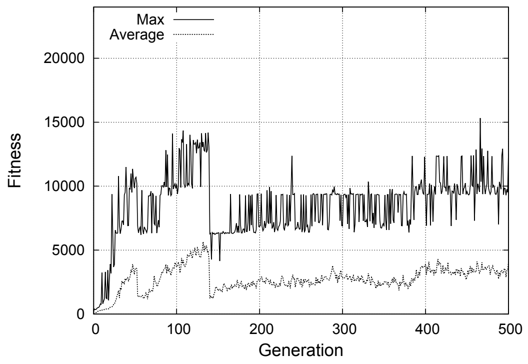
Figure 5.19: Food sources that SRS collected



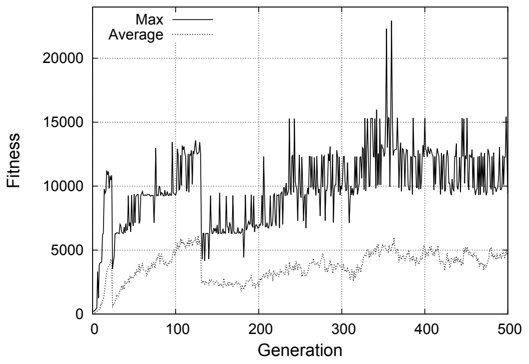
(a) Conventional evolution with CMA-NeuroES



(b) Incremental evolution with FES

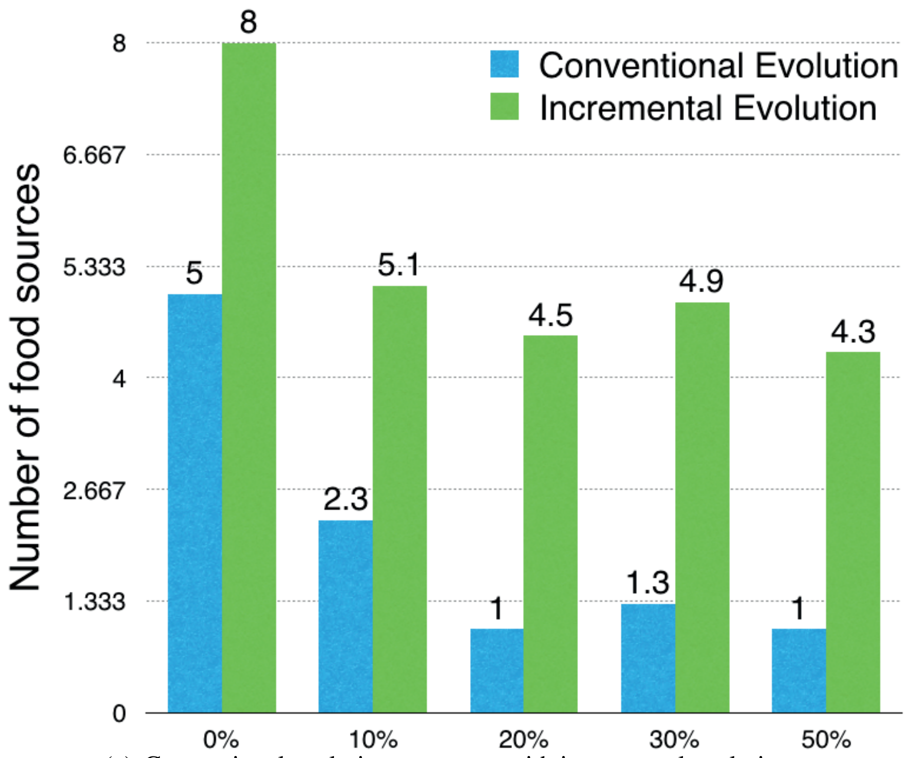


(c) Incremental evolution with Real coded GA

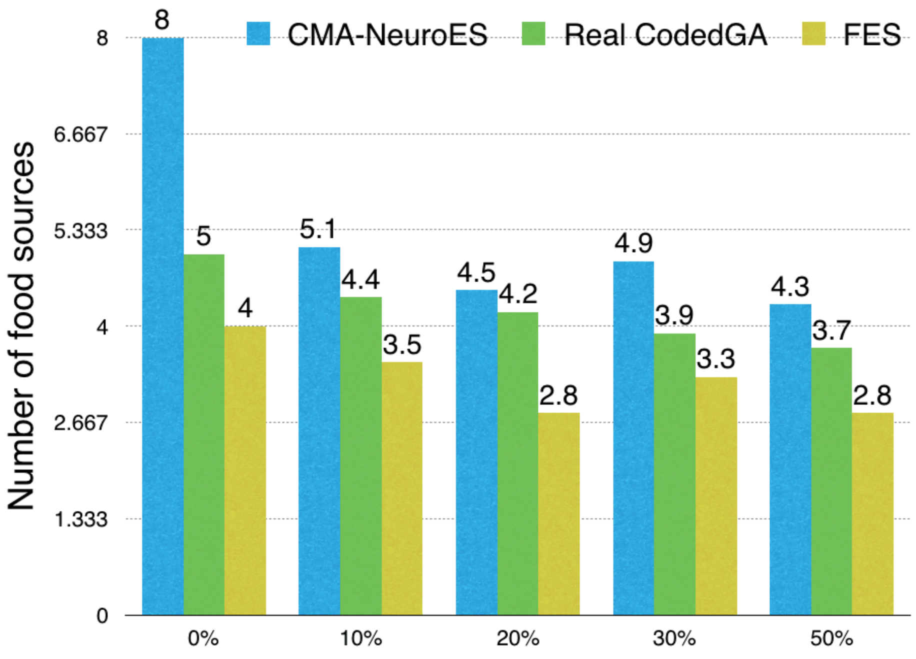


(d) Incremental evolution with CMA-NeuroES

Figure 5.20: Fitness transitions of the best(solid line) and the average (dotted line) for each controller's best performed run

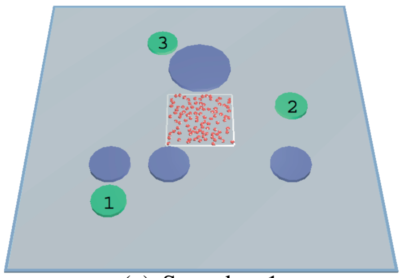


(a) Conventional evolution compares with incremental evolution

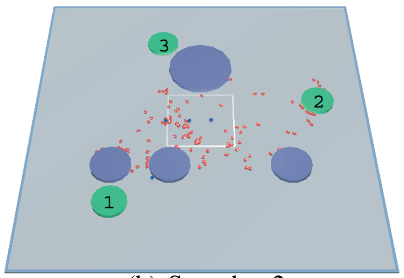


(b) CMA-NeuroES compares with other algorithms using incremental evolution

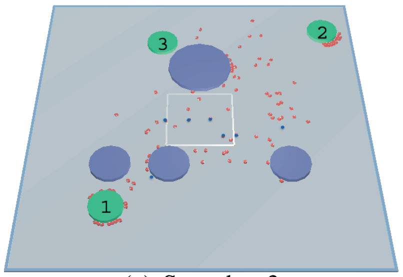
Figure 5.21: The results of breakdown test



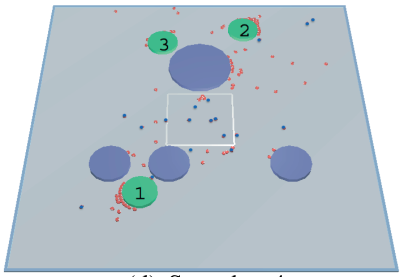
(a) Snapshot 1



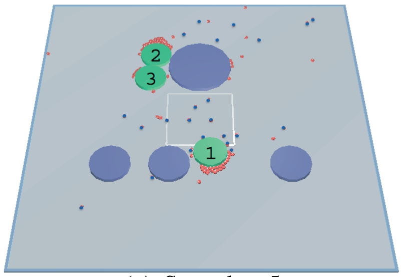
(b) Snapshot 2



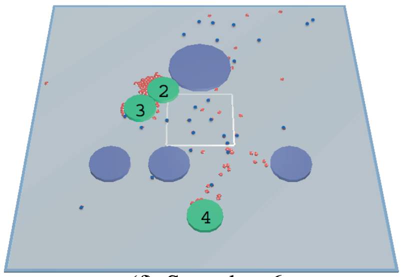
(c) Snapshot 3



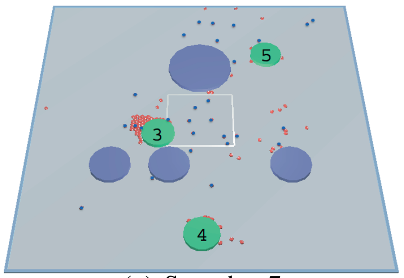
(d) Snapshot 4



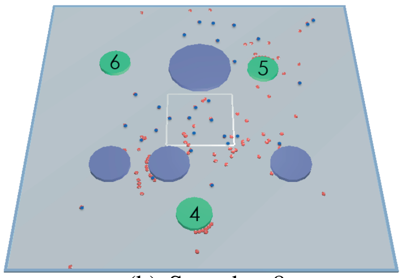
(e) Snapshot 5



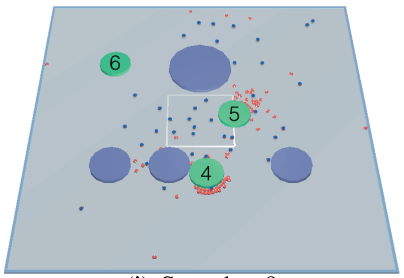
(f) Snapshot 6



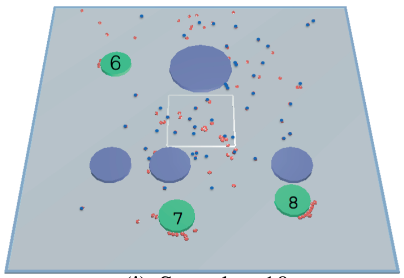
(g) Snapshot 7



(h) Snapshot 8



(i) Snapshot 9



(j) Snapshot 10

Figure 5.22: Snapshots of the best cooperative collective behavior found by CMA-NeuroES with incremental evolution approach of breakdown rate 50%

Chapter 6

Understanding Autonomous Task Allocation by a Clustering Approach

6.1 Introduction

Swarm robotics(SR) (Şahin and Winfield, 2008) (Şahin, 2005) is a novel approach inspired by social animals, such as ants, wasps and birds. These examples of social animals show that simple individuals can successfully accomplish difficult tasks when they coordinate as a system. This kind of system-level behavior appear to be robust, scalable and flexible in which impressed researchers from robotics areas. To take advantage of these social animals, a new research field of robotics is known as swarm robotic systems (SRS). Researchers expected the SRS to accomplish tasks beyond the capabilities of a single robot.

The design methodology of SRS can be divided into two categories: (*i*) *Behavior based design* is the most commonly used design method that the individual behaviors of the robots are designed by hand. Researchers in this field proposed many different approaches to control the SRS. Kube and Zhang (Kube and Zhang, 1997) in task-modeling uses a design method of a robot controller using a finite state machine carefully designed by a human programmer without a global controller. They demonstrated its effectiveness in box-pushing problems, which the collective behavior is obtained after individual behaviors iteratively adjust and tuned. (*ii*) *Automatic design methods* can be divided into two main methods:

reinforcement learning and evolutionary robotics (ER), which ER is the application of artificial evolution to robotic systems with a sensory-motor interface to the environment, i.e., evolving a robot controller that represented as an artificial neural network.

The collective behaviors of SRS are basic behaviors that could be combined to take over complex real-world applications. These collective behaviors can be classified into four main categories: *spatially-organizing behaviors* focus on how to organize and distribute robots and objects in space, *navigation behaviors* focus on how to organize and coordinate the movements of a swarm of robots, *collective decision making behaviors* focus on letting a group of robots agree on a common decision or allocate among different parallel task, *other collective behaviors* are behaviors that no belong to the categories above.

If a SRS have the ability of coordinate these collective behaviors, the most complex real-world applications such as food foraging problem and construction problem can be solved. In food foraging problems, the SRS requires the self-organize behavior to cooperate and move heavy food sources by organized groups. The SRS also need navigation behavior to search for food source and find the way back to the nest. Last but not least, the decision-making behaviors allow different groups of robots to do different works at the same time.

As another important research direction of SRS, only few researchers draw attention to the analyze method for SRS. Our research group advocated the analyze of SRS's collective behavior will help us understand deeper details of the swarm robotics system.

As we know interdependent subgroups with different collective behaviors can be found in social insects, which are known to have the ability of dividing complex problems into subtasks. In that case, SRS is expected to perform task allocation through different tasks and it can be detected through behavior analysis for researchers to grasp the characteristics of collective behavior of SRS.

Given that biological swarms allocate tasks appropriate to situations, we have developed a technique for extracting autonomous specialization developed in a robotic swarm. This technique identifies subgroups of robots that perform the same role in a given situation. Because of the interaction of robots, the swarm robotics system can be considered as a network, furthermore, we divided SRS into subgroups by using a clustering method. However, there are no researches mentioned about the duration of SRS's behavior. In order to grasp much more details of SRS's behavior, we import a definition of duration times for

SRS's behavior, which helps us to know the relationship between behavior's duration time and our food foraging task.

In this study, we propose a method for analyzing duration time in SRS's subgroups for a food foraging problem. Realizing that swarm robotics system is mainly inspired by biological swarming behavior, we believe the times of behavior's duration will help us to know more about the behavioral sequence in the SRS from another point of view.

The remainder of this paper is organized as follows. Section 2 introduces the task and our proposed evolution method for generating collective behavior. Section 3 describes the clustering method and analyzes the duration time of behavioral sequence-based approach. The results and discussion of our proposed analyze method are presented in Section 4. Section 5 is the conclusion of our work and the future works are also suggested for analyzing the collective behavior of SRS.

The cooperative food foraging problem was inspired by the behavior of social animals searching for food sources distributed around their nest. As one of the most complex real-world applications for a robotic swarm, the challenge is to find better search strategies that maximize the ratio of bring food to the nest. Fig. 6.1 deals with the food foraging problem in this thesis. The field is a 5000×5000 unite length square. The nest, which is a 1000×1000 square goal area, is located at the center of the field. 100 autonomous mobile robots are randomly placed in the nest at the initial condition. Three food sources "F" are randomly placed in the field. In our simulation experiment, we set 24 dynamical friction for every food sources that requires at least 5 robots to move it cooperatively in a certain direction (The maximum pushing power of a robot is 5, based on the physical settings of our simulator). Soon after a food source is collected, it disappears and a new food source is placed on the field randomly during 5000 time steps. This means the SRS should maximize the ratio of bringing food sources to the nest, which means food sources should be collected as many as possible.

All robots in the SRS are assumed to have the same specifications as shown in fig. 6.2. Each robot is 50 units in diameter and has two types of sensors: eight IR sensors and an omni-direction camera. The eight IR sensors are arranged around each robot. Each IR sensor gives a value that is inversely proportional to the distance to an object, which might be a food source, an obstacle, the wall, or other robots within the sensor range of 64 units.

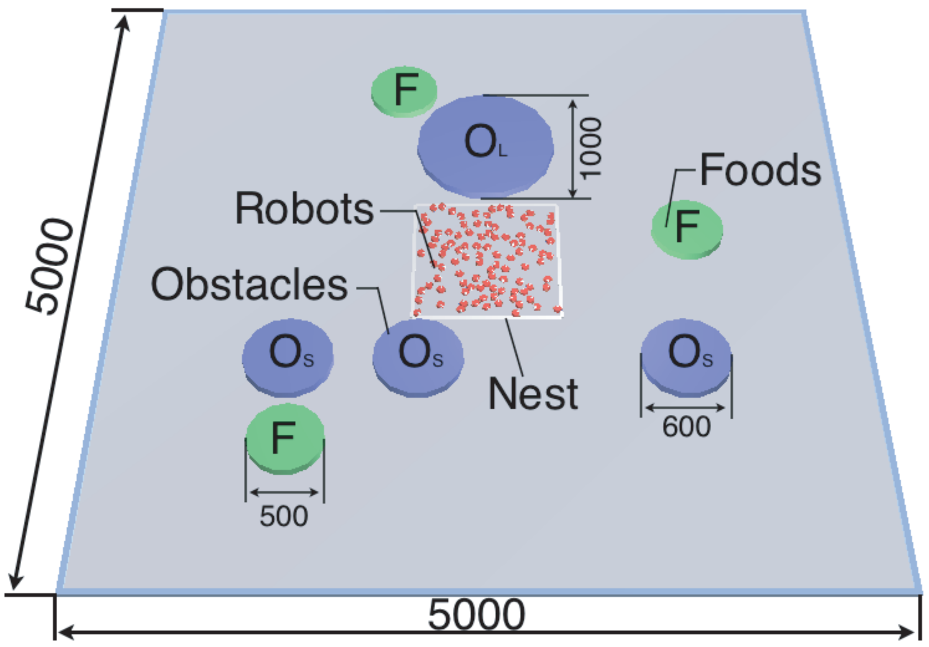


Figure 6.1: The cooperative food foraging problem

The values are normalized between zero and one. The omni-vision camera are located at the center of each robot.

The robot's sensor abilities are summarized as follows:

Information obtained by two types of sensors forms an input layer of a robot controller composed of 24 inputs connected to a motor on the right and an output layer composed of two neurons connected to the motor on the left. In addition, each input neuron receive the Gaussian noise, whose mean and standard deviation are 0 and 0.03, respectively. Four fully inter-connected hidden layers are adopted from our preliminary experiments for computer simulation. The neurons of recurrent artificial neural networks (RANN) are connected as shown in fig. 6.3 as our previous work (Yu et al., 2013). Therefore, the number of synaptic connections is 156. All robots are assumed to have the same RANN controller.

All computer simulations were conducted under the following conditions. The population size was set at 100. A size-two tournament selection and size-one elite preservation were adopted. The mutation rate was set at 1.0. In each generation, the synaptic connections were mutated at the specified rate and Gaussian noise (of mean and standard deviation

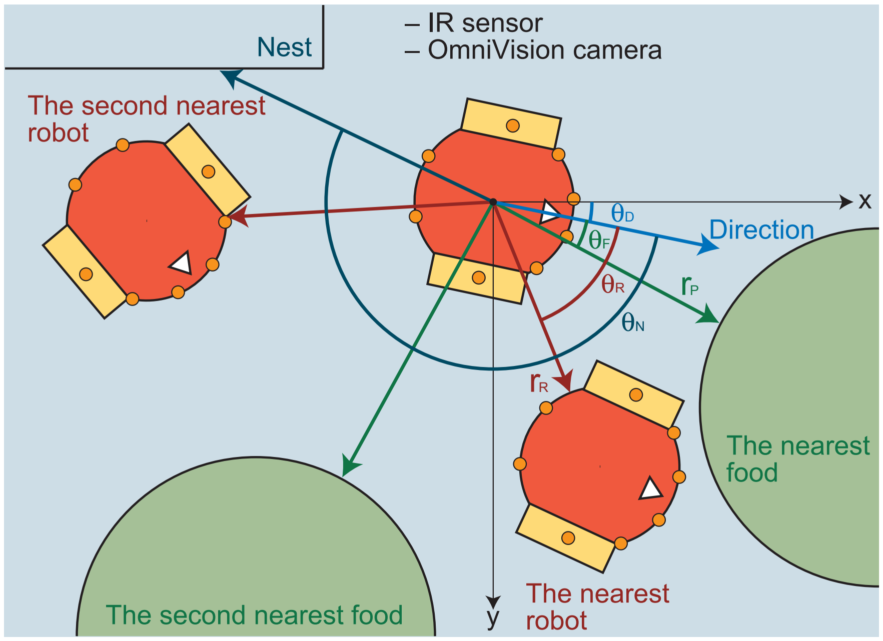


Figure 6.2: Robot information

0 and 0.05, respectively). The system was evolved through 300 generations. The evolution of the fittest individual, identified from the best trial, is shown in Fig.

6.2 Analysis Method

6.2.1 Extracting Subgroups

From the viewpoint that task allocation is a key consideration in swarm robotic systems, we proposed a method for dividing a SRS into subgroups, each of which plays different roles (Ohkura et al., 2011). In our previous work, a robotic swarm is mapped onto a complex network, where robots are represented as nodes, and informational connections between them are represented as directed links. Using this network representation, various tools from the field of complex networks can be applied to extract community structures.

In order to evaluate the quality of the community structure, modularity measure (Newman, 2003) is introduced. Subgroups with the maximum likelihood can be extracted according to the largest modularity (Newman, 2004).

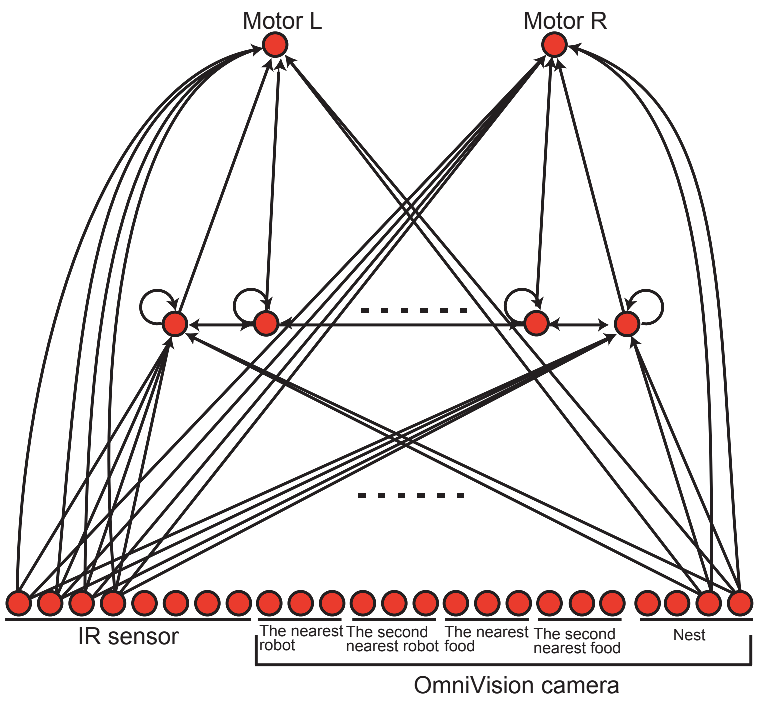


Figure 6.3: Artificial neural networks for robot controller

in this thesis, an approximation method called CNM method, which the operation of the algorithm involves finding the changes in Q that would result from the amalgamation of each pair of communities, choosing the largest of them, and performing the corresponding amalgamation. This method result in a considerable saving of calculation time.

1. Calculate the modularity assuming that the whole network is a single subgroup
2. Calculate the initial values of changed modularity matrix and and populate the max-heap with the largest element of each row of the matrix
3. Select the largest changed modularity, join the corresponding communities, update the matrix of changed modularity, the heap and increment the matrix of Modularity
4. Repeat step 2 until only one community remains.

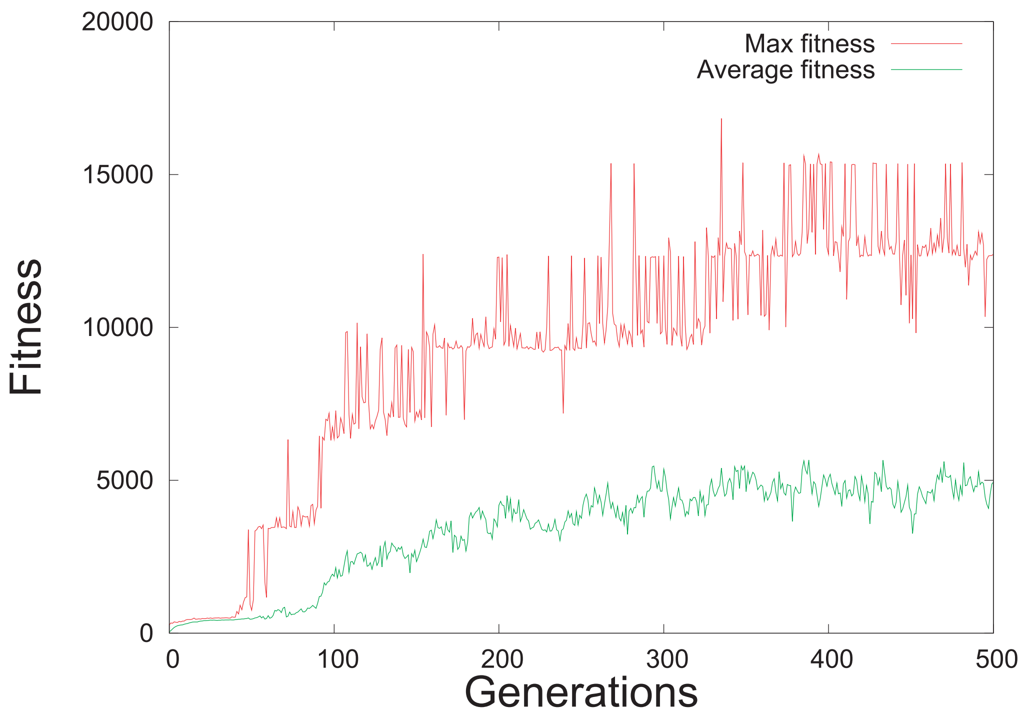


Figure 6.4: Fitness transition

This procedure requires a shorter calculation time than the GN method that allowed us to analyze a large SRS over 100 robots.

6.2.2 Analysis Based on Time Duration

Our proposed time duration based method focus on the duration that a subgroup continues to take the same actions. In another word, it shows how long does it take for a subgroup to change its behavior. In our food foraging task, for example, subgroup will change its behavior from searching food source to touching a food source after the robots find a food source.

In time duration based analysis, the expected value is calculated as we assumed that the time duration of behavior's frequencies are random. This kind of assumption can be represented by a negative exponential distribution:

$$y = 1 \quad e^{\;bx}$$

Here b is the probability of changing into another collective behaviors in every unit time. x stands for the duration time of behavior's frequencies' frequency y. In this thesis, we set 10 time step as a unit time.

6.2.3 Test Method of Contingency Table

An analysis method using a contingency table discloses the significance of behavioral sequences. This method analyses residual differences between expected and observed behavior's frequencies in order to locate sources of non-random association between variables (Fagen and Mankovich, 1980). In general χ^2 test is used for contingency table analysis. The χ^2 goodness of fit test is able to compare observed transitions with expected ones. Hence transition frequencies are examined in order to test particular hypotheses about transition probabilities. The standard analysis procedure is as follows:

In this thesis, χ^2 test is adopted as a fundamental approach. The standard analysis procedure is as follows:

1. Calculate about each cell of the transition matrix.

2. Conduct χ^2 test in one degree of freedom. Here, χ^2 is calculated as,

$$\chi^2 = \frac{(e_{ij} \quad f_{ij})^2}{e_{ij}},$$

   where f_{ij} is the number of observations and e_{ij} is the expected value in the i, jth cell of a behavioral contingency table.

If a χ^2 value is greater than its corresponding rejection region (χ^2_a=3.84), the behavior sequence is said to be observed significantly. In this method, the contingency table is necessary to postulate whether the expected value are large. For this reason it is recommended that expectations should not be smaller than 5 (Ohkura et al., 2011).

6.2.4 Behavior Category

Set up the repertoires that represent the roles of subgroups. These repertoires are defined based on observations as in ethology. Behaviors of the robots within the subgroup are classified into repertoire from output of these robots. Classification criteria of each repertoire is defined as follows:

Pf . Food source pushing
Action to push the package. Robots transport the package toward desired direction in a coordinated manner. The power of subgroup is the sum of the force vectors of the robots in the subgroup. The classification criteria for this action is whether robot in the subgroup is touching a package, and angular difference between direction of the power of subgroup and transport direction of the package is under $\pm 60°$.

Ap . Assist pushing
Action to push the package, however the classification criteria is whether the robots in a subgroup is touching a package, and angular difference between direction of the power of subgroup and transport direction of the package is over $\pm 60°$.

Tf . Food source touching
Action only in contact with the package. Since the number of robots needed to push the package is not enough or robot push in a direction away from target, Robots cannot transport the package. The classification criteria is whether the robots in a subgroup is touching a package, and angular difference between direction of the power of subgroup and transport direction of the package is over $\pm 90°$. The package is not moved.

To . Obstacle touching
Action to hit with the obstacle. If the robots in a subgroup is in contact with the obstacle, the action is categorized into this.

Sf . Food searching
Action to search in the field. Actions except the previous five is categorized in this case.

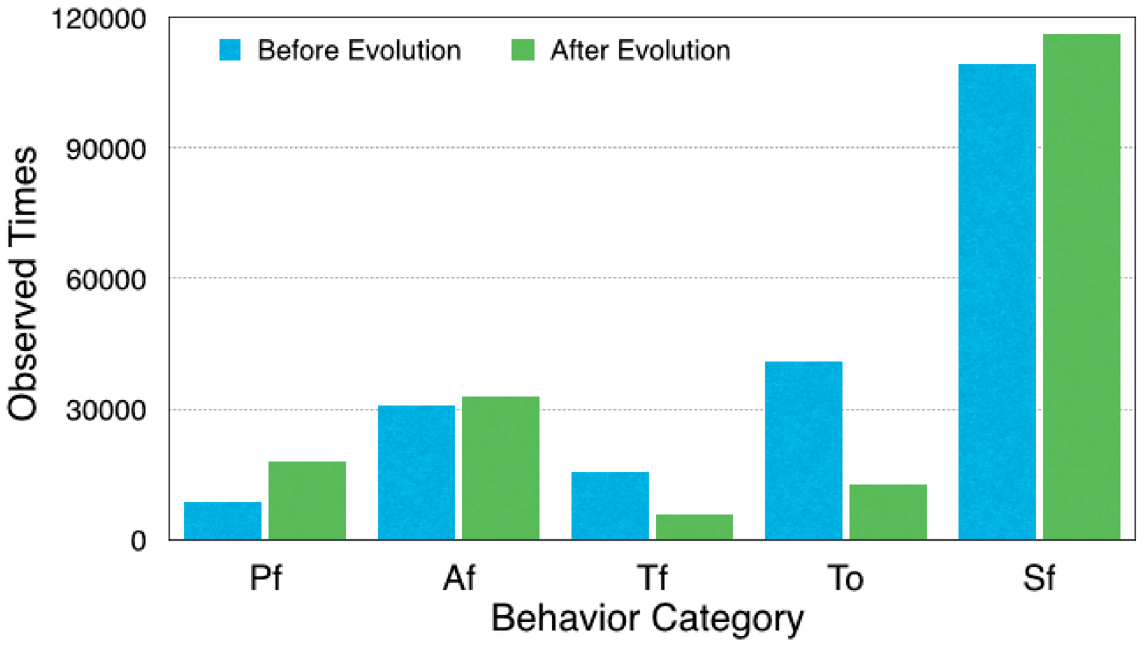


Figure 6.5: Observed times before and after evolution

6.3 Results

Fig.5 shows the results of observed times before and after evolution. In the early stage of evolution, we observe food searching behavior happens the most and the SRS touches obstacle frequently. At that time, the behavior of food source pushing hardly happens. After evolution, the duration time of food source pushing behavior is double that of early stage. On the other hand, the duration time of food source touching and obstacle touching decreased with the evolution of SRS. The duration time of assisting pushing behavior and food searching behavior increase only a little.

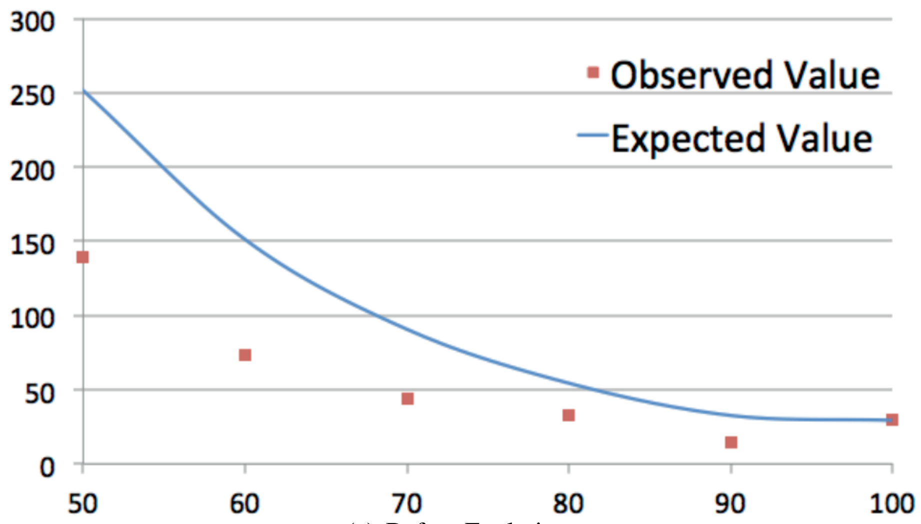
Fig.6 is the result of duration time's χ^2 test. In this figure, the curve is the expectation value and the small squares stand for actual value. After evolution, duration over 100 time steps' behavior increases rapidly as we expected.

Table 1 shows the duration catalogue of SRS at an early stage and Table 2 shows the duration catalogue of SRS at a late stage. First, expected value are calculated by observed value matrices with χ^2 values.

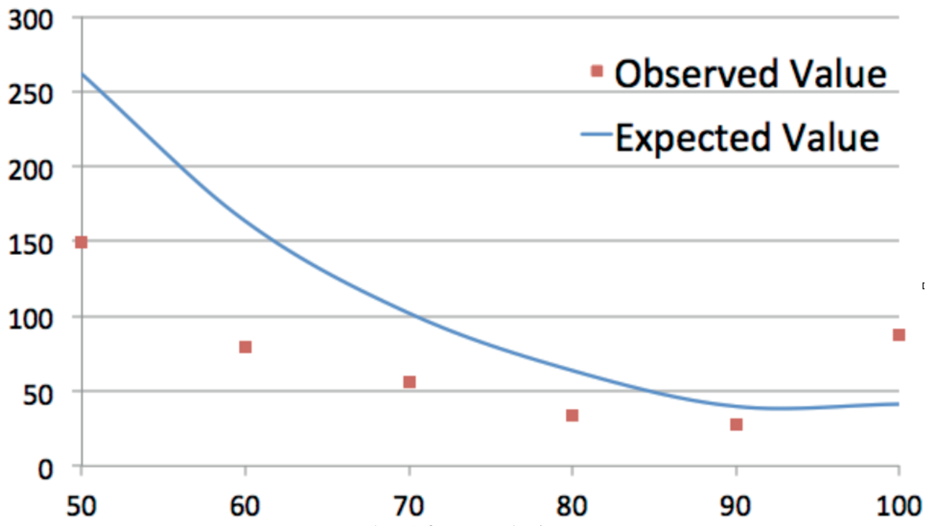
Compared with the early stage, the SRS increased the duration time for $Food\ source$ *pushing* and decreased the duration time for *Obstacle touching*. This appearance is understandable because in order to collect food sources, the SRS should avoid touching obstacles. The $Assist$ *pushing* does not changed very much because there are 100 robots in our swarm robotics system and there will always have some robots surrounding the food source trying to help foraging process. The duration time of $Food$ *source touching* behavior decreased because robots at the early stage can only touch the food sources with out any purposes. After evolution, for the lack of food sources, SRS take most of its time for $food$ *searching*, pushing food sources cooperatively to the same direction lead to the result of fast and efficient foraging. At last, our analyze method shows the SRS will successfully achieved food foraging problem with the increasing of duration over 100 time steps' behaviors.

Fig. 6.7 is snapshots of SRS's behavior after artificial evolution. Several subgroups are extracted in each time step with different colors and SRS performs successful cooperative food foraging. All the robots are placed randomly in the nest at first, as shown in Fig. 6.7(a). The first three food sources are found by small subgroups of robots and a large subgroup are trying to collect a food source from the beginning of simulation (Fig. 6.7(b)).

As soon as a food source is collected, a new food source appears randomly in the field. Several subgroups of robots are trying to collect food sources while some subgroups are moving around the field searching for food sources (Fig. 6.7(c-d)). In Fig. 7(e), robots are cooperating to move a food source, however, at the same time, another food source get stuck in the narrow pass. A small subgroup with 3 robots are trying to move a food source. It is impossible because every food source requires at least 5 robots to move it (Fig. 6.7(f)). In Fig.6.7(g)-(h), over half of the robots are moving along the field searching for food sources. Finally, the subgroups collected another two food sources and more subgroups of robots are working on the food source stuck in the narrow pass.



(a) Before Evolution

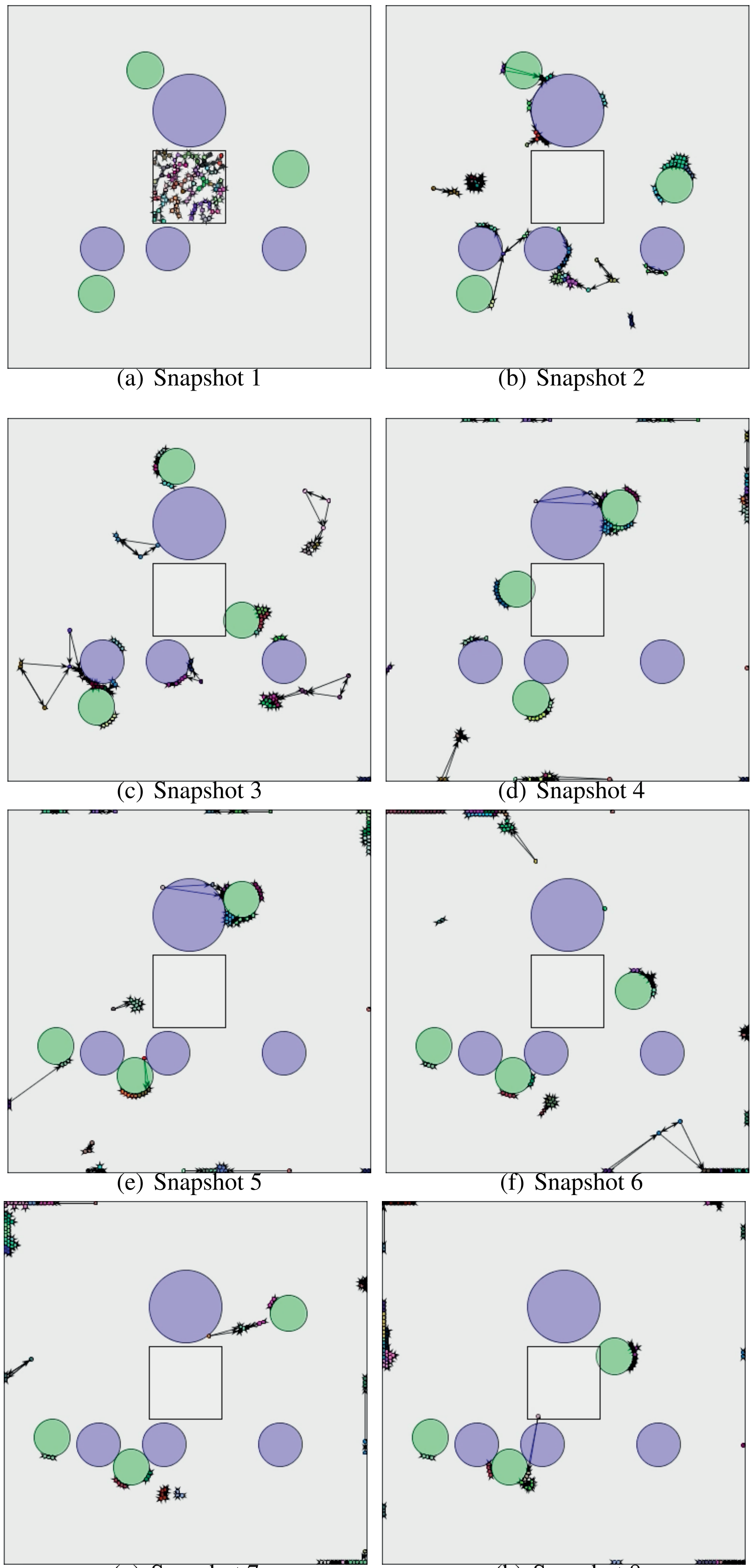


(b) After Evolution

Figure 6.6: The result of duration time's χ^2 test

6.4 Summary

We proposed a novel method for analyzing the collective behavior of SRS by using the duration time of subgroups' behavior. Firstly, we extracted the subgroups by utilizing a clustering method based on the theory of complex networks. Behavior of the subgroups are classified based on observation. Then, we extracted the feature of behavioral sequence and applied the proposed method to a food foraging problem involving 100 autonomous mobile robots. The duration based behavior analyze approach helps researchers to grasp the relationship between behavior sequence and duration time, which make it clearly to observe behavior generation process from another view point.


(a) Snapshot 1 (b) Snapshot 2

(c) Snapshot 3 (d) Snapshot 4

(e) Snapshot 5 (f) Snapshot 6

(g) Snapshot 7 (h) Snapshot 8

Figure 6.7: Behavior of SRS after artificial evolution in cooperative food foraging problem

Chapter 7

Conclusion

Swarm robotics is a novel approach to the coordination of large numbers of robots and has emerged as the application of swarm intelligence to multi-robot systems. Different from other swarm intelligence studies, swarm robotics puts emphases on the physical embodiment of individuals and realistic interactions among the individuals and between the individuals and the environment. In this thesis, we proposed, i.e., a method that a robot controller is designed by evolving artificial neural networks, is adopted.

Firstly, we point out that the definition of swarm intelligence was extended in 2004 and there is almost no relationship between the cellular robotic systems with SRS. After that, among many evolutionary algorithms to evolving artificial neural networks in my experiment, four evolutionary algorithms, CMA-ES, FES, FESplus and DE, are adopted for computer simulations. The CMA-ES is known as an efficient evolutionary algorithm for many test functions and optimization problems in continues domain and we were anticipate its performance. As benchmark of swarm robotic systems, a cooperative package pushing problems and a simple food foraging problems are conducted to examine the performance of CMA-NeuroES with other three evolutionary robotics approaches.

Results have shown that the evolutionary robotic approach is successfully applied to cooperative transport problems by adapting CMA-NeuroES. Swarm behavior emerged as a result of CMA-NeuroES. Using a recurrent artificial neural network and CMA-ES to adapt weights led to better performance for the SRS. CMA-NeuroES evolves faster and has a higher success rate. Unlike some of the evolutionary algorithms, CMA-NeuroES does not require tedious parameter tuning such as DE.

Secondly, we successfully applied the ER approach to a specific complex cooperative food foraging problem with a large SRS including one hundred homogenous robots. Swarm behavior emerged as a result of CMA-NeuroES with incremental evolution. The incremental evolution approach of staged evolution and environmental complexification helps ER avoid the bootstrap problem, which in highly complex tasks there are sometimes no initial search pressures exist. The result of incremental evolution outperforms the conventional evolution approach for cooperative food foraging. In addition, a robustness test confirmed that the incremental evolution approach with CMA-NeuroES is robust, because it can solve the same cooperative food foraging problem, even when half of the robots are stopped. We expect that our proposed method and robustness test will also hold for other SRS benchmarks.

Thirdly, we proposed a novel method for analyzing the collective behavior of SRS by using the duration time of subgroups' behavior. We extracted the subgroups by utilizing a clustering method based on the theory of complex networks and behavior of the subgroups are classified based on observation. In another word, we extracted the feature of behavioral sequence and applied the proposed method to a food foraging problem involving 100 autonomous mobile robots. The duration based behavior analyze approach helps researchers to grasp the relationship between behavior sequence and duration time, which make it clearly to observe behavior generation process from another view point.

The contribution of this paper is (*i*) that we are the first one to apply CMA-NeuroES to swarm robotic systems and successfully generated cooperative behavior, and (*ii*) we compared the result with other evolution strategies and experimentally showed that CMA-NeuroES is a better approach when applied to SRSs. (*iii*) we proposed a analysis approach based on complex networks for us to understanding the task allocation in a cooperative food foraging problem.

About the future work, CMA-NeuroES may face some limitation with real-world applications, we should examine it through other benchmark problems. For a desired problem to be solved, and a proposed behavioral design at the individual level, researchers must obtain guarantees for system-level performance. We would also like to try to apply CMA-NeuroES to a topology and weight evolving artitificial neural network approach. Since the appropriate topology for a given problem is not apparent in most SRS benchmark problems, the capability of a topology search for the ANN controller is still needed.

In addition, we will also focus primarily on the analysis of SRSs (Everitt, 1993) (Newman, 2003) (Newman, 2004). As an SRS can be considered as a large network with interactions among robots, we investigated finding subgroups in a robotic swarm (Ohkura et al., 2011) using the technology of temporary networks. The next step will be to describe how subgroups in an SRS develop and change in a large robotic swarm using a duration table. That will enable researchers to understand the detail of task allocation in an SRS from a macroscopic viewpoint.

Bibliography

R.R. Murphy, INTRODUCTION to AI ROBOTICS, Massachusetts Institute of Technology, pp. 3-4, 2000.

E. Şahin, A. Winfield. Special issue on swarm robotics, Swarm Intellgence, 2: pp.69-72, 2008

E. Şahin, Swarm Robotics: From Sources of Inspiration to Domains of Application, Swarm Robotics WS2004, LNCS 3342, pp.10-20, 2005.

E. Şahin, S. Girgin, L. Bayindir and A. E. Turgut. Swarm Robotics, SwarmIntelligence, -introduction and Applications-, Berlin, Heidelberg, Springer Verlag, pp.87-100, 2008.

O. Soysal and E. Şahin, A macroscopic model for self-organized aggregation in swarm robotic systems, In E. Sahin, W.M. Spears, and Alan F. T. Winfield, editors, Second International Workshop on Swarm Robotics at SAB 2006, Vol. 4433, pages 27-42, Berlin, Germany, 2006.

M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence, Volume 7, Issue 1, pp.1-41, 2013.

I. Harvey, P. Husbands, D. Cliff, Issues in Evolutionary Robotics, Journal of Adaptive Behavior, Vol. 2, pp.73-110, 1993.

Meyer, P. J. Husbands, P. Harvey, I., Evolutionary Robotics: A Survey of Applications and Problems, Proc. of the 1st European Workshop Evolutionary Robotics, pp. 1-21, 1998.

C. R. Kube and H. Zhang, Task Modelling in Collective Robotics, Autonomous Robots, Vol. 4 No. 1, Kluwer Academic, pp.53-72, 1997.

M. Dorigo, E. Tuci, R. Gro, V. Trianni, TH. Labella, S. Nouyan, C. Ampatzis, JL. Deneubourg, G. Baldassarre, S. Nolfi, et al, The swarm-bots project. In: Swarm Robotics. Springer, pp. 31-44, 2005.

M. Dorigo, D. Floreano, LM. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, et al, Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. Robotics Automation Magazine, 20(4), pp. 60-71, 2013.

Martinoli, A., Modeling swarm robotic systems: A case study in collaborative distributed manipulation, Proceedings of the Seventh International Symposium on Distributed Robotics Autonomous Systems, 23(4-5), pp. 415-436, 2004.

I. Harvey et al., Evolutionary Robotics: the Sussex Approach, Robotics and Autonomous Systems, Vol.20, pp. 205-224, 1997.

Yim, M., Zhang, Y., Lamping, J., Mao, E., Distributed control for 3D metamorphosis , Autonomous Robots 10, pp.41-56, 2001.

W. Liu and A. Winfield. Modeling and optimization of adaptive foraging in swarm robotic systems, International Journal of Robotics Research, 29(14): pp.1743-1760, 2010.

R. Guimera and L. A. N. Amaral, Functional cartography of complex metabolic networks. Nature 433, 895-900, 2005.

L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas, Comparing community structure identification. J. Stat. Mech. p. P09008, 2005.

X. Yao and Y. Liu, Fast Evolution Strategies, Control and Cybernetics, 26(3), pp.467-496, 1997.

H. Moriguchi and S. Honiden, CMA-TWEANN: Efficient Optimization of Neural Networks via Self-adaptation and Seamless Augmentation, Genetic and Evolutionary Computation Conference (GECCO'12), pp.903-910, 2012.

N. Hansen, The CMA Evolution Strategy: A Tutorial, Towards a new evolutionary computation Advances in estimation of distribution algorithms, Springer, pp.75-102, 2011.

R. Storn and K. Price, Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Space, *Morgan Kaufmann*, Journal of Global Optimization 11, pp.341-359, 1997.

M. Dorigo, L. M. Gambardella, *Ant colonies for the traveling salesman problem*, BioSystems, vol. 43, pp. 73-81, 1997.

T. Fukuda, S. Nakagawa, *Approach to the Dynamically Reconfigurable Robotic System*, Journal of Intelligent and Robotic Systems, Vol. 1, No. 1, pp. 55-72, 1989.

J. Kennedy, R. C. Eberhart, *Particle Swarm Optimization*, In proceedings of the IEEE International Joint Conference on Neural Networks, pp. 1942-1948, 1995

N. Hansen et al, On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation, *Morgan Kaufmann*, L. Eshelman (Ed.), pp.57-64, 1995.

C.R.Kube and H.Zhang,Task Modeling in Collective Robotics, Autonomous Robots, Vol.4, No.1, pp. 53-72, Kluwer Academic, 1997.

M. J. Mataric, Learning Social Behavior, Robotics and Autonomous Systems, Vol.20, pp. 191-204, 1997.

A. Martinoli, Proceedings of the Seventh International Symposium on Distributed Robotics Autonomous Systems, DARS' 04, Toulouse, France, 2004.

R. Arkin, G. Bekey, *Robotcolonies*-editorial, Autonomous Robots 4, 1997.

G. Theraulaz, E. Bonabeau, *Coordination in distributed building*, Science, 269(4): 686-688, 1995.

N. Hansen and A. Ostemeier, Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, *IEEE International Conference on Evolutionary Computation*, pp.312-317, 1996.

G. Beni, J. Wang, *Swarm Intelligence in Cellular Robotic Systems*, Proceedings of NATO Advanced Workshop on Robots and Biological Systems Vol 102, 1989.

N. Hansen and A. Ostemeier, Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The (μ/μ_I,λ)-ES, EUFIT'97, pp.650-654, 1997.

I. Akyildiz, W. Su, Y Sankarasubramaniam, Cyanic: A survey on sensor networks. IEEE Communications Magazine, pp.102-115, 2002.

G. Chirikjian, Kinematics of a metamorphic robotic system, IEEE International Conference on Robotics and Automation, pp.449-455, 1994.

D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, Pheromone robotics, In E. Sahin and William Spears, editors, Swarm Robotics Workshop: State-of-the-art Survey, number 33-42 in Lecture Notes in Autonomous Robots, 11(3):319-324, 2001.

D. Payton, R. Estkowski, and M. Howard, Pheromone robotics and the logic of virtual pheromones, In E. Sahin and William Spears, editors, Swarm Robotics Workshop: State-of-the-art Survey, number 3342 in Lecture Notes in Computer Science, pages 45-57, Berlin Heidelberg, 2005.

A. Christensen, R. O. Grady, and M. Dorigo, A mechanism to self-assemble patterns with autonomous robots, Technical report, TR/IRIDIA/2007-009, 2007.

D. Floreano, P. Durr and C. Mattiussi, Neuroevolution: from architectures to learning, *Evolutionary Intelligence*, 1(1): pp.47-62, 2008.

M. J. B. Krieger, J.B. Billeter, and L. Keller, Ant-like task allocation and recruitment in cooperative robots, Nature, 406:992-995, 2000.

A. E. Eiben et al., Collective Specialization for Evolutionary De- sign of a Multi-Robot System, Swarm Robotics, SAB2006 Second Int. Workshop, Revised Selected Papers, LNCS, Vol.4433, pp. 189- 205, 2007.

E. Tuci, et al., Evolving Homogeneous Neuro-Controllers for a Group of Heterogeneous Robots: Coordinated Motion, Cooperation, and Acoustic Communication, Artificial Life, Vol.14, No.2, pp. 157-178, 2008.

P. J. Angeline et al., An Evolutionary Algorithm That Constructs Recurrent Neural Networks, IEEE Trans. on Neural Networks, Vol.5, pp. 54-65, 1994.

X. Yao and Y. Liu, A New Evolutionary System for Evolving Artificial Neural Networks, IEEE Trans. on Neural Networks, Vol.8, No.3, pp. 694-713, 1997.

F. Gomez and R. Miikkulainen, Solving Non-Markovian Control Tasks with Neuroevolution, Proc. of the Sixteenth Int. Joint Conf. on Artificial Intelligence, pp. 1356-1361, 1999.

K. Stanley and R. Miikkulainen, Evolving Neural Networks through Augmenting Topologies, Evolutionary Computation, Vol.10, No.2, pp. 99-127, 2002.

N. Siebel and G. Sommer, Reinforcement Learning of Artificial Neural Networks, Int. J. of Hybrid Intelligent Systems, No.4, No.3, pp. 171-183, 2007.

W. Liu, A. Winfield, J. Sa, J. Chen, and L. Dou, Strategies for Energy Optimisation in a Swarm of Foraging Robots, volume 4433, pages 14-26. Springer Verlag, Berlin Germany, 2006.

H. Hamann and H. Worn, An analytical and spatial model of foraging in a swarm of robots, In E. Sahin, W.M. Spears, and Alan F. T. Winfield, editors, Second International Workshop on Swarm Robotics at SAB 2006, volume 4433, pages 43-55, Berlin, Germany, 2006.

R. O. Grady, R. Grofi, A. L. Christensen, F. Mondada, M. Bonani, M. Dorigo, Performance benefits of self-assembly in a swarm-bot, Technical report, TR/IRIDIA/2007-008, 2007.

K. Lerman and A. Galstyan, Mathematical model of foraging in a group of robots: Effect of interference, Autonomous Robots, 13:127-141, 2002.

D. Floreano, J. Urzelai, *Evolutionary robotics: The next generation*, Evolutionary Robotics III, Ontario (Canada): AAI Books, 2000.

D. Cliff, I. Harvey, and P. Husbands, *Explorations in evolutionary robotics*, volume 2, 1993.

R.A.Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Autonomation, 2:14-23.

Brooks, R. A., *Cambrian Intelligence*, The MIT Press, 1999.

R. C. Arkin, *Behavior-Based Robotics*, The MIT Press, 1998.

R. C. Arkin, Motor schema-based mobil robot navigation, International Journal of Robotics Research, 8: 92-112, 1989

N. Zaera, C. Clifi, J. Bruten, (Not) evolving collective behaviours in synthetic fish, in From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour, (Cambridge, MA), pp. 635-6, MIT Press, 1996.

M. J. Mataric, D. Clifi, Challenges in evolving controllers for physical robots, Journal of Robotics and Autonomous Systems, vol. 19, pp. 67-83, Oct. 1996.

H. H. Lund, J. Hallam, Evolving sufficient robot controllers, in Proceedings of the Fourth IEEE International Conference on Evolutionary Computation, (Piscataway, NJ), pp. 495-499, IEEE Press, 1997.

N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies. Evolutionary computation, 9(2):159-95, 2001.

N. Hansen, A. Auger, R. Ros, S. finck and P. Posik, Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009 *Proceedings of GECCO'10*, ACM, pp.1689-1696, 2010.

N. Hansen,S. Muller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es), Evolutionary Computation, 11(1):1-18, 2003.

N. Hansen, S. Kern, Evaluating the cma evolution strategy on multimodal test functions. In: Parallel problem solving from nature-PPSN VIII. Springer, :282-91, 2004

R. Ros, N. Hansen. A simple modification in cma-es achieving linear time and space complexity. Parallel Problem Solving from Nature-PPSN X. Springer, 296-305, 2008.

J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, MIT Press, Cambridge, MA, 1992.

J. R. Koza, Hierarchical genetic algorithms operating on populations of computer programs, Proc. of the Eleventh Int. Joint Conf. on Artificial Intelligence (IJCAI-89), 1 : 768-774, 1989.

C. Igel, Neuroevolution for reinforcement learning using evolution strategies, Proceedings of CEC'03, Vol.4, pp.2588-2595, 2003.

I. Rechenberg, Evolution strategies: Optimierung Technicher Systeme nach Prinzipien des Biologischen Evolution, Frommann Holzboog, Stuttgart, 1973.

J. H. Holland, Genetic algorithms and the optimal allocation of trials, SIAMJ. of Computing, 2 pp. 88-105, 1973.

L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement Learning: A Survey, Journal of Articial Intelligence Research 4, 237-285, 1996.

D. Goldberg, GENETIC ALGORITHMS in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

T. Back, F. Hoffmeister, H. Schwefel, A survey of evolution strategies, Proc. of the 4th Int. Conf. on Genetic Algorithms, 1991.

T. B ack, H. P. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evolutionary Computation, 1(1) : 1-23, 1993.

L. J. Eshleman, and J. D. Schaer, Real-Coded Genetic Algorithms and Interval-Schemata, Foundations of Genetic Algorithms, Vol. 2, pp. 187-202, 1993.

D. B. Fogel and L. J. Fogel, An introduction to evolutionary programming, Selected Papers from the European Conf. on Artificial Evolution, 1063 : 21-33, 1996.

R. Alami, H. Asama, R. Chatila, Proceedings of the Seventh International Symposium on Distributed Robotic Autonomous Systems, DARS04, Toulouse, France, 2004.

J. Ferber, Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, Addison Wesley Reading Vol. 1, 1999.

K. Sugawara, M, Sano, Cooperative acceleration of task performance: Foraging behavior of interacting multi-robots system, Physical, 100, pp.343-354, 1997.

T. Yu, T. Yasuda, K. Ohkura, Y. Matsumura, and M. Goka, Cooperative Transport by a Swarm Robotic System Based on CMA-NeuroES Approach, Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Vol.17, No.6, pp. 932-942, 2013.

M. Dorigo, and E. Şahin, Special issue: Swarm robotics, Autonomous Robots, 17: 111-113, 2004.

C. V. Jones and M. J. Mataric. Behavior-based coordination in multi-robot systems. In S. S. Ge and F. L. Lewis, editors, Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications, pages 549-569. CRC Press, Boca Raton, FL, 2006.

R. Beckers, O. E. Holland, and J.-L. Deneubourg, From local actions to global tasks: Stigmergy and collective robotics, In R.A. Brooks and P. Maes, editors, Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV), pages 181-189, Cambridge, MA, USA, July 1994. MIT Press.

E. Bonabeau, M. Dorigo, and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Santa Fe Institute Studies on the Science of Complexity, Oxford University Press, 1999.

S. Camazine, J.-L. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. Self-Organization in Biological Systems. Princeton University Press, Princeton, NJ, 2001.

N.R. Franks, S.C. Pratt, E.B. Mallon, N.F. Britton, and D.J.T. Sumpter. Information flow, opinion- polling and collective intelligence in house-hunting social insects. Philosophical Transactions of the Royal Society B: Biological Sciences, 357(1427):1567-1583, 2002.

I.D. Couzin. Collective minds. Nature, 455(7129):715-715, 2007.

I.D. Couzin and J. Krause. Self-organization and collective behavior of vertebrates. Advances in the Study of Behavior, 32:1-75, 2003.

D. J.T. Sumpter. The principles of collective animal behaviour. Philosophical Transactions of the Royal Society of London: Series B, 361:5-22, 2006.

G. Beni, From swarm intelligence to swarm robotics, In Şahin, E., Spears, W., eds, Swarm Robotics: State-of-the-art Survey, Lecture Notes in Computer Science 3342, Springer-Verlag, pages 1-9, 2005.

S. Camazine, J. L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, it Self-Organisation in Biological Systems, Princeton University Press, NJ, USA, 2001.

P. Grasse, *a reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la theorie de la stigmergie*. Essai. Insectes Sociaux, 6:41-83, 1959.

K. Ohkura, T. Yasuda, Y. Kawamatsu, Y. Matsumura and K. Ueda, MBEANN: Mutation-Based Evolving Artificial Neural Networks, *Advances in Artifical Life, the 9th European Conference on Artificial Life*, LNAI4648, pp.936–945, 2007.

K. Ohkura, T. Yasuda, Y. Kotani, and Y. Matsumura, A Swarm Robotics Approach to Cooperative Package-Pushing Problems with Evolving Recurrent Neural Networks, *Proceedings of SICE Annual Conference 2010*, LNAI4648, pp.706–711, 2010.

X. Yao, Evolving Artificial Neural Networks, Proc. of the IEEE, Vol.89, No.9, pp. 1423-1447, 1999.

K. Ohkura, T. Yasuda, and Y. Matsumura, Coordinating the Adaptive Behavior for Swarm Robotic Systems by Using Topology and Weight Evolving Artificial Neural Networks, *Proceedings of WCCI 2010 IEEE World Congress on Computational Intelligence, 2010 IEEE Congress on Evolutionary Computation*, pp.1788–1795, 2010.

K. Ohkura, T. Yasuda, T. Sakamoto and Y. Matsumura, Evolving Robot Controllers for a Homogeneous Robotic Swarm, *Proceedings of 2011 IEEE/SICE International Symposium on System Integration*, pp.708–713, 2011.

K. Ohkura, Y. Matsumura, T. Yasuda, and T. Matsuda, Evolutionary Robotics Approach to Autonomous Task Allocation for a Multi-Robot System, Proc. of Artificial Neural Networks in Engineering 2008, Intelligent Engineering Systems Through Artificial Neural Networks, Vol.18, pp. 121-128, ASME Press, 2008.

J. B. Mouret and S. Doncieux, Incremental evolution of animats' behaviors as a multi-objective optimization , Lecture Notes in Computer Science, 5040, pp.210-219, 2008.

D. Bajaj and M. H. Ang Jr., An Incremental Approach in Evolving Robot Behavior , Proceedings of the Sixth International Conference on Control, Automation, Robotics and Vision, ICARCV, 2000.

F. Gomez and R. Miikkulainen, Incremental Evolution of Complex General Behavior , Adaptive Behavior, No.5, pp.317-342,1997.

R. De Nardi, J. Togelius, O. Holland and S. Lucas, Evolution of Neural Networks for Helicopter Control: Why Modularity Matters , In: IEEE Congress on Evolutionary Computation, pp.1799-1806, 2006.

S. Nolfi, D. Paris, Evolving non-Trivial Behaviors on Real Robots: an Autonomous Robot that Picks up Objects , Proceedings of 4th Conference of the Italian Association for Artificial Intelligence, 1995.

B. S. Everitt, Cluster Analysis, Edward Arnold and Halsted Press, third edition, 1993.

M.E.J. Newman, Networks, Oxford University Press, 2010.

M. Girvan, and M. E. J. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences of the United States of America, Vol.99, pp.7821-7826, 2002.

M. E. J. Newman, Detecting community structure in networks, Department of Physics and Center for the Study of Complex Systems, University of Michigan, ANN Arbor, MI 48109–1120, USA, Received 10 November 2003.

M. E. J. Newman, Fast algorithm for detecting community structure in networks, Department of Physics and Center for the Study of Complex Systems, University of Michigan, ANN Arbor, MI 48109, USA, published 18 June 2004.

M.E.J. Newman, Modularity and community structure in networks, Proceedings of the National Academy of Sciences of the United States of America, Vol.103, No.23, pp.8577-8582, 2006.

J. Duch and A. Arenas, Community detection in complex networks using extremal optimization. Phys. Rev. E 72, 027104, 2005.

W. Liu, A. F. T. Winfield, J. Sa, J. Chen, and L. Dou, Towards Energy Optimization: Emergent Task Allocation in a Swarm of Foraging Robots, Adaptive Behavior, Vol.15, Issue 3, pp.289-305, 2007.

R. M. Fagen, and N. J. Mankovich, Two-act transitions, partitioned contingency tables, and the *significant cells problem*, Animal Behaviour, Vol.28, Issue 4, pp.1017-1023, 1980.

G.C. William, Some Methods for Strengthening the Common x^2 Tests, Biometrics, Vol.10, No.4, pp.417-451, 1954.

K. Ohkura, T. Yasuda, Y. Matsumura, *Analyzing macroscopic behavior in a swarm robotic system based on clustering*, SICE Annual Conference (SICE), pp. 13-18, 2011.

S. Kazadi, Swarm Engineering. PhD thesis, California Institute of Technology, Pasadeba, CA, USA, 2000.

A. F. T. Winfield, C. J. Harper, and J. Nembrini, Towards dependable swarms and a new discipline of swarm engineering, In Proceedings of the International Workshop on Simulation of Adaptive Behavior, SAB 2004, Vol. 3342 of lecture notes in computer science, pages 126-142. Springer, Berlin, Heidelberg, 2004.

V. Crespi, A. Galstyan, and K. Lerman, Top-down vs bottom-up methodologies in multi-agent system design, Autonomous Robots, 24(3):303-313, 2008.

Research Achievements

Journal papers

1. Tian Yu, Toshiyuki Yasuda, Kazuhiro Ohkura, Yoshiyuki Matsumura and Masanori Goka, “Cooperative Transport by a Swarm Robotic System Based on CMA-NeuroES Approach“, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol.17 No.6, pp.932–942, 2013

2. Tian Yu, Toshiyuki Yasuda, Kazuhiro Ohkura, Yoshiyuki Matsumura and Masanori Goka, “Robust swarm robotics system using CMA-NeuroES with incremental learning“, International Journal of Engineering Research & Technology, Vol. 4, No. 11, pp. 217-226, 2015

International Conference

1. Tian Yu, Toshiyuki Yasuda, Kazuhiro Ohkura, Yoshiyuki Matsumura, and Masanori Goka, “Apply Incremental Evolution with CMA-NeuroES Controller for a Robust Swarm Robotics System“, Proceedings of the SICE Annual Conference 2014, pp.295–300, 2014.

2. Tian Yu, Tsuguo Fujita, Toshiyuki Yasuda, Kazuhiro Ohkura, Yoshiyuki Matsumura and Masanori Goka, “Cooperative Transport by a Swarm Robotic System Based on CMA-NeuroES Approach“, Proceeding of the 16th Asia Pacific Symposium on Intelligent and Evolutionary Systems, pp.1-7, 2012.

3. Tian Yu, Toshiyuki Yasuda, Kazuhiro Ohkura, “A duration based behavior analyze approach for swarm robotics system“, The 34th Chinese Control Conference & SICE Annual Conference, pp. 295-300 July 28-30, 2015.

4. Tian Yu, Toshiyuki Yasuda, Kazuhiro Ohkura, “Understanding autonomous task allocation by clustering a swarm robotics system“, SWARM 2015: The first International Symposium on Swarm Behavior and Bio-Inspired Robotics, October 28-30, Kyoto, Japan, pp. 192-193, 2015.