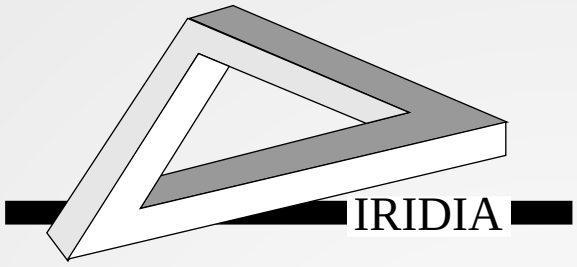


Evolution of Coordinated Motion Behaviors in a Group of Self-Assembled Robots

V. TRIANNI

Technical Report No.

TR/IRIDIA/2003-25
May 2003

DEA thesis

Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle

50 av. F. Roosevelt • CP 194/6 • B-1050 Bruxelles • Belgium
Phone: +32-2-650.27.29 • Fax: +32-2-650.27.15

Evolution of Coordinated Motion Behaviors in a Group of Self-Assembled Robots

by

Vito Trianni

Université Libre de Bruxelles, IRIDIA
Avenue Franklin Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
vtrianni@ulb.ac.be

Supervised by

Marco Dorigo, Ph.D.

Maître de Recherches du FNRS
Université Libre de Bruxelles, IRIDIA
Avenue Franklin Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
mdorigo@ulb.ac.be

May, 2003

A thesis submitted in partial fulfillment of the requirements of the *Université Libre de Bruxelles, Faculté de Sciences Appliquées* for the

DIPLOME D'ETUDES APPROFONDIES (DEA)

Abstract

In this work, we introduce a swarm robotic system, called a *swarm-bot*. A *swarm-bot* is a self-assembling and self-organizing artifact composed of a swarm of *s-bots*, mobile robots with the ability to connect to/disconnect from each other. In particular, we address the problem of synthesizing controllers for the *swarm-bot* using Artificial Evolution. We describe the motivation behind the choice of the evolutionary approach and we provide examples of its application, detailing the results obtained in different tasks, namely coordinated motion and hole avoidance. We show how evolution is able to produce simple but effective solutions, which lead to the emergence of self-organization in the *swarm-bot*.

Acknowledgments

First of all, I would like to thank my supervisor Marco Dorigo, for the help he gave me in many occasions. I thank also Stefano Nolfi and Gianluca Baldassarre, for the helpful discussions we always have in Rome. I thank Ciro, Josh, Halva, Rodi, Gianni, Max and all my colleagues for making IRIDIA everything but a research lab. A particular thought goes to my parents and to my sisters Manuela and Sarah, who are always ready to support me in (all) my decisions. Last but not least, I want to thank Claudia for her constant support and her love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work addresses the problem of defining the control system for a group of autonomous robots. In particular, we apply techniques derived from Artificial Evolution, and we show how they can produce simple but effective and robust solutions.

There are multiple motivations that lay behind the choice of Artificial Evolution as a tool for synthesizing controllers for a group of robots. First, Artificial Evolution can bypass many difficulties encountered in the hand design. In fact, even in a single-robot domain, the problem of designing the control system is not trivial at all. The designer must discover the rules that must be encoded into the controller, in order to achieve a certain goal. To do so, it is necessary to know the environment in which the robot should act, and to predict the outcome of a sequence of actions performed by the robot. When the environment is dynamic and unpredictable, designing the control system could be very challenging. In a distributed multi-robot domain, this problem is worsened by the fact that each robot is an independent unit that can take its own decisions depending on the current sensory pattern and its internal state. Furthermore, robots interact with each other, making the system much more dynamic and unpredictable. In a similar scenario, Artificial Evolution may perform well as it directly tests the behavior displayed by the robot embedded in their environment. This approach, working in a bottom-up direction, bypasses the decomposition problems given by a top-down approach, typical of behavior-based or rule-based systems. Furthermore, Artificial Evolution can exploit the richness of solutions offered by the complex dynamics resulting from robot-robot and robot-environment interactions.

In this work, we present the results obtained from the ongoing work

within the SWARM-BOTS project[1]. The aim of the SWARM-BOTS project is the development of a new robotic system, called a *swarm-bot* [44, 33]. The *swarm-bot* is defined as an artifact composed of simpler autonomous robots, called *s-bots*. An *s-bot* has limited acting, sensing and computational capabilities, but can create physical connections with other *s-bots*, thereby forming a *swarm-bot* that is able to solve problems the single individual cannot cope with.

A basic ability that a *swarm-bot* should display is *coordinated motion*, that is, the ability to move coherently across the environment as a result of the cooperation of the *s-bots* assembled in the *swarm-bot*. We chose to study coordinated motion not only because it is of fundamental importance, but also because it represents the first step toward the solution of many other cooperative problems, like navigation on rough terrains or collective prey retrieval. In this work, we present the results obtained in a coordinated motion task, where a *swarm-bot* is composed of 4 *s-bots* connected in a linear formation. We also present the case in which the same *swarm-bot* is placed in an arena with holes, where it should display hole avoidance behaviors. We show that the evolved controllers are efficient and present also generalization properties both to changes in size and shape of the *swarm-bot*, and to the introduction of obstacles in the environment.

In the rest of this chapter, we will first present the state-of-the-art, describing the research fields that constitute the starting point of this work. This is the topic of Section 1.1. In Section 1.2, we present in detail the SWARM-BOTS project and the *swarm-bot*. Finally, Section 1.3 briefly summarize the contents of this report.

1.1 Background

In the last decade there has been a growing interest in the development of complex robotic systems which could present features like versatility, robustness or capacity to perform complex task in unknown environments. In order to achieve these features, the single-robot approach was often abandoned in favor of more complex systems, involving multiple robots working in strict cooperation. In fact, developing and controlling a single, multi-purpose robot is a complex task, and it may also be very expensive. Moreover, the single-robot approach suffers the problem that even small failures may prevent the accomplishment of the whole task. A group of simple and cheap

[1]A project funded by the Future and Emerging Technologies Programme (IST-FET) of the European Community, under grant IST-2000-31010.

robots may be able to efficiently accomplish many tasks that go beyond the capabilities of the individual robot. This idea is at the base of the research in the Collective Robotics field and in the Metamorphic Robotics field, which cover most of the related research done so far. On a parallel track, the research in autonomous robotics has faced the challenge of synthesizing the controllers for such robotics systems. Among the different approaches that have been proposed, we are mainly interested in the study of Evolutionary Robotics, which applies techniques derived from Artificial Evolution to the definition of controllers for autonomous robots (for a review see [37]). In this section, we present the state-of-the-art in all these research fields, which constitutes the starting point of our research.

1.1.1 Collective Robotics

The field of Collective Robotics focuses on the study of robotic systems that are composed of a number of autonomous robots which concurrently act in order to reach a common goal (for an overview of the field, see for example [39]). The main motivation behind the study of collective robotic systems lays in the possibility to decompose the solution of a complex problem into sub-problems that are simpler and that can be faced by simple robotic units.

Collective robotics research has mainly focused on the achievement of coordination of several systems. For example, Gerkey and Matarić [21] propose a dynamic task allocation method based on auction exchange in order to achieve cooperation in a group of robots. Agassounon et al. [1] use a scalable algorithm based on a threshold model for the allocation of robots in a puck collecting and clustering task. Melhuish [31] describes a clustering task collectively performed by a group of cooperating robots.

Another interesting aspect of collective robotics is given by the robustness that can be achieved by providing redundancy to the whole system. For example, Parker [38] defined a software architecture for fault tolerant control of heterogeneous robots which allows a robot to select the correct action to be performed depending on the requirement of the mission, the activities of the other robots, the environmental conditions, and its own internal state. Goldberg and Matarić [22] demonstrate the effectiveness of a behavior-based approach for the definition of robust and easily modifiable controllers for distributed multi-robot collection tasks.

A controversial aspect in the collective robots community is given by the use of communication. In some cases, communication can be useful for modeling the internal state of other agents, or for committing the execution

of a particular action to a teammate. Bonarini and Trianni [8] have shown that the communication of "cooperation proposals" can help learning cooperative behaviors. Matarić [30] showed how communication can be used to transmit sensory information to other robots in order to increase the coordination of the group. Communication was also used as a mean to distribute reward to other members of the group in a reinforcement learning task. Balk and Arkin [4] have shown that cooperation can emerge in a group of robots if they are not able to independently accomplish a given task. They show that, depending on the task, communication may or may not be helpful, and that often very simple forms of communication are sufficient to the accomplishment of a cooperative task.

1.1.2 Metamorphic Robotics

The major effort in Metamorphic Robotics research has been to study single robots composed of a collection of identical modules where each module is a simpler robot. Usually, every module is in contact with at least another module so that a more complex structure is defined. All modules have the same physical structure and each module is autonomous from the viewpoint of computation and communication.

Chirikjian et al. [13] describe a metamorphic robot composed of identical hexagonal modules that can aggregate as a two-dimensional structure with varying geometry. Robot configuration is computed by a centralized control that uses mathematical properties of the lattice connectivity graph associated to the structure. The work is closer to geometrical and kinematics research where the goal is to compute the minimum number of moves to reach a given configuration rather than to the problem of controlling in real time a complex robot structure. Yim et al. [54] have developed PolyBot, a metamorphic robot defined by a sophisticated basic module with on-board computing capabilities. Also in this case, however, the robot shape is defined by a centralized control. Murata et al. [34] consider a system of 2D modules called Fracta that can achieve planar motion by walking over each other. The reconfiguration motion is actuated by varying the polarity of electromagnets that are embedded in each module. Kamimura et al.[27] have developed MTRAN, which get a lot of attention due to excellent results with real hardware. This system uses a large number of modules with only one degree of freedom, exists physically and can self-reconfigure. Despite the very good hardware flexibility, it has a centralized control algorithms. Shen at al. [47, 12] with CONRO proposed another work that follows the above-mentioned directions. Robot morphology is ensured by modular iden-

tical structures strongly coupled by physical connectors. Robot shapes are predefined and module moves are precomputed by planners based on global information while no effort is made on distributed/on-line control, adaptation and self-reconfiguration. Only recently, a decentralized control has been developed for this system by Støy et al. [49]. This system allows to manually change the position of the hardware modules in the structure while the system is running and each module autonomously re-adapts its behavioral role in the system.

1.1.3 Evolutionary Robotics

The problem of defining a controller for a robotic system has been approached from many different directions: inferential planners, behavior-based robotics and learning classifier system are only some example of the possible ways of controlling a robot. Among these, Evolutionary Robotics is a very promising technique for the synthesis of robot controllers [37]. It is inspired on the Darwinian principle of selective reproduction of the fittest individual in a population and makes use of genetic algorithms [26]. Starting from a population of *genotypes*, each encoding the control system (and sometimes the morphology) of the robot, the evolutionary process evaluates the performance of each individual controller, letting the robot free to act in its environment following the genetically encoded rules. The fittest robots are allowed to reproduce generating copies of their genetic material, which can be changed by several genetic operators (e.g., *mutation*, *crossover*). This process is iterated a number of times (*generations*) until a satisfying controller is found that meets the requirements stated by the experimenter in the performance evaluation (*fitness function*).

Many difficult control problems have been easily solved relying on the evolutionary approach. For example, Nolfi [36] successfully evolved a controller for the Khepera robot [32] in order to find and stay close to a target object. The Khepera, equipped only with infrared proximity sensors, was placed in a rectangular arena surrounded by walls and containing the target cylindrical object that had to be found. The evolved controller outperformed the behavior-based one, as this task is difficult to be solved by hand design, In fact, a difficult discrimination must be performed between the sensory pattern generated by a wall and the one generated by the target obstacle. Harvey et al. [24] addressed the problem of navigation acquiring information about the evironment from a camera. They evolved both the morphology of the visual receptive field and the architecture of the neural network. Using these settings, they successfully synthesized an individual for approaching a

triangular shape painted on a wall and contemporary avoiding a rectangular one, guided by the vision system. Floreano and Mondada [20] evolved a homing navigation behavior for a Khepera robot, using a recurrent neural network. They showed that the internal dynamics of the recurrent network could encode a sort of map of the environment that leads to an efficient homing behavior.

More recently, the evolutionary robotic community has approached the problem of defining collective behaviors. For example, Baldassare et al. [6] evolved group behaviors for simulated Khepera robots, which had to aggregate and navigate toward a light target. Quinn [41] evolved coordinated motion behaviors with two Khepera. On the same track, Quinn et al. [42] studied coordinated motion with three wheelchair robots.

1.2 Swarm Robotics: the SWARM-BOTS Project

Swarm robotics is a novel approach to the design and implementation of robotic systems. These systems are composed of *swarms* of robots which tightly interact and cooperate to reach their goal. Swarm robotics can be considered as an instance of the more general field of collective robotics (see Section 1.1.1). It is inspired by the social insect metaphor and emphasizes aspects like decentralization of the control, limited communication abilities among robots, emergence of global behavior and robustness. In a swarm robotic system, although each single robot composing the swarm is a fully autonomous robot, the swarm as a whole can solve problems that the single robot cannot solve because of physical constraints or limited abilities.

As mentioned above, this work is carried out within the SWARM-BOTS project, whose aim is the development of a swarm robotic system, called *swarm-bot*. A *swarm-bot* is defined as an artifact composed of a swarm of *s-bots*, mobile robots with the ability to connect to/disconnect from each other (for more details regarding the hardware, see Section 3.1). *S-bots* have simple sensors and motors and limited computational capabilities. Their physical links are used to assemble into a *swarm-bot* able to solve problems that cannot be solved by a single *s-bot* (see Figure 1.1).

The *swarm-bot* concept lies between the two main streams of robotics research described above, that is, collective robotics and metamorphic robotics. In fact, in collective robotics, autonomous mobile robots interact with each other to accomplish a particular task, but, unlike *s-bots*, they do not have the ability to attach to each other by making physical connec-

Figure 1.1: Graphical visualization of an *s-bot*.

tions. On the other hand, a self-reconfigurable robotic system consists of connected self-contained modules that, although autonomous in their movements, remain attached to each other, lacking the full mobility of *s-bots*.

Some examples can be useful to clarify the capabilities of this robotic system. In the *swarm-bot* form, the *s-bots* are attached to each other and the robotic system is a single whole that can move and reconfigure along the way when needed. For example, it might have to adopt different shapes in order to go through a narrow passage or overcome an obstacle. Physical connections between *s-bots* are important for building pulling chains, as for example in an object retrieval scenario (see Figure 1.2a). They can also serve as support if the *swarm-bot* is going over a hole larger than a single *s-bot*, as exemplified in Figure 1.2b, or when the *swarm-bot* is passing through a steep concave region, in a navigation on rough terrain scenario. Anyway, there might be occasions in which a swarm of independent *s-bots* is more efficient: for example, when searching for a goal location or when tracking an optimal path to a goal.

The above examples represent the family of tasks a *swarm-bot* should be able to perform. Although these tasks present many differences from each other, they share many common aspects, among which the capability to perform *aggregation* and to distributely *coordinate* the activity of the group. Aggregation is of particular interest because it is a prerequisite for the development of other forms of cooperation: for example, in order to assemble in a *swarm-bot*, *s-bots* should first be able to aggregate. Therefore, the aggregation ability can be considered as the precondition for the realization of other tasks that the *swarm-bot* is expected to be able to carry out. On the other hand, the ability to coordinate the activities of the group is crucial for the effectiveness of a *swarm-bot*: for example, when carrying a heavy

(a) (b)

Figure 1.2: Graphical visualization of possible scenarios involving a *swarm-bot*. (a) Retrieving a circular object. (b) Passing over a trough.

object that a single *s-bot* cannot move, all *s-bots* should coordinate and pull or push in the same direction, in order to maximize the performance of the *swarm-bot*. Similarly, when two or more of such objects have to be transported, it is desired that the whole group of *s-bots* coordinates its activity focusing on a single object rather than having small and inefficient groups attempting to move different targets. The aggregation task has been studied in previous research activities and solved both with hand design and using artificial evolution [44, 50]. In this work, we focus on the coordinated activity task, trying to solve the problem of coordinated motion in a *swarm-bot*, as described in Chapter 4.

1.3 Report Layout

This report is organized as follows. In Chapter 2 we discuss about the motivation that led us to the choice of Artificial Evolution as a tool for synthesizing controllers for the *swarm-bot*. We describe the features and challenges of a swarm robotic system like the *swarm-bot*. We introduce the notion of self-organization, which can be exploited for gaining useful insights about the working principles of swarm robotic systems. We detail the problems that have to be addressed when designing a control system for the *swarm-bot*, and we show how Artificial Evolution can cope with them.

In Chapter 3, we present the experimental setup used in the experiment we performed. In particular, we describe the simulation model defined for an *s-bot* and its sensory-motor configuration. We also describe the evolutionary algorithm we employed in all the performed experiments.

In Chapter 4, the results obtained in the attempt to evolve coordinated

motion behavior for the *swarm-bot* are presented. We describe the task and the peculiarity of the experiments performed. We show that we were able to evolve efficient behaviors, that generalize to different situation in which the *swarm-bot* has size and shape different from those used during the evolution. Besides, we show how the behavior also generalizes in situations where obstacles are added into the environment.

In Chapter 5, we present the hole avoidance task, which consists of coordinated motion in an environment with holes. These hazards have to be recognized and avoided by the *swarm-bot*, as the holes are too big to let the *swarm-bot* pass over. We present the peculiarity of the experimental setup and the results achieved evolving three different neural network architectures, and we analyze their performance. Finally, we show how the evolved strategies are able to display generalization properties.

In Chapter 6, we draw the conclusions of this work, highlighting the important aspects of this research. Finally, we indicate the future directions to be followed.

Chapter 2

Why Artificial Evolution?

In this chapter, we describe the challenges we are facing in developing a control system for a *swarm-bot* and the methodologies we can apply to the solution of the design problem. In particular, we highlight the principal features of a swarm robotic system and we explain how exploiting the notion of Self-Organization can be of fundamental importance for a complex system like a *swarm-bot*. We also discuss the possible solutions to the design of a control system for the *s-bots*, that could allow them to self-organize. Among these, Artificial Evolution can be considered very promising, as it allows the evolution of a controller in a bottom-up approach, without much intervention from the designer.

2.1 The Challenges of Swarm Robotics

As mentioned above, swarm robotics is characterized by the presence of swarms of robots that tightly interact in order to reach their goals. Controlling such robotic systems is a challenging task: decentralization, robustness, adaptivity, embodiment, complex dynamics are features that have to be taken into account when developing a control system. All these features characterize the behavior of insect colonies and other animal societies, which are the main source of inspiration for swarm robotics. In this section, we will discuss the advantages these features can bring to a *swarm-bot*, along with the challenges they issue. During the discussion, we will also provide examples of natural systems which exhibit these important features.

2.1.1 Decentralization

When developing the controller for a multi-agent system like the *swarm-bot*, the first problem to be faced concerns the choice between a *centralized* and a *distributed* approach. In literature, we can find many examples of both, and there are also many intermediate approaches, in which control is decentralized only to some extent.

Speaking in general words, the centralized approach can be defined as a single machine/agent/entity that takes decisions for all the other agents that have to be controlled. For example, a group of robots that has to perform a foraging task could receive instructions from a remote workstation that, knowing the environment and the position of all the robots, could drive them toward the food sources, possibly optimizing the number of robots that can exploit each food source. The centralized planning of the instructions to be sent to each robot requires the combination of the state space of all the robots in a single space, whose dimension grows exponentially with the number of robots [29]. Furthermore, a communication medium must be provided between the centralized controller and the robots, and this medium must be reliable, because loosing contact with some robots will make them useless. Because of these requirements, the cost of centralization increases exponentially with respect to the size of the group.

On the contrary, decentralization leads to the distribution of the decision making process among all the agents composing the group. Each agent is responsible for its own actions, which are taken independently from the other individuals, leading to a noticeable reduction in the complexity of the control systems. In this way, the behavior of a single agent can be very simple, but the entire group can still display complex behaviors. A striking example of decentralization is given by many animal societies. For example, insects like ants or bees take most of their decisions in a distributed way without being governed by any leading individual in the colony, and at the same time achieving an extremely efficient and organized behavior at the colony level. Ants are able to trace the shortest path from the nest to a food source without any a priori knowledge about the environment [14]. This behavior is a result of the decisions taken individually by each ant and by the local interactions among ants and between ants and environment. Similarly, a fish school can move in a very coherent way, but there is no leader that instructs the group on the direction to take and the turns to perform. In fact, the flocking behavior of the fish school is a result of individual decisions and local interactions (for a review, see also [9]).

We mentioned local interactions as a mean to achieve coordination in

a group. In some cases, these interactions can be thought of as a sort of communication among agents. Indeed, in a decentralized system, a communication medium among agents is often provided, as communication can favor cooperation [30]. Direct communication turns to be required when it is necessary to know the internal state of other agents in order to take some decision, and it is difficult to infer the internal state from the observation of the other agent's behavior. However, not always direct communication can increase the efficiency in solving a problem, but often simpler form of communication are sufficient [4]. In a *swarm-bot*, we intend to use indirect communication among individuals (*stigmergy* [23]), thus limiting the use of direct communication. Stigmergy takes places if an individual modifies the environment in a way that can be sensed by other individuals or that can influence their behavior. Stigmergy is a form of communication that takes place trough the environment, favoring local interactions among individuals, thus coordinating and regulating the activity of the group. This phenomenon is often observed in insect societies. For example, ants lay a chemical substance, called *pheromone*, which attracts other ants: using this simple signal, ants can efficiently perform tasks like foraging, recruiting and nest building.

2.1.2 Robustness

Robustness is directly linked to decentralization. In a centralized system, in fact, the failure of the central controller would affect the whole group, while a decentralized system, not relying on a single controller, can continue to work even if some of its parts are not available any more.

However, a distributed control alone is not enough to obtain robustness. For example, let us consider a group of agents that are able to achieve their goal performing an ordered sequence of sub-tasks, each executed by a specialized agent. In this case, the system may be decentralized, but not robust, because the failure of one agent will lead to the failure of the entire group. In the swarm robotic system, the main way to achieve robustness is to provide redundancy to the system, replicating its parts many times. Thereby, the removal of some components will not affect the functionality of the system, but it will lead to a graceful degradation of the performance. Redundancy and the consequent robustness are typical features of insect colonies, that are able to function even after the removal of many individuals.

2.1.3 Adaptivity

In some cases, robustness in a system can be a direct consequence of its adaptivity. A striking example of robustness as a result of adaptivity is given by ant species of the *Pheidole* genus. In most species of this genus workers are physically divided into two fractions: the small minors, who fulfill most of the quotidian tasks, and the larger majors, who are responsible for seed milling, abdominal food storage, defense or a combination of these. Wilson [53] experimentally changed the proportion of majors to minors. By diminishing the fraction of minors, majors get engaged in the tasks usually performed by minors and replace them efficiently. Wilson [53] observed, that within one hour of the ratio change, majors adapt themselves to the new situation and take over the minors' work. The colony is able to achieve robustness by adapting to the new, unexpected situations. Similarly in a *swarm-bot*, this kind of adaptation can lead to a dynamic division of labor that is very important in order to perform complex tasks.

Adaptivity at the colony level is one of the most important characteristics displayed by social insects. Colonies with thousands of individuals need to adapt to changing conditions very quickly. The individual behaviors are flexible to several internal and external factors such as food availability, climatic conditions or phase of colony development. In a similar way, a *swarm-bot* should be able to adapt to changing environmental conditions, in order to obtain the best achievable performance.

2.1.4 Embodiment and Complex Dynamics

Adaptivity of the system is even more important when the system is embodied and physically interacts with its environment. The result of physical interactions is difficult to predict and requires that the system is able to adapt to many different situations.

Speaking in general words, embodiment stresses the importance of physical aspects of the system (like mass, friction, shape) and its interaction with the environment. Therefore, an embodied system is characterized by complex dynamics that are the result of the agent-environment interaction. Exploiting this complexity can be useful in order to produce complex behaviors with relatively simple controllers. An experimental evidence is given in [37], where the task of discriminating walls from cylindrical obstacles is presented. The results showed that this task cannot be efficiently solved by a disembodied neural network that was trained using the back-propagation algorithm. However, the task was successfully solved by the same network

embodied in a Khepera robot that was let free to move. This suggests that physical interactions with the objects to be discriminated simplifies the task.

The importance of physical interactions is amplified in a swarm robotic system, where each agent dynamically interacts not only with the environment, but also with all the other agents. In other words, a swarm robotic system is characterized by very complex dynamical interactions among agents and between agents and environment. In a *swarm-bot*, physical connections between *s-bots* increase the complexity of the system's dynamic. This confirms the importance of the study of embodied controllers in a *swarm-bot*, where *s-bots* can influence each other also by creating physical forces.

2.2 Self-Organization and Swarm Robotics

In the previous section, we discussed about the main features of a swarm robotic system like the *swarm-bot*. The inherent complexity of a *swarm-bot* suggests that the design of its controller is a particularly challenging task. Unfortunately, there are no basic principles that can be followed in order to design a controller for swarm robotic systems. However, we can approach the solution of this problem looking at complex system theories. In particular, useful insights can be found in the notion of *self-organization*.

Self-organization can be defined as "the appearance of structure or pattern [in a system] without an external agent imposing it" [25]. More precisely, self-organization explains how, in a system, global level order emerges from the numerous local interactions that takes place among the lower-level components of the system. In other words, a system self-organizes driven by its own components, which interact relying only on local information, without any reference to the system as a whole.

The notion of "self-organization" started to be discussed in the middle of the 20^{th} century by a multi-disciplinary group of scientists, like the thermodynamicist Nicolis and Prigogine or the cyberneticians Ashby and Von Foerster [35, 3, 51]. Prigogine won the Nobel prize for his study of *dissipative systems*, that is, systems able to continuously dissipate energy preserving a particular dynamic state. These systems are able to maintain constant or decrease their own entropy dissipating the excess energy in the surroundings. A well known example is given by the Bénard convection cells that can be observed when heating a thin layer of a vegetable oil. The vertical temperature gradient in the horizontal oil layer causes an ordered movement of the molecules in the liquid that results in a global hexagonal pattern, which can be observed on the substrate. Prigogine suggested that

self-organization typically takes place in non-linear systems far from their thermodynamic equilibrium point.

In the same period, Ashby and von Foerster begun their work on self-organization. Ashby noted that a self-organizing system is a system that evolves toward a state of equilibrium, called *attractor*. On the other hand, Von Foerster supported the notion of "order form noise", claiming that injecting noise in a system can move it across its state space, with the possibility of ending in a more stable (ordered) state.

Starting from these pioneers, the importance of self-organization has been recognized in the study of many complex systems, ranging from chemistry to biology. As mentioned above, global order in a self-organizing system is the result of *local interactions* among the individuals composing the system. When in the disordered state, individual actions and interactions are deeply influenced by randomness or noise, and result in the so called *random fluctuations* of the system around its state. Then, self-organization may emerge from the interplay of two basic mechanisms: *positive* and *negative feedback*. Positive feedback consists in the amplification of the random fluctuations of the system: it can be seen as a snowball effect that increases exponentially and drive the system toward a stable state. On the contrary, negative feedback serves as a regulatory mechanism, and it is often a result of the amplification itself, that exhausts the resources of the system. Negative and positive feedback are responsible for maintaining a system in its stable state, restoring the organization after any deviation caused by some external influence.

As an example, let us consider again the Bénard convection cells mentioned above. When heating the thin layer of oil, a temperature gradient is created between the bottom and the top of the layer. However, the system remains in a stable state where heat is dissipated by conduction until a certain threshold is reached. At this point, random fluctuations and local interactions may trigger the self-organization process. In fact, a small portion of the liquid at the bottom may rise slightly because of random movements of the molecules (random fluctuations). It will be surrounded by a colder region, and, being less dense, it will be pushed up (local interactions). The more it rises, the colder the surroundings and the higher the rising force (positive feedback). The same mechanism applies to a cold portion of liquid at the top: a small downward movement caused by random fluctuations is amplified by the interaction with the warmer (and thus lighter) liquid in the surroundings. The amplification process terminates once all the liquid present convection cells, that is, when the resources of the system have been exhausted (negative feedback).

Self-organization can be of particular interest for studies in swarm-robotics in general, and specifically for the *swarm-bot* as it can explain the behavior of many biological systems, like ant colonies or fish schools, from which swarm robotics takes inspiration. Moreover, a particular form of self-organization taking place in insect societies, called *self-assembling*, is strictly related to the SWARM-BOTS project: self-assembling is the self-organized creation of structures as a result of connections among individuals composing the system (for a review of self-assembling in insect societies see [2]). Animal societies present multiple forms of self-organization and self-assembling. In such systems, the interactions among individuals are made using rules of thumb that require: (i) a limited cognitive ability and (ii) a limited knowledge of the environment [9, 10, 11, 15, 17, 19, 43, 45, 46]. Also in this case, we can recognize the basic features of self-organization: local interactions, random fluctuations, positive and negative feedback mechanisms. As an example, we can mention aggregation in the bark beetle larvae *Dendroctonus micans* [16]. Normally, these larvae individually search for a fruitful feeding site, moving randomly (random fluctuations). When they start feeding in a good location, they start to emit a chemical signal, a pheromone that diffuses in air and serves as communication medium (local interactions, stigmergic communication). At this point, the aggregation process is triggered: in presence of a pheromone gradient, larvae react by moving in the direction of higher concentration of pheromone, thus reinforcing the chemical signal coming from the aggregation site (positive feedback mechanism). The aggregation ends when all the larvae have clustered in one location (negative feedback mechanism resulting from the exhaustion of larvae) [16].

The above example shows how order in a system, that is, the aggregate, can emerge from simple individual rules and local interactions. This kind of behavior corresponds to what we want to observe in a *swarm-bot*. Then, why not designing a *swarm-bot* able to self-organize? In fact, self-organizing systems hold the features we want to provide to a *swarm-bot*, which have been discussed in Section 2.1:

Decentralization. All the elements of a self-organizing system are, by definition, autonomous: there is no leader that drives the organization of the system, which is not a result of some recipe, blueprint or template. The control is distributed, and all parts of the system contribute to the emergence of the organization. Furthermore, every element of a self-organizing system relies only on local information and interacts locally with the other elements of the system, suggesting that its behavior can be modeled with simple rules.

Robustness. When a self-organizing system reaches its stable state, it will be very difficult to destabilize it, because it has the natural tendency to return in its stable configuration, which constitutes an attractor for the system. This is mainly a result of the feedback mechanisms that continuously maintain the organization in the system. A self-organizing system is thus robust to environmental changes, but also to the failure of some of its components, given its high redundancy.

Adaptivity, Embodiment, Complex Dynamics. A system that self-organize naturally adapts to its environment, because it reaches a stable state driven by internal forces and by the interaction with the environment itself. Furthermore, a self-organizing system presents non-linearities and complex far-from-equilibrium dynamics, that is, the system reaches a stable state that is not an equilibrium point: this make it possible to have a dynamic system with fast reactions, which favors adaptation to environmental changes and the production of new responses to new, unexpected situations. The physical properties of the system have a big influence on its dynamics, and on the type of organization that will be reached: for example, the viscosity of the oil determines the size of the Bénard convection cells. This confirms the importance of embodiment in the study of self-organizing systems

In conclusion, a *swarm-bot* should be self-organizing in order to exploit all the advantageous features that pertain to self-organizing systems. However, we still have to understand how to design the control system in order to obtain self-organization. This "design problem" is the topic of the following section.

2.3 The Design Problem

As mentioned above, we want to design the control system for the *s-bots* in order to obtain self-organization in a *swarm-bot*. However, designing such a of control system is not a trivial task. It is necessary to discover the relevant interactions between *s-bots* that lead to the emergence of the global organization. In other words, the challenge is given by the necessity to decompose the global behavior that result in the desired organization in simple mechanisms and simple interactions among the system components. Furthermore, even if we know the mechanisms that lead to the emergence of the global organization, we still have to consider the problem of encoding them into the controller of each *s-bot*. As we already mentioned, in doing this, the

environment in which the *s-bots* are embedded must be taken into account because of its importance on the dynamics of the system and its role as communication medium, allowing different forms of stigmergy. Figure 2.1 exemplifies this process. The self-organized system displays a global behavior interacting with the environment (Figure 2.1, left). In order to define the controller for the *s-bots*, it is necessary to first decompose the global behavior into individual behaviors and local interaction among *s-bots* and between *s-bots* and environment (center). Then, the individual behaviors must be encoded into the control program that drives each *s-bot* (right).

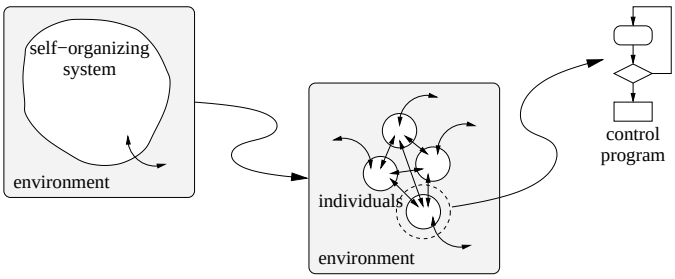


Figure 2.1: A representation of the design problem. In order to have the *swarm-bot* self-organize, we should first decompose the global behavior of the system (left) into individual behaviors and local interactions among *s-bots* and between *s-bots* and environment (center). Then, the individual behavior must be in some way encoded into a control program (right).

Summarizing, from an engineering perspective, the design problem is generally decomposed into two different phases: (i) the behavior of the system should be described as the result of interactions among individual behaviors, and (ii) the individual behaviors must be encoded into controllers. Both phases are complex because they attempt to decompose a process (the global behavior or the individual one) that emerges from a dynamical interaction among its subcomponents (interactions among individuals or between individual actions and environment).

Nolfi and Floreano [37] claim that, since the individual behavior is the emergent result of the interaction between agent and environment, it is difficult to predict which behavior results from a given set of rules, and which are the rules that will create a given behavior. Similar difficulties occurs in the decomposition of the organized behavior of the whole system into interactions among individual behaviors of the system components. Here, the understanding of the mechanisms that lead to the emergence of self-organization must take into account the dynamic interactions among indi-

vidual components of the system and between components and environment. Thus, it is difficult to predict, given a set of individual behaviors, which behavior at the system level will emerge, and it is also difficult to decompose the emergence of a desired global behavior in simple interactions among individuals.

The decomposition from the global to the individual behaviors could be simplified taking inspiration from natural systems, like insect societies, that could show us the basic mechanisms to be exploited. This approach finds its roots in the recent studies on *swarm intelligence* [7]. Starting from the animal society metaphor, swarm intelligence has emerged as a novel approach to the design of "intelligent" systems inspired by the efficiency and robustness observed in social insects in performing global tasks. The swarm-intelligent approach to the design problem starts from the observation of a natural phenomenon, followed by a *modeling* phase, which is of fundamental importance to "uncover what actually happens in the natural system" ([7], page 8). The developed model can then be used as a source of inspiration for the designer, who can try to replicate certain discovered mechanisms into the artificial system, in order to obtain dynamics similar to the natural counterpart. As exemplified in Figure 2.2, this approach requires a first decomposition step that models the phenomena observed in nature to find out which are the basic mechanisms and individual interactions. This knowledge is than exploited in the design phase, where these mechanisms are encoded into the control program.

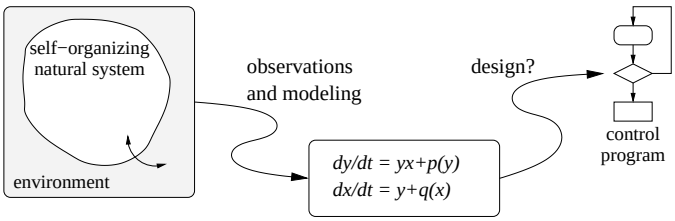


Figure 2.2: The swarm-intelligent approach to the design problem: a natural self-organizing system (left) can be observed and its global behavior modeled (center), obtaining useful insights on the mechanisms underlying the self-organization process. The model can be used as a source of inspiration for the following design phase, which leads to the definition of the control program (right).

However, it is not always possible to take inspiration from natural processes because they may differ from the artificial systems in many important

aspects (e.g., the physical embodiment, the type of possible interactions between individuals and so forth), or because there are no natural systems that can be compared to the artificial one. Moreover, the problem of encoding the individual behaviors into a controller for the *s-bots* remains to be solved. Our working hypothesis is that these problems can be efficiently solved relying on Artificial Evolution[37], as discussed in the next section.

2.4 Artificial Evolution of Group Behaviors

In this section, we will motivate why Artificial Evolution can efficiently solve the design problem. In fact, artificial evolution eliminates the problem of decomposition at both the level of finding the mechanisms that lead to the emergent global behavior, and the level of implementing those mechanisms into a controller for the *s-bots*. Artificial evolution relies on the evaluation of the system as a whole, that is, on the emergence of the desired global behavior starting from the definition of the individual ones. This approach is exemplified in Figure 2.3: the controller encoded into each genotype is directly evaluated looking at the resulting global behavior. The evolutionary process is responsible of selecting the "good" behaviors and discarding the "bad" ones. Moreover, the controllers are directly tested in the environment, thus they can exploit the richness of solutions offered by the dynamic interactions among *s-bots* and between *s-bots* and environment, which are normally difficult to be exploited by hand design.

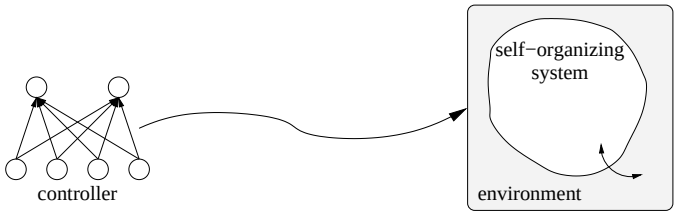


Figure 2.3: The evolutionary approach to the design problem: controllers (left) are evaluated for their capability to produce the desired group behavior (right). The evolutionary process is responsible for the selection of the controllers, testing them in the environment where they should work.

It is worth noting that, while the hand design normally proceeds in a top-down direction, following a *divide and conquer* approach, the evolutionary process proceed in the bottom-up direction, directly evaluating controllers for their suitability to the requirements defined by the designer. Artificial

Evolution does not need any arbitrary decomposition of the problem into sub-problems, but it relies on the automatic process of selection and reproduction (which is often referred to be self-organized itself).

The main problem of the divide and conquer approach is well explained by Nolfi and Floreano [37]. The decomposition of a global behavior into sub-components is often performed from a *distal* description of the behavior, that is, a description from the observer point of view. On the other hand, the control rules correspond to a *proximal* description of the behavior, that is, a description of the coupling of sensory (and internal) states to motor actions. The distal description of the behavior is a result of the agent-environment interactions, and therefore it may be impractical to define the controller at the proximal level. The divide and conquer approach may fail when, following the distal description, the global behavior is arbitrarily decomposed in sub-parts that does not have a one-to-one mapping with the sub-components of the control system. On the contrary, the evolutionary approach can overcome this problem defining a controller at the proximal description level, while testing and evaluating it at the distal level. In this way, no arbitrary choice is performed by the designer, but the process is left free to choose and test any possible solution that can produce the desired global behavior.

The application of the evolutionary approach to group behaviors has already been successfully applied in previous work, where we evolved an aggregation behavior for a group of *s-bots* [50]. We observed that evolution is able to find simple but efficient solutions to the aggregation task, producing self-organizing behaviors similar to those observed in natural systems, like the one shortly described in Section 2.2.

Before concluding, it is worth mentioning that the advantages offered by Artificial Evolution are not costless. On the one hand, it is necessary to identify initial conditions that assure *evolvability*, i.e., the possibility to progressively synthesize better solutions starting from scratch. On the other hand, artificial evolution may require long computation time and it is often unfeasible to apply it on real robots. For this reason, software simulations are often used. The simulations must save as much as possible the interesting features of the robot-environment interaction. Therefore, we have chosen to develop our simulations using a rigid body dynamics simulator, which can accurately simulate the dynamics and collisions of our *s-bots* in a 3-D environment. In Chapter 3, we will describe the setup of our experiments, introducing the simulated model of the *s-bot* and describing the evolutionary algorithm we used for the synthesis of self-organizing behaviors for the *swarm-bot*.

Chapter 3

Experimental Setup

This chapter is dedicated to the description of the experimental setup. It is thus introductory to the following chapters, in which the experiments we performed are described in detail. In particular, we will introduce the *s-bot* hardware and the simulated model in Section 3.1. In Section 3.2, the sensor and actuator configuration used throughout all the experiments is detailed. Finally, in Section 3.3, we describe the evolutionary algorithm we used in order to evolve coordinated movement behaviors.

3.1 The *S-bot* Model

In this section, we describe the *s-bot* hardware and the simplified simulation model we devised in order to run the evolutionary experiments. Figure 3.1 shows the first *s-bot* prototype that has been produced[1]. The mobility of the *s-bot* is ensured by a combination of two tracks and two wheels, called *Differential Treels*© *Drive* which are mounted on a chassis containing motors and batteries. Each track is connected to the wheel of the same side and it is controlled by an independent motor. This particular combination of tracks and wheels allows an efficient rotation on the spot due to the position of the wheels and makes navigation simpler on moderately rough terrains, while more complex situations can be tackled by a *swarm-bot*. The chassis can rotate with respect to the main body (turret) by means of a motorized axis. This ensures an independent movement of the turret where the sensors and the grippers for physical connections to other *s-bots* or objects are located.

[1]Details regarding the hardware and simulation of the *swarm-bot* can also be found in [40] and in the project web-site (`http://www.swarm-bots.org`).

The *s-bots* have two different way of creating physical interconnections to self-assemble into a *swarm-bot* configuration: rigid and semi-flexible. Rigid connections between *s-bots* are implemented by a gripper mounted on the *s-bot* turret. This gripper can also be used to lift another *s-bot* if necessary. Semi-flexible connections are implemented by a flexible arm actuated by three servo-motors mounted on the turret. These three degrees of freedom allow to extend and move laterally and vertically the arm. The *s-bot* grippers can grasp other *s-bots* on a T-shaped ring around the *s-bot* turret.

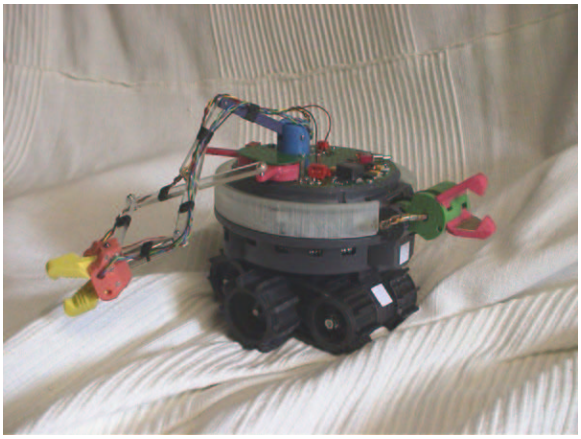


Figure 3.1: The first *s-bot* prototype, provided of the tracks system, the body holding the rigid and the flexible grippers, and many sensor systems.

Given the hardware implementation, we have defined a simple *s-bot* model in order to develop fast simulations, which could preserve the features of the real *s-bot* we are interested in (see Figure 3.2). For this purpose, we relied on the Vortex^{TM} SDK, which offers the necessary functionalities to develop accurate 3-D dynamic simulators. The *s-bot* turret is modeled as a cylinder (radius: $6\,cm$, height $6\,cm$), connected to the chassis by a motorized hinge joint. The chassis is a sphere (radius: $1.4\,cm$) to which 4 spherical wheels are connected (radius: $1.5\,cm$), two lateral and two passive wheels in the front and in the back, which serves as support. The lateral wheels are connected to the chassis by a motorized joint and a suspension system, thus they are responsible for the motion of the *s-bot*. In this way, a differential drive mechanism is implemented, modeling the external wheels of the physical realization. On the contrary, the other wheels are not present in the real *s-bot*, which is provided of tracks instead. These wheels model the balancing role of tracks, but, being not motorized, they do not contribute to the

motion of the *s-bot*. Connections between *s-bots* are simulated dynamically creating a joint between the two bodies. However, in this work, *s-bots* are always assembled into a *swarm-bot*, thus we do not consider the problem of dynamically creating connections.

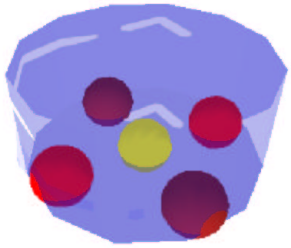


Figure 3.2: The simulated *s-bot* model. The body is transparent to show the chassis (center sphere), the motorized wheels (lighter spherical wheels) and the passive wheels (darker spherical wheels). The position of the virtual gripper is shown with an arrow painted on the *s-bot*'s body. On the contrary, the front direction of the chassis is not shown. In the following, we will display the direction of forward motion drawing a cone in place of the spherical chassis

The *s-bot* model is thus simple enough to obtain fast simulations. Wheels are modeled as spheres and not as cylinders in order to simplify both the collisions detection between the wheels and the ground, and the computation of the dynamics of the different bodies. The chassis, having no other functionality than connecting the different parts of the *s-bot*, is modeled as a sphere, which is the simplest object to be simulated. The *s-bot* turret on the contrary is modeled as a cylinder, the simplest shape close to the real *s-bot*. This allows to simulate in a realistic way collision among *s-bots* and between *s-bots* and walls or obstacles. On the contrary, the computation of collisions involving wheels, chassis, walls and obstacles are all disabled, as these objects cannot collide, thus improving the performance of the simulator.

3.2 Sensor and Actuator Configuration

The hardware realization of an *s-bot* includes many sensor systems, among which infrared proximity sensors, light sensors, directional microphones and

an omni-directional camera. Even if these sensors are provided by the developed simulator, they are not used in this work. On the contrary, we provide each *s-bot* with a *traction sensor*, placed at the turret-chassis junction and able to detect the direction and the intensity of the traction force that the turret exerts on the chassis. This particular kind of sensor proved to be of fundamental importance for coordinated movement tasks [6], and it is at the moment of writing under development on the real *s-bot*.

The working principle of the traction sensors is very simple (see Figure 3.3): when *s-bots* are connected in a *swarm-bot* configuration, their movement can produce pulling/pushing forces on other *s-bots*. In particular, the turret of each *s-bot* physically integrates the forces that are applied to the *s-bot* by the other *s-bots*, and takes also into account the movements of the *s-bot*'s chassis. As a consequence, the traction sensor provides the *s-bots* with an indication of the average direction toward which the *swarm-bot* is trying to move as a whole. More precisely, it measures the mismatch between the directions toward which the entire team and the *s-bot* chassis are trying to move. The intensity of the traction is also an indication of the size of this mismatch.

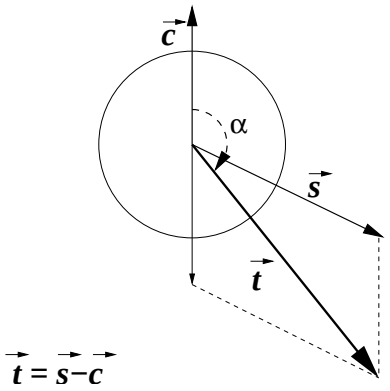
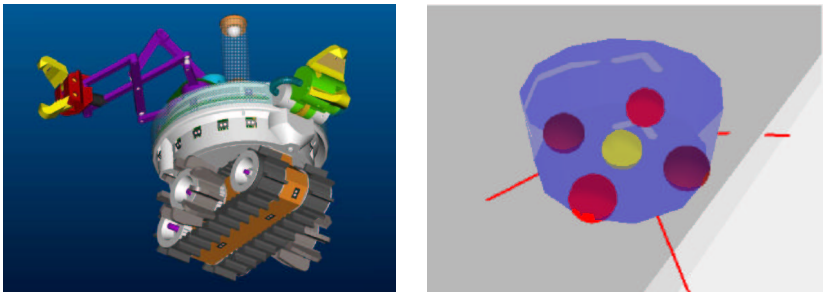


Figure 3.3: Traction sensor working principle: the traction $\vec{t}$ is the resultant of the vectorial summation of all the forces applied by other *s-bots* on the turret ($\vec{s}$) minus the force exerted by the movement of the chassis ($\vec{c}$). α is the resulting direction of the traction, while $|\vec{t}|$ is the resulting intensity.

Besides traction sensors, in this work we also make use of *ground sensors*, which are merely proximity sensors pointed to the ground. The use of these sensors is justified by the fact that, when exploring an unknown environment, the *s-bots* should be able to recognize the presence of hazards like holes or troughs and behave in order to avoid to fall into them. Also,

these sensors can be used as a measure of the roughness of the terrain on which they are moving, and used to select different strategies of exploration. The real *s-bot* is provided with 4 ground sensors, positioned along the axis of the chassis, as can be observed in Figure 3.4a. In the simulated *s-bot* we tried a different configuration that we consider to be more useful to avoid hazards. In this case, the 4 ground sensors are evenly distributed around the chassis of the *s-bot*, starting at 45 degrees from the direction of movement (see Figure 3.4b). This sensor distribution will be implemented in the hardware exploiting some of the proximity senors present on the turret.



(a) (b)

Figure 3.4: Ground sensors. (a) a graphical representation of the hardware implementation, where ground sensors are positioned along the axis of the chassis. (b) The simulated ground sensors, displayed as lines exiting from the *s-bot* are positioned around the body in order to better recognize the direction of hazards like holes in the ground.

Concerning the actuators, each *s-bot* can control its wheels independently. In order to do so, the control system can specify a desired angular speed to be reached and a maximum torque to be applied by the motor controlling the lateral wheels. The maximum speed values has been set to $10\, rad/s$, which correspond to a maximum speed of the *s-bot* of $0.15\, m/s$. The maximum torque to be applied is set to $0.2\, Nm$. In addition to the wheels, the movements of the *s-bot* are also influenced by the motor controlling the rotation of the turret with respect to the chassis. However, it is not independently controlled with respect to the wheels, its desired angular speed ω_t being defined as

$$\omega_t = \frac{\omega_l - \omega_w}{2}, \tag{3.1}$$

where ω_l and ω_r are the desired angular speed of the left and right wheel respectively. The maximum speed and torque values are the same as for the

wheel motor. This way of controlling the rotation of the turret is designed in order to help the re-orientation of the tracks when, due to the roughness of the terrain or to pulls or pushes coming from other *s-bots*, one or both wheels do not touch the ground. In these cases, the torque applied to the turret motor will result in a rotation equivalent to what would result from the action of the wheels.

As already mentioned before, we are interested in evolving behaviors for coordinated movement in an already assembled *swarm-bot*, while the problem of self-assembling will be considered in future research. Thus, the gripper actuators are not used in this context, as *s-bots* are always connected to each other. The connections are rigid and do not allow relative motion of the *s-bots*. In some test experiments, we made connections flexible to study the effect of relative position change on the evolved controllers.

3.3 The Evolutionary Algorithm

In this section, we describe the evolutionary algorithm used to evolve controllers for coordinated motion. The details of the evolved controller and of the fitness function may change from experiment to experiments, and they will be described later.

We use a generational evolutionary algorithm for the evolution of the *s-bot* neural controller. The initial population is composed by μ randomly generated genotypes. Each genotype is binary encoded, and can be mapped into a controller for a single *s-bot*. The length L of the genotype is fixed and depends on the controller that is evolved. This controller is cloned in each of the n *s-bots* involved in the experiment [5]. The fitness F of each genotype is estimated allowing the group of *s-bots* to "live" for M "epochs" and then averaging the obtained value:

$$F = \sum_{e=1}^{M} F_e, \tag{3.2}$$

where F_e is the fitness estimation obtained from a single epoch. The value F_e depends on the evolved behavior. Each epoch e lasts a maximum of T simulation cycles, each cycle corresponding to 100 ms of real time.

The best λ genotypes of each generation are allowed to reproduce, each generating μ/λ offspring[2]. Each bit has a probability p/L of being replaced by the opposite value. The reproduction is asexual, as no recombination is

[2]For sake of simplicity, we use λ values that are divisors of μ values.

performed. Furthermore, parents are not copied to the offspring population (no elitism). The evolutionary process is stopped after a maximum value of N generations.

This algorithm is very simple and straight forward, but we found that is enough to evolve simple but efficient controllers for group of robots [50, 5, 6]. Table 3.1 summarizes the parameters that have to be set for defining an evolutionary run, and it gives also some default values that are common to all the experiments presented here.

Table 3.1: The parameters of the evolutionary algorithm. Default values common to all the experiments performed in this work are also shown.

Parameter	Explanation	Default
μ	Population Size	100
L	Length of the genotype (bits)	—
n	Number of *s-bots* involved in the experiment	4
M	Number of epochs per fitness estimation	5
T	The duration of a single epoch e (simulation cycles)	—
λ	The number of parents allowed to reproduce in order to build the population of the following generation	20
p	The average number of bits mutated in a genotype, resulting from a probability p/L of mutation per bit	2
N	Maximum number of generations	100

Chapter 4

Evolving Coordinated Motion

Coordinated motion is a basic capability that should be provided to a *swarm-bot*. In fact, when assembled into a *swarm-bot*, *s-bots* partly loose their autonomy, as they are physically connected to other *s-bots*. Nonetheless, the autonomy in motion of the *swarm-bot* as a whole must be preserved. This implies that *s-bots* should coordinate in order to move as a single being, collectively avoiding obstacles or aiming to a particular location.

The problem of coordinated motion in a *swarm-bot*, which is detailed in Section 4.1, has been studied by Baldassarre et al. [6], and was efficiently solved using Artificial Evolution. In this chapter, we present the results obtained replicating the experiments presented in [6], adapting them to the simulation model presented in Section 3.1. The experimental setup of these experiments is described in Section 4.2. We show that we are able to obtain comparable results and similar generalization features of the evolved controller (see Section 4.3 and 4.4). The experiments presented in this chapter can be considered as the starting point for the solution of the hole avoidance problem, which is presented in Chapter 5.

4.1 The Coordinated Motion Task

Generally, the first problem to be faced when trying to control an autonomous robot is how to make it move efficiently in a given environment. Depending on the robot, this task can be very simple or incredibly complex. For example, a wheeled robot can be easily controlled by setting the speed of its wheels. On the contrary, the motion of a humanoid robot is still an

open problem in the robotic community. Also the environment in which the robot is placed may influence the complexity of the motion: a flat terrain without obstacles does not create many problems, while a rough terrain with holes and obstacles is clearly more challenging. A different source of complexity is present in the coordinated motion task, where the robotic system is composed of a number of independent entities that have to coordinate their actions in order to move coherently.

Coordinated motion is a well studied behavior in biology, being observed in many different species. For example, we can think of flocks of starlings coordinately flying or to schools of Atlantic cods swimming in perfect unison. These examples are not only fascinating for the charming patterns they create, but they also represent interesting instances of self-organizing behaviors. Many researchers have provided models for schooling behaviors of fish, and replicated them in artificial life simulations [9]. These model explain the coordinated movement only with simple attraction and repulsion rules between individuals (positive and negative feedback), which are based only on local information about the position and heading of neighboring fish. Similarly, the behavior of groups of artificial fish (called *e-boids*) has been evolved to display schooling behaviors, obtaining interesting results [52]. Finally, evolutionary computation has been used also to evolve coordinated motion behaviors in group of physical robots [41, 42]. In this case, groups of 2 and 3 robots where asked to move as far as possible from their starting location, and the results showed the emergence of coordinated motion, notwithstanding the limited sensing abilities of the robots.

In this work, coordinated motion is performed in a group of physically linked *s-bots*. This additional constraint limits the individual abilities of each *s-bot* in the group, because physical connections prevent most of the possible movements. Furthermore, the motion of one *s-bot* can influence the dynamics of the whole *swarm-bot* because physical links transmit the forces applied by one *s-bot* to the rest of the group. However, the *s-bots* can orient their rotating chassis toward a common direction and move in a coherent way. Therefore, a very precise coordination in the orientation of the rotating bases is necessary, because even a small difference can affect the performance in the motion of the *swarm-bot*.

4.2 Experimental Setup

The experiment performed in this work replicates those presented in [6]. Here, the *swarm-bot* is formed by 4 assembled *s-bots* in a linear configuration,

as shown in figure 4.1. Physical connections are rigid, so that no relative movement is allowed between *s-bots*. Initially, the chassis of each *s-bot* is randomly oriented. In this way, the *s-bots* have to solve the problem of collectively choosing a common direction where to move and then have to cover the maximum possible distance.

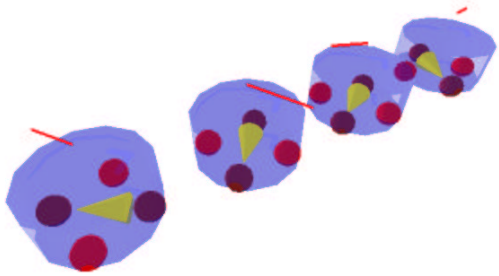


Figure 4.1: The initial configuration for the evolution of coordinated movement behaviors. The *swarm-bot* is composed of 4 *s-bots* linearly connected (physical links are not drawn). The red line on top of each *s-bot* indicates the intensity and direction of the traction felt by each *s-bot*.

4.2.1 Controller Setup

In these experiments, the *s-bot* is equipped with a traction sensor that provides compact information about the average direction where the *swarm-bot* in trying to move (see Section 3.2). The information provided by the traction sensor refers to both intensity and direction of traction. We encoded these two values in 4 variables, in order to limit the discontinuities of the information on the direction of traction, which can directly pass from $-\pi$ to π or the other way round, and may cause problem at the control level. These four variables v_i encode the intensity of the traction from four different preferential orientations with respect to the chassis $\theta_i = i \cdot \pi/2$, having $i \in \{0, 1, 2, 3\}$. In particular, for each preferential orientation θ_i, this intensity decreases linearly with respect to the absolute difference between the sensor's preferential orientation and the direction of traction, and is 0 when

this difference is bigger than $\pi/2$:

$$v_i = \begin{cases} I \cdot \left(1 - \frac{|\alpha - \theta_i|}{\pi/2}\right) & \text{if } |\alpha - \theta_i| \leq \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases}, \qquad i \in \{0, 1, 2, 3\}, \tag{4.1}$$

where α is the direction of the traction and I is its intensity, linearly scaled in the interval $[0, 1]$.

Each *s-bot* controller is a neural network with 4 sensory neurons that receive the variables v_i, plus one bias neuron. These are directly connected with 2 motor neurons. The activation state of the motor units is normalized between $[-10, +10]$ and used to set the desired angular speed of the two corresponding wheels and the turret-chassis motor, as described in Section 3.2.

The connection weights of the neural controller of the *s-bots* are evolved following the algorithm described in Section 3.3. Each connection weight is represented in the genotype by 8 bits that are transformed into a number in the interval $[-10, +10]$. Therefore, the total length of the genotype is $10 \times 8 = 80$ bits. The other parameters of the algorithm are set to their default value (see Table 3.1).

4.2.2 Fitness Estimation

To allow the *swarm-bot* to move as fast and as straight as possible, we devised a fitness estimation F_e based on the Euclidean distance between the center of mass of the team at the beginning and at the end of each epoch:

$$F_e = \frac{\|\mathbf{X}(0) - \mathbf{X}(T)\|}{\mathrm{D}}, \tag{4.2}$$

$$\mathbf{X}(t) = \frac{1}{n} \sum_{j=1}^{n} \mathbf{X}_j(t), \tag{4.3}$$

where n is the number of *s-bots* involved in the experiment, $\mathbf{X}_j(t)$ is the coordinates vector of the j^{th} *s-bot* at cycle t, $\mathbf{X}(t)$ is the resulting coordinates vector of the center of mass of the group, and D is the maximum distance that a single *s-bot* can cover in T cycles by moving straight at maximum speed. The duration T of each epoch is fixed to 150 simulation cycles, and corresponds to 15 seconds of real time.

In Table 4.1, we summarizes the parameters of the evolutionary algorithm specific to the coordinated motion task. Along with the definition of the fitness estimation F_e, they complete the description of the evolutionary algorithm given in Section 3.3.

Table 4.1: Parameters of the evolutionary algorithm specific to the coordinated motion experiments.

Parameter	Explanation	Value
L	Length of the genotype (bits)	80
T	The duration of a single epoch e (simulation cycles)	150

4.3 Results

The experiment were replicated 10 times, starting with different randomly generated populations. The average fitness over the 10 replications is shown in Figure 4.2. The plot indicates that the evolutionary experiment was successful, as a very good performance was achieved in all replications.

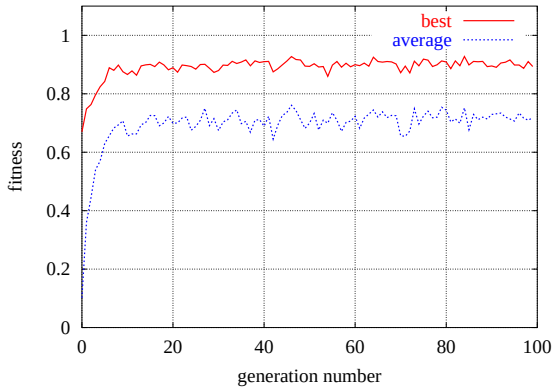


Figure 4.2: Average fitness over the 10 replications of the experiment for the evolution of coordinated motion. The best and the average fitness of the population are plotted versus the number of generations.

We tested the best controllers produced in each replication, evaluating them for 100 epochs. The average fitness values are presented in Table 4.2. It shows that most of the evolved controllers display a satisfactory average performance.

The behaviors obtained in each replication present many similarities. Direct observation has shown that at the beginning of each epoch, the *s-bots* start to move in the direction they were positioned, but the physical connections transform this disordered motion into traction forces, which are exploited to coordinate the group. When an *s-bot* feels a traction force, it

Table 4.2: Average performance of the best controller evolved in each replication of the experiment.

Replication	Performance
1	0.87888
2	0.83959
3	0.88338
4	0.71567
5	0.79573
6	0.75209
7	0.83425
8	0.85848
9	0.87222
10	0.76111

rotates its chassis in order to cancel this force. Once the chassis of all the *s-bots* are oriented toward the same direction, the traction forces disappear and the coordinated motion of the *swarm-bot* starts (see Figure 4.3). This is possible because, as mentioned in Section 3.2, the traction sensor gives an indication of the mismatch between the direction of the chassis of the *s-bot* and the average direction of motion of the *swarm-bot*. Thus, a high value returned by the traction sensor corresponds to high mismatch, and results in a fast rotation of the chassis in order to compensate this mismatch.

An example of the coordination activity is given in Figure 4.4, showing

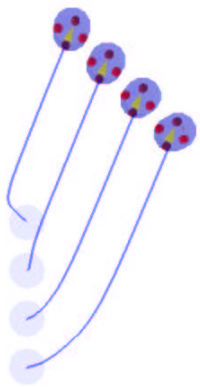


Figure 4.3: Trajectories drawn by a *swarm-bot* during coordinate motion.

the dynamics of the coordination. Figure 4.4a plots the angular distance between the chassis of each *s-bot* and the average orientation of all the chassis in the *swarm-bot*, scaled in the interval $[0, 1]$. It shows that after a transitory period, the *s-bots* converge toward a common orientation. Similarly, Figure 4.4b plots the traction intensity felt by each *s-bot*, also scaled in the interval $[0, 1]$. Also in this case, a transitory period is followed by a stable state in which the traction intensity is zero, indicating that the coordination has been achieved. It is worth noting the similarities between the two graphs, which confirms that an high traction correspond to an high mismatch in the direction of the chassis and triggers a fast reaction, as explained above. Figure 4.4 also highlights how the reaction of an *s-bot* to the traction intensity and angle variation presents complex dynamics that have been exploited by the evolutionary algorithm to synthesize an efficient controller.

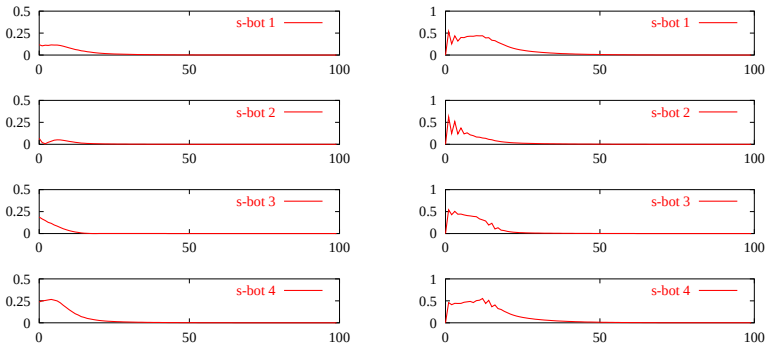


Figure 4.4: Coordination dynamics. (a) Angular distance between the chassis of each *s-bot* and the average orientation of the chassis in the *swarm-bot*. The average orientation is normalized between the values [0,1], and is plotted against the simulations cycles. (b) the traction intensity felt by each *s-bot*, scaled in the interval $[0, 1]$ and plotted against the simulation cycles elapsed.

The obtained results are qualitatively similar to those presented in [6]. Nonetheless, a quantitative difference is present in the performance, due to the different simulation models used. These differences result in different dynamics of the *swarm-bot* with respect to similar initial conditions in the two cases. This also confirms the importance of the embodiment in the design of the controllers.

4.4 Generalization Properties

The evolved strategy for coordinated motion is very robust, being able to work in many different settings. For example, this strategy scales very well with the number of *s-bots* forming in the chain: also in this case, coordinated motion emerges after a transitory phase in which the *s-bots* collectively choose a common direction. This generalization property can be explained by the fact that the traction sensor still integrates the forces applied on the turret by other *s-bots*. However, the transmission of forces is less efficient with increasing chain size and causes a longer duration of the coordination phase.

The evolved strategy generalizes well also to different shapes of the *swarm-bot*. Figure 4.5 shows the trajectories drawn by each *s-bot* in a *swarm-bot* having a star formation. This reveals that the information coming from the traction sensor has the same property, and the evolved behavior is able to exploit it, no matter the configuration of the *swarm-bot*.

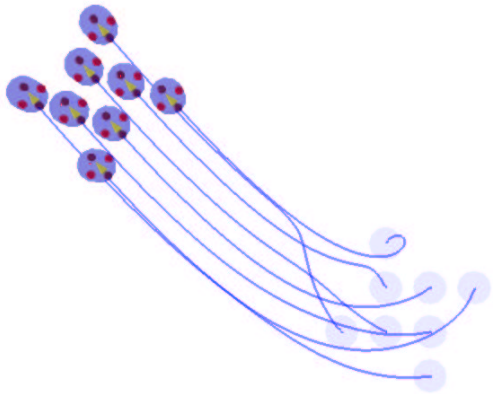


Figure 4.5: Generalization of the coordinated motion behavior to a different number of *s-bots* and a different shape.

The coordinated motion behavior displays another interesting feature: it is able to perform collective obstacle avoidance. When an *s-bot* hits an obstacle with its body, the turret exerts a force on the chassis in a direction opposite to the obstacle. This force is felt as a traction pulling the *s-bot* away from the obstacle. In response to this traction, the *s-bot* rotates its chassis in order to cancel the traction, as explained above. Moreover, the rigid connections between *s-bots* transmit the force resulting from the collision to the whole group, which triggers a fast change in the direction of movement

of the whole group. This behavior is shown in Figure 4.6, where 4 *s-bots* forming a chain move in a square arena and, once they hit a wall, collectively change direction of motion, without remaining stuck against the wall. It is worth noting that the traction sensor works as an omni-directional bumper distributed on the whole body of the *swarm-bot*, allowing collective obstacle avoidance.

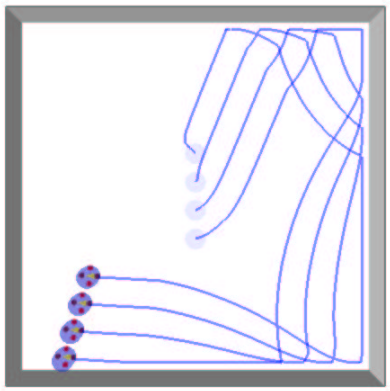


Figure 4.6: Generalization of the coordinated motion behavior to obstacle avoidance. Here, the traction sensor works as a distributed, omni-directional bumper.

Finally, we tested the evolved behaviors using flexible connections between *s-bots*, allowing relative motion between connected *s-bots*. Flexible connections allow the connecting *s-bot* to rotate around the body of the connected *s-bot*, without changing the distance. Using this type of connections, the shape of the *swarm-bot* can change during motion, and traction is transmitted by means of the connection only to some extent. Nevertheless, the evolved strategy still works. Figure 4.7 shows the case of a chain of 8 *s-bots* with flexible connections, placed in a squared arena with cylindrical obstacles. The initial coordination phase makes the chain deform, but after a while the *swarm-bot* displays coordinated motion. Also the collision with an obstacle leads to a change in the shape, which enable the *swarm-bot* to pass through narrow passages and restart the coordinated motion afterward.

In conclusion, we have shown that the evolved strategy displays very robust behavior, which is able to cope with both changes in the number of *s-bots* forming the *swarm-bot*, and variations of the shape of the *swarm-bot*. Also environmental changes, like the presence of walls or obstacles, do not degrade the performance of the evolved controllers, which are able to

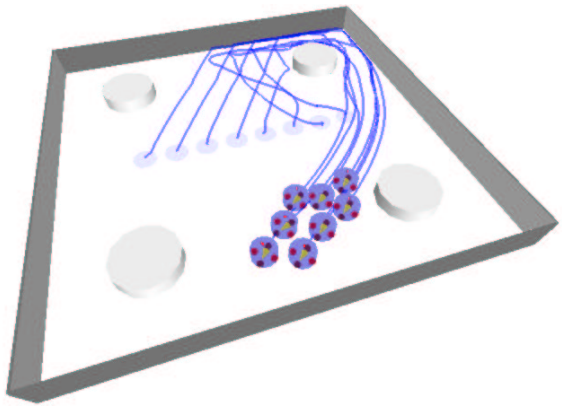


Figure 4.7: Generalization of the coordinated motion behavior to the introduction of flexible links between robots. The *swarm-bot* is able to change shape and efficiently avoid obstacles.

display a collective obstacle avoidance behavior. Artificial evolution was able to exploit the complex group dynamics arising from the physical interactions among *s-bots*, producing a behavior that presents the typical features of self-organization. In the following chapter, we will show how similar features can be produced also for the hole avoidance task.

Chapter 5

Evolving Hole Avoidance

In the previous chapter we presented the coordinated motion task, and we showed how evolution can synthesize simple but efficient and robust controllers for the *s-bots*. In this chapter, we face a similar problem, that is, coordinated motion in an environment that presents hazards like holes or troughs, that have to be avoided by the *swarm-bot*. This task is presented in Section 5.1. The experimental setup for the evolution of hole avoidance behaviors is detailed in Section 5.2. Finally, Section 5.3 presents the results obtained using artificial evolution for different types of neural controllers.

5.1 The Hole Avoidance Task

The hole avoidance task can be considered as an instance of the family of "navigation on rough terrain" tasks. The ability to cope with rough terrains, holes, gaps or narrow passages is a very important feature for an intelligent robotic system, that can open many possible application scenarios, like rescue in a collapsed building or space exploration. Research in this direction has focused mainly on the development of rovers provided with articulated wheels or tracks, like the *pathfinder* [48], well known for the big impact on mass media resulted from the mission on Mars. A different approach to rough terrain navigation is presented by reconfigurable robotics, where robots can adopt different shapes in order to cope with different environmental conditions [12, 49, 54].

In the *swarm-bot* case, navigation on rough terrain is achieved by means of the cooperation between *s-bots* which can self-assemble and build structures that can cope with hazardous situations like avoiding a hole or passing over a trough. In such cases, rigid connections serve as support for those

s-bots that are suspended over the gap. This approach to rough terrain navigation also as a natural counterpart in ants of the species *Œcophilla longinoda* [28], which are able to build chains connecting one to the other, creating bridges that facilitate the passage of other ants.

In this work, we study the problem of coordinated motion in an environment that presents holes too large to be traversed by a *swarm-bot*. Thus, holes must be recognized and avoided, so that the *swarm-bot* does not fall into them. The difficulty in this task lays in the fact that *s-bots*, having only limited sensing capabilities, cannot feel the presence of a hole until they are very near to its edge. Therefore, when not joining a *swarm-bot*, *s-bots* may fall into holes being unable to efficiently react, above all when edges present irregularities like convex angles. In the *swarm-bot* configuration, physical connections serve as support for those *s-bots* that are near an edge, making the *swarm-bot* able to safely react. However, it is necessary to communicate the presence of a hole to the whole group, which must consequently reorganize to choose a safer direction of motion.

5.2 Experimental Setup

In order to study the hole avoidance task, we designed an arena that presents holes with both concave and convex angles (see Figure 5.1). The arena is a square box (side 3 m), having 4 square holes (side 60 cm). The hole edges present convex angles, which are difficult to detect by an *s-bot*. Besides, the borders of the arena itself are hazards to be avoided and present concave angles.

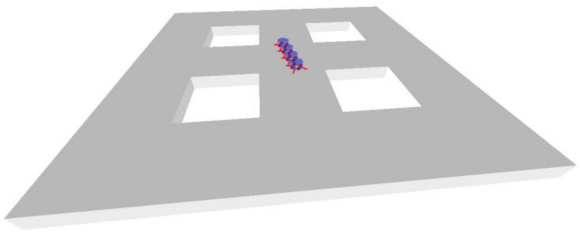


Figure 5.1: The arena employed for the hole avoidance task.

Also in this case, the *swarm-bot* consists of a linear structure made by 4 *s-bots*. The sizes of the arena and of the holes have been chosen in order to leave enough space for the passage of the *swarm-bot*, no matter the orienta-

tion of the chain. This can allow an efficient navigation in the arena once the *s-bots* are able to avoid holes.

5.2.1 Controller Setup

The *s-bots* are provided with a traction sensor, which returns direction and intensity of the traction exerted by the turret on the chassis. These values are encoded in 4 virtual sensors, as described in Section 4.2.1. Besides the traction sensors, an *s-bot* can exploit the information coming from 4 ground sensors positioned around the chassis of the *s-bot* (see Section 3.2). The value returned by each ground sensor is normalized in the interval $[0, 1]$ and passed to the neural controller. Thus, the neural network has 8 sensory inputs coming from traction and ground senors, and 2 motor outputs controlling the wheels and the turret-chassis motor.

We performed 3 different sets of experiments, each characterized by a different type of controller. In the first set, we used a simple *perceptron*, where the sensory units and a bias unit were directly connected to the motor units. The weights of the perceptron were evolved. Each weight, ranging in the interval $[-10, 10]$, was represented in the genotype by 8 bits, corresponding to a genotype length $L_1 = 18 \times 8 = 144$ bits.

In the other two sets of experiments, we used recurrent neural networks in order to test whether internal dynamics of the neural network could lead to an adaptive advantage in the hole avoidance task. In the second set of experiments, we employed an *Elman* architecture [18], characterized by a fully recurrent hidden layer made by 4 neurons, which are also fully connected to the input and output layers. In total, there are 62 connection weights to be evolved, which corresponds to a genotype length $L_2 = 62 \times 8 = 496$ bits.

In the last set of experiments we used a modified Elman architecture, called *dynamic* network. Here, each hidden neuron i has a time constant τ_i, so that the activation state H_i of the unit is computed by means of a moving average:

$$H_i(t+1) = \tau_i \cdot H_i(t) + (1 - \tau_i) \cdot A_i, \qquad \tau_i \in [0, 1] \tag{5.1}$$

where A_i is the sum of the activation coming from all the connecting neurons. Also time constants were under the control of the evolutionary algorithm, and were represented in the genotype by 8 bits. In this case, the length of the genotype is $L_3 = 66 \times 8 = 528$ bits. In all sets of experiments, the remaining parameters were set to their default values given in Table 3.1.

5.2.2 Fitness Estimation

In order to evolve hole avoidance behaviors, we devised a fitness function that favors coordinated motion, exploration of the arena and a fast reaction to the detection of the hole's borders. The fitness estimation F_e is given by the average of two components:

$$F_e = \frac{F_{e_1} + F_{e_1}}{2} \tag{5.2}$$

In order to compute the fitness components, we divide each epoch e into two sub-epochs, e_1 and e_2. In the former, we test the genotype for its ability to perform coordinated motion in a flat environment, having a fitness estimation F_{e_1} computed with Equation (4.2). This sub-epoch lasts $T_{e_1} = 150$ cycles. Here the *s-bots* start connected in a linear chain, having the orientation of their chassis randomly initialized, and having to learn to move coordinately.

In the latter sub-epoch, the fitness estimation F_{e_2} is given by

$$F_{e_2} = F_s \times F_x, \tag{5.3}$$

where F_s is a *survival* sub-component and F_x is an *exploration* sub-component. The survival sub-component F_s is designed to reward only those genotypes that reach the end of the epoch without falling into a hole. It is computed as follows:

$$F_s = \begin{cases} 1 & \text{if } T_s = T_{e_2} \\ 0 & \text{otherwise} \end{cases}, \tag{5.4}$$

where T_{e_2} is the length of the sub-epoch e_2 and T_s is the number of cycles the *swarm-bot* "survived" without falling into a hole. This sub-component penalizes every fall, even if it happens at the end of the sub-epoch, thus favoring more robust behaviors.

The second sub-component is designed in favor of those genotype that are able to explore the arena in depth. In this case, the arena is virtually divided in 25 squared zones of 60 cm side. The genotype is rewarded for the number of visited zones during the sub-epoch, as formalized as follows:

$$F_x = \frac{z(T_s)}{Z(T_{e_2})}, \tag{5.5}$$

where $z(t)$ is the number of visited zones at cycle t and $Z(t)$ corresponds to the maximum number of zones that can be visited in t cycles. This sub-component also has the side effect of favoring coordinated motion, because,

in order to explore, the *swarm-bot* must be able to efficiently move. However, without the component F_{e_1}, F_x is not sufficient to evolve efficient motion. Some tests we performed using only F_x as fitness function showed that the evolved behaviors exploited the shape of the arena, kept constant during the evolution. In fact, the *swarm-bot* learned to circle around one hole, without learning to avoid falling in different situations.

In sub-epoch e_2, *s-bots* are positioned at the center of the arena and start in the usual chain configuration, but their chassis are all initialized with the same random orientation. Also the chain is randomly oriented at the beginning of each sub-epoch. In this way, there is no need of a coordination phase at the beginning of the sub-epoch, the focus being put on hole avoidance. The sub-epoch lasts $T_{e_2} = 200$ cycles.

Table 5.1 summarizes the parameters of the evolutionary algorithm specific for the hole avoidance task. It also specifies the parameter specific for the different sets of experiments performed. Along with the definition of the fitness estimation F_e, these values complete the description of the evolutionary algorithm given in Section 3.3.

Table 5.1: Parameters of the evolutionary algorithm specific for the hole avoidance task and different sets of experiments.

Parameter	Explanation	Value		
		Perceptron	Elman	Dynamic
L	Length of the genotype (bits)	144	496	528
T_{e_1}	The duration of a single sub-epoch e_1 (simulation cycles)	150		
T_{e_2}	The duration of a single sub-epoch e_2 (simulation cycles)	200		

5.3 Results

In this section, we present the results obtained evolving hole avoidance behaviors using the three different controllers described above. For each controller, we replicated the evolutionary experiments 10 times. The average fitness values, computed over all the replications, are shown in Figure 5.2.

The average performance of the best individual and of the population are plotted against the generation number. All different neural architectures perform well, reaching a high fitness value. There is no clear difference among the plot of the different architectures, except for the fact that the perceptron evolves faster than the other two networks.

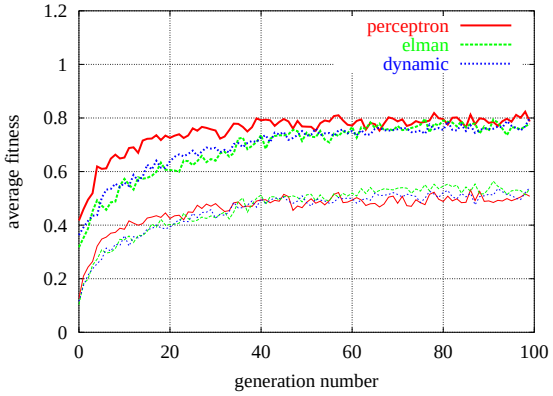


Figure 5.2: Average fitness over 10 replications of the experiment.

In order to test the performance of the evolved controllers, we evaluated the best individuals of the last generation of each replication of the experiments. The corresponding results are shown in Table 5.2. It can be noted that the average performance of every controller is significantly lower than the average values achieved during the evolutionary runs. This is due to a super-estimation of the individual performance related to random initialization and a small sampling size (5 epochs per fitness estimation). This fact, however, does not disturb the evolutionary process, because at each generation all the individuals are evaluated with the same random initializations, thus ensuring a fair comparison between individuals.

Table 5.2 also highlights the best controllers evolved for each of the three different neural architectures. From these data, it is possible to see that the best evolved controller is the dynamic network obtained in the 6^{th} replication, performing better than the best controllers of both the perceptron and the Elman architectures (evolved, respectively, by the 7^{th} and 6^{th} replication). Performing a Wilcoxon signed rank test with continuity correction, the dynamic neural network resulted statistically better (p-value of 0.007511 in the dynamic/perceptron test, 0.002356 in the Dynamic/Elman test).

It is worth noting that the dynamic architecture presents some replication with a low fitness. In particular, the 10^{th} replication did not end with

Table 5.2: Mean performance of the best individuals of each replication of the experiments, averaged over 100 epochs. The best evolved individuals of each neural network architecture are highlighted in bold.

Replication	Performance		
	Perceptron	Elman	Dynamic
1	0.6640	0.6564	0.6870
2	0.6541	0.6715	0.5696
3	0.6502	0.6257	0.6701
4	0.6079	0.6241	0.6075
5	0.5835	0.5951	0.5297
6	0.6376	**0.6894**	**0.7287**
7	**0.6866**	0.5942	0.6564
8	0.6397	0.6592	0.6005
9	0.6640	0.5798	0.6935
10	0.6458	0.6500	0.3913

an efficient solution. This can be explained by the fact that the search space is big and evolution may require more generations to find a suitable solution.

Direct observation of the behaviors evolved showed that all efficient solutions rely on similar strategies. Coordinated motion is achieved in the same way as described in Section 4.3. Concerning hole avoidance, when one *s-bot* detects an edge, it rotates the chassis and changes the direction of motion in order to avoid falling. This change in direction is felt by the other *s-bots* by means of the traction sensors, and triggers a coordination phase that ends up in a new direction of motion away from the edge. A key role in the functioning of this strategy is played by the motor controlling the rotation of the chassis with respect to the turret of an *s-bot*. In fact, this motor has a stabilizing effect on the rotation of the chassis even if one of the wheels is suspended on the edge. This gives the chance of changing its direction of motion to an *s-bot*, even when partially suspended and, consequently, it can cause a traction force that can be felt by the other *s-bots*. If the turret-chassis motor were not provided any rotation of the wheel touching the ground would cause a rotation of the chassis and consequently the loss of contact with the ground, loosing the possibility to influence the behavior of other *s-bots*. Figure 5.3 shows the trajectory displayed by a *swarm-bot* performing a hole avoidance task.

This kind of strategy may fail mainly for two reasons: the inertia of the

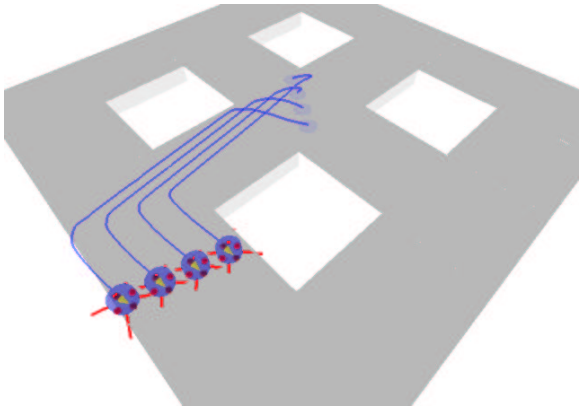


Figure 5.3: Trajectories displayed by a *swarm-bot* performing a hole avoidance task. It can be seen that, during the last turn, one *s-bot* was partially suspended, as the trajectory goes out from the border. However, falling was successfully avoided also using the turret-chassis motor (see text for more details).

swarm-bot and movements during the coordination phase. Inertia can cause problems mainly when the direction of motion of the *swarm-bot* is more or less perpendicular to the edge of the hole. In this case, only the *s-bot* at the head of the chain can feel the presence of the hole. If the *swarm-bot* is moving at full speed toward the edge, the heading *s-bot* may not be able to rapidly change direction of motion because of the high inertia of the *swarm-bot*, given also that the other *s-bots* are unaware of the presence of the hole and continue moving at full speed. The avoidance can still be performed if the next *s-bot* feeling the hole is able to change direction of motion. However, in this case it often happens that the *swarm-bot* falls during the coordination phase. Movements and rotations of the *swarm-bot* during the coordination phase are the second main cause of failure of the evolved strategies. In fact, it may happen that while coordinating, the *swarm-bot* reaches an edge and the lack of coherence in the movements may cancel the avoidance effort of those *s-bots* that feel the presence of the hole. This may happen mainly after a successful avoidance, when the *swarm-bot* is close to a corner of the arena.

5.3.1 Robustness Properties

The causes of failure described above occur in many evolved behaviors, also in the best rated behavior produced by the dynamic neural network. This means that in some cases the evolved behavior is not robust enough to cope with all possible hazardous situation.

In order to understand to what extent the evolved behavior are robust with respect to the hole avoidance task, we performed some evaluations of the performance of the best controllers of each replication. To do so, we defined two performance metrics related to the fitness function used in these experiments, but slightly modified in order to test the robustness of the controller. The first is a "survival factor", which corresponds to the fraction of time the *swarm-bot* survives without falling into a hole, and is given by:

$$P_s = \frac{T_s}{T_p}, \tag{5.6}$$

where T_s is the number of cycles the *swarm-bot* survived without falling, and T_p is the total amount of cycles used for this performance evaluation. This metric is clearly related to the fitness component F_s, but it gives more information about the robustness of the behavior with respect to the avoidance of falling.

The second performance metric is the "exploration factor" P_x, related to the fitness component F_x, given by

$$P_x = \frac{z(T_s)}{Z(T_p)}, \tag{5.7}$$

where $z(t)$ and $Z(t)$ have the same meaning as in Equation (5.5), but are computed with a different number of cycles. This metric gives us an idea on how good a controller is in exploring the environment. In particular, it penalizes a situation in which the *swarm-bot* remains trapped in a particular location while trying to avoid to fall. Given that *swarm-bots* that are able to survive longer have also more time to explore the arena, we have combined the previous metrics in a single "exploration per survival cycle" performance metric, given by:

$$P = \frac{P_x}{P_s}, \tag{5.8}$$

which should give a fair comparison of the capability of exploring the arena while not falling into a hole.

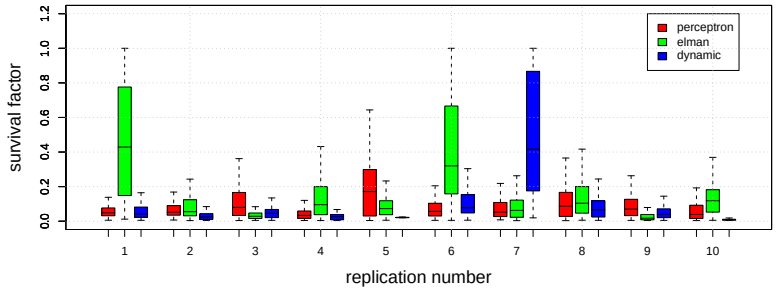
These performance metrics are evaluated for $T_p = 10000$ cycles (1000 seconds), a very long duration with respect to the one used in the fitness

estimation. As a consequence, we have $Z(T_p) = 25$, which corresponds to the exploration of the whole arena. The long evaluation time is intended to let the *swarm-bot* test many different situations while navigating in the arena, situations that could have never appeared before and to which the *swarm-bot* should prove to be robust enough. We repeated the performance measures 100 times for the best individual produced by every replication of the experiment. The results are shown in Figure 5.4.

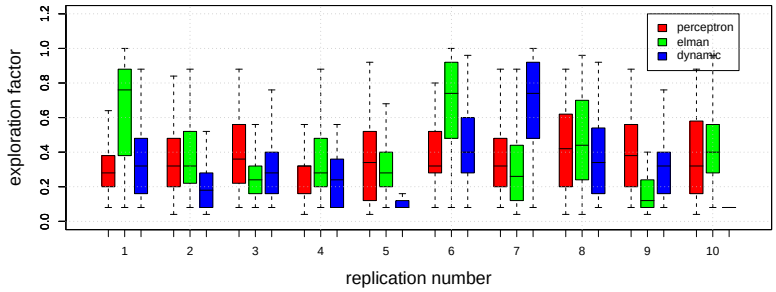
The box-plot shows that most of the evolved controllers have a small survival factor with an average in general lower than 0.2, which corresponds to 2000 cycles spent without falling (see Figure 5.4a). This suggests that there exist situations that evolved controller cannot cope with. However, there are 3 controllers that outperform the others, that is the the Elman networks of the 1^{st} and 6^{th} replications and the dynamic network of the 7^{th} replication. There are no perceptron networks that can be considered robust with respect to surviving. Therefore, we can suppose that Elman and dynamic networks are more robust than a simple perceptron concerning the hole avoidance task, particularly with respect to the capacity to avoid falling into holes.

The exploration factor confirms that the networks best rated by the survival factor were able also to explore efficiently the environment, as it can be seen in Figure 5.4b. However, it is clear that these controllers, being able to survive longer, have also more time to explore the arena. In fact, the discounted exploration metric, plotted in Figure 5.4c, penalizes them with respect to the other controllers. However, the results obtained with the exploration factor are interesting, as they reveal that the controllers that survive longer are also able to coordinately move trough the environment, without being blocked in the attempt to avoid a hole. Their exploration speed is anyway slow with respect to other controllers, as shown in Figure 5.4c.

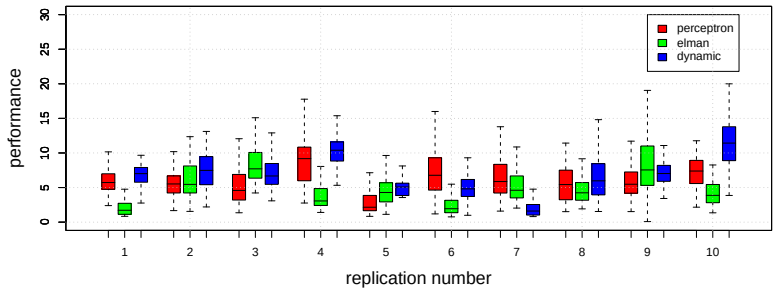
In conclusion, these data reveal that it is possible to evolve neural networks with complex architecture (Elman and dynamic) that show good robustness with respect to the hole avoidance task, while simple perceptrons do not seem to be comparable. However, in order to consistently evolve such robust behaviors, it is necessary to devise a fitness function that explicitly rewards robustness. A similar fitness function must test the controller in as many difficult situations as possible and for long time. The drawback is that such fitness estimation may require an excessive amount of time. There clearly is a trade-off between the desired robustness and the time needed to achieve it.



(a)



(b)



(c)

Figure 5.4: Performance evaluation for the robustness of the evolved behaviors: (a) survival factor (P_s); (b) exploration factor (P_x); (c) combined performance metric (P_x/P_s).

5.4 Generalization Properties

The generalization properties of the coordinated motion task, described in Section 4.4, are a result of the physical connections among *s-bots* and, above all, of the traction sensors we used. Given that the hole avoidance task inherits many features from the coordinated motion task, we expect to observe the same generalization properties. However, we mentioned above that the reaction to the edge detection is influenced by the orientation of the chain and by its inertia. Varying the size and the shape of the *swarm-bot* will worsen this conditions, and may lead to inefficient behaviors.

First, we tested the obstacle avoidance generalization, surrounding the arena used for the hole avoidance task with walls. As shown by Figure 5.5, the *swarm-bot* is able to avoid both holes and obstacles, efficiently exploring the arena. The obstacle avoidance behavior is similar to the one described in Section 4.4, that is, traction sensors work as a distributed bumper for the *swarm-bot*.

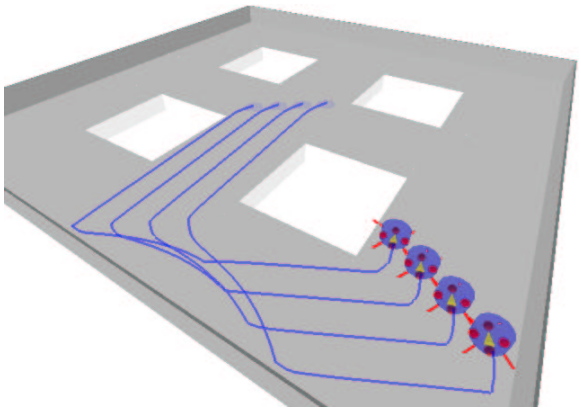


Figure 5.5: Generalization properties: obstacle avoidance. The *swarm-bot* is placed in an arena containing holes and surrounded by walls, and it is able to survive avoiding both holes and obstacles.

The second generalization test was performed using different size and shape for the *swarm-bot*. Figure 5.6 shows the case of a star formation in a squared arena without holes, but with open borders. We do not show a case with internal holes in the arena for presentation purposes: in fact, when working with a high number of *s-bots* and rigid connections, the passages between holes are to narrow to be traversed by the *swarm-bot* without having one or more *s-bots* that sense a hole and trigger a direction change. In these

cases, the trajectories were too confusing to be understood. The case we show in Figure 5.6 is anyway representative of the generalization behavior. The star formation is able to avoid to fall out of the arena, but it can be seen that some of the *s-bots*' trajectories lay outside of the arena. This is due to the higher inertia of the star formation, which can easily push an *s-bot* out, without being able to stop. It is a penalizing factor for the generalized behaviors, as they become less efficient in avoiding holes. However, given the higher number of *s-bots* forming the group, the fall can be avoided when the edge is sensed by other *s-bots* approaching it, so, in general, the performance is satisfying.

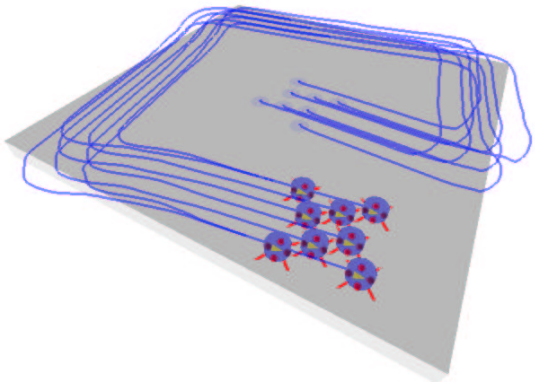
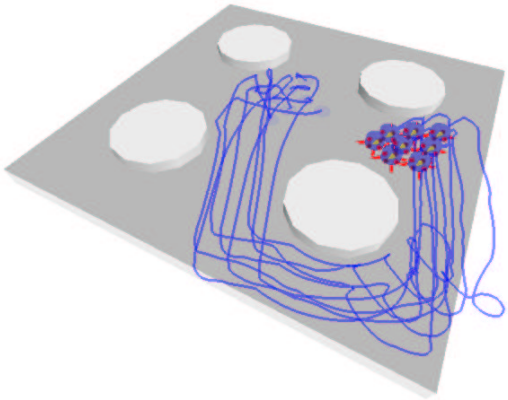


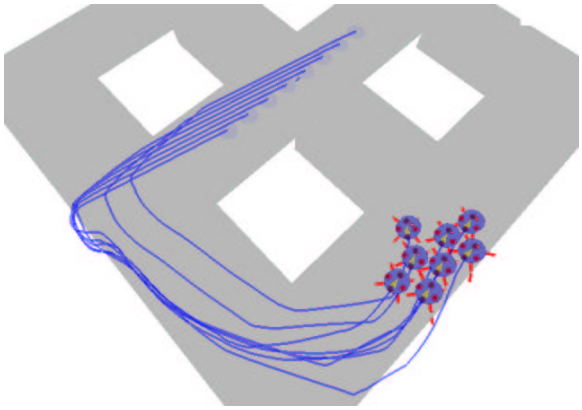
Figure 5.6: Generalization properties: size and shape change. The *swarm-bot* is composed of 8 *s-bots* rigidly connected to form a star formation. In this case, the *swarm-bot* is able to avoid falling, but its high inertia makes the behavior less efficient.

In the last generalization test, we use flexible links in the *swarm-bot* so that it can change shape during the exploration (see Figure 5.7). We performed tests with both a star and a chain formation composed of 8 *s-bots* each. The flexible star formation case is shown in Figure 5.7a, where the *swarm-bot* was placed in a squared arena with four big cylindrical obstacles and no walls on the printer. Figure 5.7a shows that the flexible formation was able to perform coordinated motion, obstacle and hole avoidance, changing shape when it had to go through a narrow passage having an obstacle on the left and the arena border on the right. It can be noticed that the flexible formation adapts more easily adapt to the environment, and in some situations can avoid holes more efficiently than a rigid structure. In fact, the *s-bots* do not completely feel the inertia of the *swarm-bot*, because they can

move deforming the structure and adapting to the edge of the hole. This fact is even more evident in Figure 5.7b, where a chain formation was placed in the original arena. Here, when the chain reached the edge, it completely deformed without having a single *s-bot* being pushed out of the arena.



(a)



(b)

Figure 5.7: Generalization properties: hole and obstacle avoidance with flexible structures. (a) The *swarm-bot*is composed of 8 *s-bots* flexibly connected in a star formation. The arena contains big obstacles, which create some narrow passages with the border of the arena itself. (b) The *swarm-bot* is composed of 8 *s-bots* flexibly connected to form a chain. The arena contains holes, but no obstacles.

Chapter 6

Conclusions

In this work, we have presented a new robotic concept, called a *swarm-bot*, defined as an artifact composed of simpler autonomous robots, called *s-bots*. An *s-bot* has limited acting, sensing and computational capabilities, but can create physical connections with other *s-bots*, thus forming a *swarm-bot* that is able to solve problems an individual cannot cope with. We presented the results obtained in the attempt to control a *swarm-bot*. In particular, we chose to exploit Artificial Evolution for synthesizing the controllers for the *s-bots*, and for obtaining self-organization in the robotic system. The solutions found by evolution are simple, general and in many cases they generalize to different environmental situations. This demonstrates that evolution is able to produce a self-organized system that relies on simple and general rules, a system that is consequently robust to environmental changes and to the number of *s-bots* involved in the experiment.

6.1 Obtained Results

We presented a set of experiments for the evolution of coordinated motion behaviors in a group of simulated *s-bots* that are physically connected to form a *swarm-bot*. We showed that the problem can be solved in a rather simple and effective way by providing the *s-bots* with a traction sensor and by evolving the neural controllers. The evolved strategy exploits the fact that the body of a *swarm-bot* physically integrates the effects of the movements of the single *s-bots*. The traction sensor allows *s-bots* to detect the result of this integration. In this way, the problem of producing coordinated movements can be easily solved. In fact, these sensors allow *s-bots* to have direct access to global information about what the entire group is doing.

In a second set of experiments, we described how coordinated motion can be performed in an environment presenting holes, that have to be avoided in order to not fall into them. Here, the evolved strategies strongly rely on the traction forces produced by those *s-bots* that feel the presence of an hazard. Using the information given by the traction sensors, the whole group can change the direction of motion when heading toward a hole.

We also showed how neural controllers are able to generalize in rather different circumstances, even if they were evolved for a particular case, that is, for the ability to produce coordinated movement and hole avoidance in a *swarm-bot* composed of four *s-bots* forming a linear structure. We have observed that (i) evolved controllers produce coordinated movements in *swarm-bots* with varying size, topology, and type of links, and (ii) they display obstacle avoidance when placed in an environment with obstacles. These results suggest that this strategy might constitute a basic functionality that, complemented with appropriate additional functions, might allow *swarm-bots* to display a large number of interesting behaviors.

The traction sensor was found to be a very powerful mean of achieving coordination in the *swarm-bot*. In fact, it allows to exploit the complex dynamics arising from the interaction among *s-bots* and between *s-bots* and environment. It provides robustness and adaptivity features with respect to environmental or structural changes of the *swarm-bot*. It is very versatile, as it also functions as a distributed bumper for the *swarm-bot* used for obstacle avoidance. Besides, traction forces are used as a sort of communication of the presence of an hazard. This communication among *s-bots* is neither direct nor explicit, but can be considered as an implicit stigmergic communication, as it takes place through the environment, that is, through the bodies and the physical connections between *s-bots*.

Concerning the hole avoidance task, we performed three sets of experiments evolving different neural network architectures: simple perceptrons and two recurrent architectures (Elman and dynamic neural networks). We were able to obtain satisfying controllers using all the three types of neural networks. However, a robustness analysis showed that not all the behaviors were equally efficient. We found that, among the evolved strategies, only some recurrent networks resulted in robust solutions, while the behaviors produced by simple perceptrons never displayed a similar performance.

6.2 Future Work

In this work, we tried to evolve neural networks with complex architectures in order to understand if internal dynamics of the network could provide adaptive advantage in the hole avoidance task. We showed that some of the evolved networks are more robust than simple perceptrons, but the collected data cannot state that they are consistently better. The understanding of the dynamics of the evolved networks can give useful insights on the mechanisms that produce the observed behaviors. Thus, we plan to analyze in detail the evolved neural networks, performing neuro-ethological analysis (lesion and correlation studies). Such analysis can help us understanding the functionality of the sub-components of the neural network, in order to state if the better performance displayed was effectively a result of the higher complexity of the neural network. If this is the case, we will perform new experiments aiming at consistently evolving those features that are relevant for a robust and efficient hole avoidance behavior.

The hole avoidance task represents the first step toward the solution of more difficult problems. We plan to continue studying problems that belong to the "navigation on rough terrain" family, like passing over a trough or coping with rough terrain. Finally, we will face the challenge given by functional self-assembling for all-terrain navigation, that is, we will study the problem of forming or disbanding *swarm-bots* given the environmental conditions, in order to maximize the efficiency in the navigation task. This problem also requires the formation of suitable shapes, which must be adapted to the environment the *swarm-bot* is coping with. For example, if the *swarm-bot* has to pass over a trough that is as large as a single *s-bot*, then the optimal shape is a circle with a radius bigger than the diameter of an *s-bot* because in this way the trough can be traversed no matter which is the orientation of the *swarm-bot*. This is a very particular case, but in general the optimal shape depends on the number of *s-bots* available and on the environmental conditions. Thus, the shape formation must be an emergent result of the interaction among *s-bots* and between *s-bots* and environment. Therefore, *s-bots* should be able to self-organize and self-assemble in order to build the most suitable *swarm-bot*.

Bibliography

[1] W. Agassounon, A. Martinoli, and R.M. Goodman. A scalable, distributed algorithm for allocating workers in embedded systems. In *Proc. of the IEEE Conf. on System, Man and Cybernetics SMC-01*, pages 3367–3373. IEEE Press, 2001.

[2] C. Anderson, G. Theraulaz, and J.-L. Deneubourg. Self-assemblage in insects societies. *Insectes Sociaux*, 49:99–110, 2002.

[3] W.R. Ashby. *Design for a Brain*. Wiley & Sons, New York, 1952.

[4] T. Balch and R.C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):27–52, 1994.

[5] G. Baldassarre, S. Nolfi, and D. Parisi. Evolving Mobile Robots Able to Display Collective Behaviours. In C.K. Hemelrijk and E. Bonabeau, editors, *Proceedings of the International Workshop on Self-organisation and Evolution of Social Behaviour*, pages 11–22, Monte Verità, Ascona, Switzerland, September 8-13, 2002.

[6] G. Baldassarre, S. Nolfi, and D. Parisi. Evolution of collective behavior in a team of physically linked robots. In *Proceedings of EvoROB2003*, Essex, UK, April 2003. To appear.

[7] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, NY, 1999.

[8] A. Bonarini and V. Trianni. Learning fuzzy classifier systems for multi-agent coordination. *Information Science*, 136:215–239, 2001.

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

[10] S. Camazine and J. Sneyd. A model of collective nectar source selection by honey bees: self-organization through simple individual rules. *Journal of Theoretical Biology*, 149:547–571, 1991.

[11] S. Camazine, J. Sneyd, M.J. Jenkins, and J.D. Murray. A mathematical model of self-organized pattern formation the combs of honeybees colonies. *Journal of Theoretical Biology*, 1:295–311, 1990.

[12] A. Castano, W. Shen, and P. Will. CONRO: Towards Deployable Robots with Inter-Robot Metamorphic Capabilities. *Autonomous Robots*, 8:309–324, 2000.

[13] G.S. Chirikjian. Kinematics of a Metamorphic Robotic System. In E. Straub and R. Spencer Sipple, editors, *Proceedings of the International Conference on Robotics and Automation. Volume 1*, pages 449–455. IEEE Computer Society Press, 1994.

[14] J.-L. Deneubourg, S. Aron, S. Goss, and J.M. Pasteels. The self-organizing exploratory patterns of the argentine ant. *Journal of Insect Behaviour*, 3:159–168, 1990.

[15] J.-L. Deneubourg and S. Goss. Collective patterns and decision making. *Ethology Ecology Evolution*, pages 295–311, 1989.

[16] J.-L. Deneubourg, J. C. Gregoire, and E. Le Fort. Kinetics of the larval gregarious behaviour in the bark beetle *Dendroctonus micans*. *Journal of Insect Behavior*, 3:169–182, 1990.

[17] C. Detrain. Field study on foraging by the polymorphic ant species pheidole pallidula. *Insectes Sociaux*, 37(4):315–332, 1990.

[18] J.L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[19] T.D. Fitzgerald. *The tent caterpillars*. Cornell University Press, Ithaca, New York, 1995.

[20] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, 3(26):396–407, 1996.

[21] B.P. Gerkey and M.J. Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[22] D. Goldberg and M. Matarić. Design and evaluation of robust behavior-based controllers. In T. Balch and L.E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*. A K Peters, 2002.

[23] P.P. Grassé. La reconstruction du nid et les coordinations inter-individuelle chez *Bellicositermes natalensis* et *Cubitermes*. La théorie de la stigmergie: Essai d'interpretation des termites constructeurs. *Insectes Sociaux*, 6:41–83, 1959.

[24] I. Harvey, P. Husbands, and D. Cliff. Seeing the light: Artificial evolution, real vision. In D. Cliff, P. Husbands, J.-A. Meyer, and S.W. Wilson, editors, *From Animals to Animats 3: Proc. of the Third Int. Conf. on Simulation of Adaptive Behavior*, pages 392–401. The MIT Press, 1994.

[25] F. Heylighen. The Science of Self-Organization and Adaptivity. In *The Encyclopedia of Life Support Systems (EOLSS)*, Knowledge Management, Organizational Intelligence and Learning, and Complexity. Developed under the Auspices of the UNESCO, Eolss Publishers, Oxford, UK, 2003.

[26] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[27] A. Kamimura, S. Murata, E. Yoshida, H. Kurokawa, K. Tomita, and S. Kokaji. Self-Reconfigurable Modular Robot - Experiments on Reconfiguration and Locomotion. In *Proc. of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS2001*, pages 606–612, Piscataway, NJ, October 29 - November 3 2001. IEEE. Conference, Maui, Hawaii, USA.

[28] A. Lioni, C. Sauwens, G. Theraulaz, and J.-L. Deneubourg. Chain formation in *Œcophilla longinoda*. *Journal of Insect Behaviour*, 15:679–696, 2001.

[29] M.J. Matarìc. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16:321–331, 1995.

[30] M.J. Matarić. Using communication to reduce locality in distributed multiagent learning. *Journal of Experimental and Theoretical Artificial Intelligence,*, 10(3):357–369, 1998.

[31] C. Melhuish. Exploiting domain physics: Using stigmergy to control cluster building with real robots. In Dario Floreano, Jean-Daniel Nicoud, and Francesco Mondada, editors, *Proceedings of the 5th European Conference on Advances in Artificial Life (ECAL-99)*, volume 1674 of *LNAI*, pages 585–595. Springer, 1999.

[32] F. Mondada, E. Franzi, and P. Ienne. Mobile Robot Miniaturisation: A Tool for Investigation in Control Algorithms. In *Proceedings of the Third International Symposium on Experimental Robotics*, pages 501–513, Kyoto, 1993.

[33] F. Mondada, G. C. Pettinaro, I. Kwee, A. Guignard, L. Gambardella, D. Floreano, S. Nolfi, J.-L. Deneubourg, and M. Dorigo. SWARM-BOT: A swarm of autonomous mobile robots with self-assembling capabilities. In C.K. Hemelrijk and E. Bonabeau, editors, *Proceedings of the International Workshop on Self-organisation and Evolution of Social Behaviour*, pages 307–312, Monte Verità, Ascona, Switzerland, September 8-13, 2002.

[34] S. Murata, H. Kurokawa, and S. Kokaji. Self-assembling machine. In E. Straub and R. Spencer Sipple, editors, *Proceedings of the International Conference on Robotics and Automation. Volume 1*, pages 441–448. IEEE Computer Society Press, 1994.

[35] G. Nicolis and I. Prigogine. *Self-Organization in Nonequilibrium Systems.* John Wiley & Sons, New York, 1977.

[36] S. Nolfi. Evolving non-trivial behavior on autonomous robots: Adaptation is more powerful than decomposition and integration. In T.Gomi, editor, *Evolutionary Robotics*, pages 21–48. AAI Books, Ontario (Canada), 1997.

[37] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines.* MIT Press/Bradford Books, Cambridge, MA, USA, 2000.

[38] E. Parker. ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation. *IEEE Transactions on Robotics and Automation*, 14:220–240, 1998.

[39] L.E. Parker, G Bekey, and J Barhen, editors. *Distributed Autonomous Robotic Systems 4.* Springer, Tokyo, Japan, 2000.

[40] G.C. Pettinaro, I.W. Kwee, L.M. Gambardella, F. Mondada, D. Floreano, S. Nolfi, J.-L. Deneubourg, and M. Dorigo. Swarm robotics: A different approach to service robotics. In *Proceedings of the 33rd International Symposium on Robotics*, Stockholm, Sweden, October 7-11 2002. International Federation of Robotics.

[41] M. Quinn. Evolving communication without dedicated communication channels. In J. Kelemen and P. Sosik, editors, *Advances in Artificial Life: Sixth European Conference on Artificial Life (ECAL 2001)*, pages 357–366, Berlin, 2001. Springer-Verlag.

[42] M. Quinn, L. Smith, G. Mayley, and P. Husband. Evolving teamwork and role allocation with real robots. In R.K. Standish, M.A. Bedau, and H.A. Abbass, editors, *Proceedings of the 8th International Conference on Artificial Life*, pages 302–311. MIT Press, 2002.

[43] F. Saffre, R. Furey, B. Kraft, and J.L. Deneubourg. Collective decision-making in social spyders: Dragline-mediated amplification process acts as a recruiting mechanism. *Journal of Theoretical Biology*, 198:507–517, 1999.

[44] E. Şahin, T.H. Labella, V. Trianni, J.-L. Deneubourg, P. Rasse, D. Floreano, L.M. Gambardella, F. Mondada, S. Nolfi, and M. Dorigo. SWARM-BOTS: Pattern formation in a swarm of self-assembling mobile robots. In A. El Kamel, K. Mellouli, and P. Borne, editors, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Hammamet, Tunisia, October 6-9, 2002. Piscataway, NJ: IEEE Press.

[45] T.D. Seeley. *The Wisdom of Hive*, pages 277–290. Harvard University Press, Cambridge, 1995.

[46] T.D. Seeley, S. Camazine, and J. Sneyd. Collective decision-making in honey bees: how colonies choose among nectar source. *Behavioral Ecology and Sociobiology*, 28:277–290, 1991.

[47] W.-M. Shen, Y. Lu, and P. Will. Hormone-Based Control for Self-Reconfigurable Robots. In C. Sierra, M. Gini, and J.S. Rosenschein, editors, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 1–8. ACM Press, 2000.

[48] H. Stone. Mars Pathfinder Microrover - A Small, Low-Cost, Low-Power Spacecraft. In *Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics*, University of Wisconsin, Madison, WI, USA, August 1-2 1996.

[49] K. Støy, W.-M. Shen, and P. Will. Global locomotion from local interaction in self-reconfigurable robots. In W.M. Shen, C. Torras, and H. Yuasa, editors, *Proceedings of the 7th international conference on intelligent autonomous systems (IAS-7)*, pages 309–316. IOS Press, 2002.

[50] V. Trianni, R. Gross, T.H. Labella, E. Şahin, P. Rasse, J.-L. Deneubourg, and M. Dorigo. Evolving Aggregation Behaviors in a Swarm of Robots. Technical Report TR/IRIDIA/2003-07, IRIDIA - Université Libre de Bruxelles, Belgium, February 2003.

[51] H. von Foerster. On Self-Organizing Systems and Their Environments. In M.C. Yovits and S. Cameron, editors, *Self-Organizing Systems*, pages 31–50. Pergamon Press, London, 1960.

[52] C.R. Ward, F. Gobet, and G. Kendall. Evolving collective behavior in an artificial ecology. *Artificial Life*, 7(2):191–209, 2001.

[53] E.O. Wilson. The Relation Between Caste Ratios and Division of Labour in the Ant Genus Pheidole (Hymenoptera: Formicidae). *Behav. Ecol. Sociobiol.*, 16:89–98, 1984.

[54] M. Yim, D.G. Duff, and K.D. Roufas. PolyBot: a Modular Recon gurable Robot. In *Proceedings of the 2000 IEEE/RAS International Conference on Robotics and Automation*, volume 1, pages 514–520. IEEE. Conference, San Francisco, 2000.