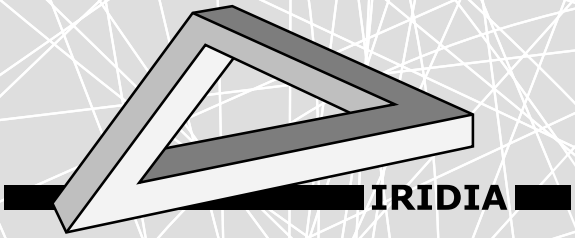

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Swarm Intelligence techniques for task allocation and sub-task merging in multi-agent systems

Christos Ampatzis

Technical Report No.

TR/IRIDIA/19

03/09/2004

DEA Thesis

Swarm Intelligence techniques for task allocation and sub-task merging in multi-agent systems

by

Christos Ampatzis

Université Libre de Bruxelles
Faculté des Sciences Appliquées
IRIDIA
CP 194/6, Avenue Franklin Roosevelt 50, 1050 Brussels, Belgium
campatzi@ulb.ac.be

Supervised by

Marco Dorigo, Ph.D.

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

August, 2004

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

DIPLOME D'ETUDES APPROFONDIES (DEA)

Abstract

In this work, we address the problem of synthesizing non-reactive controllers for a swarm robotic system, called *swarm-bot*, using Artificial Evolution. In particular, we evolve simple dynamical neural networks, in order to achieve autonomous decision-making agents that are able to integrate over time their perceptual experience. These agents cooperate in carrying out certain tasks by communicating their experience to the rest of the group. We show the applicability of our desision-making mechanisms in the realisation of a complex scenario.

Acknowledgments

First of all I would like to thank my supervisor Marco Dorigo for giving me the chance to work on such a challenging project and for his useful guidance and advice. I would also like to thank especially Vito, but also Ciro, Rodi, Halva and all people at IRIDIA for providing everyday advice and help, and for being patient with my initial ignorance. Thanks also go to Joke for her support, tolerance, patience and love. I would also like to express my admiration for Isaac Asimov, who in his book of 1950 "I Robot" (in the story "Runaround") describes a scene where a robot acts VERY similar to my simulated one. Thanks to Dimitrios Evangelinos for letting me know. Last but definitely not least, I would like to thank Elio Tuci, for an endless list of reasons: for giving me the chance to work on this very interesting subject, for helping me during the writing of this document and for priceless advice during my experimental work. Without Elio I would not have been able to reach this point.

Contents

1 Introduction **1**
- 1.1 Background . . . 3
  - 1.1.1 Collective Robotics . . . 3
  - 1.1.2 Metamorphic Robotics . . . 4
  - 1.1.3 Evolutionary Robotics . . . 5
- 1.2 Swarm Robotics and The Swarm-Bots Project . . . 7
  - 1.2.1 Swarm Robotics . . . 7
  - 1.2.2 The SWARM-BOTS project . . . 8
- 1.3 Report Layout . . . 11

2 Evolving Time-Dependent Structures **12**
- 2.1 Literature Review . . . 12
- 2.2 Integration Over Time . . . 14
  - 2.2.1 CTRNNs . . . 17
- 2.3 The Tuci *et al.* Experiment : "Evolving the "feeling" of time through sensory-motor coordination: a robot based model" . 18
  - 2.3.1 The simulation . . . 23
  - 2.3.2 The controller . . . 23
  - 2.3.3 The evolutionary algorithm . . . 24
  - 2.3.4 The experiment - The evaluation function . . . 24
  - 2.3.5 Results . . . 26
  - 2.3.6 Discussion and Conclusions . . . 28

3 Replicating Tuci *et al.* in SWARMBOTS3D **31**
- 3.1 Methodological Issues . . . 31
- 3.2 The SWARMBOTS3D Simulator . . . 32
  - 3.2.1 Sensor, Actuator and Network Configuration . . . 34
  - 3.2.2 The Simulation and the new evaluation function . . . 36
- 3.3 The Replication with SWARMBOTS3D . . . 37

3.4 The Minimal Simulation Approach 38
3.5 Results 40
3.5.1 The Replication 40
3.5.2 Analysis of the evolved behavioral strategies 41
3.5.3 Post Evaluation in the Minimal Simulator Environment 43
3.5.4 Robustness of the evolved solutions 45
3.5.5 Post Evaluation in SWARMBOTS3D 47

4 Evolving communicating agents 52
4.1 The Task 53
4.2 The Simulation 54
4.3 Results 57
4.3.1 Post-Evaluation in the Minimal Simulator 57
4.3.2 Post-Evaluation in SWARMBOTS3D 60

5 Conclusions 63
5.1 Results 63
5.2 Future Work 64

List of Figures

1.1 Graphical visualization of an *s-bot*. 8
1.2 Graphical visualization of possible scenarios involving a *swarm-bot*. 9
1.3 Picture of the scenario 10

2.1 The task 20
2.2 A picture of a Khepera robot on the left. Plan of the robot on the right, showing sensors and motors. The robot is equipped with two ambient light sensors (L_1 and L_2) and a floor sensor indicated by the black square F. The left and right motor (M_1 and M_2) are controlled by a dynamic neural network (NN). A simple sound signalling system, controlled by an output of the network, is referred to as S. 22

3.1 The simulated *s-bot* model 33
3.2 The simulated *s-bot* models 34
3.3 The sound emitting system of the *s-bot* 36
3.4 The Fitness during the evolution. The top thin line corresponds to the fitness of the best individual, while the dotted line refers to the average fitness of the population. 38
3.5 Average fitness during the evolution. All plots are the average over the 20 replications of the experiment. The top thin line corresponds to the average fitness of the best individual, while the dotted line below refers to the average fitness of the population. 41

3.6 Behavioral analysis. The sensor activity and the corresponding motor output are plotted for 700 simulation cycles. L1 and L2 refer to the light sensors, while F refers to the floor sensor. M1 and M2 correspond to the motors of the two wheels, and S refers to the sound signalling. When S is bigger than 0.5, the robot emits a signal. 43

3.7 Robustness analysis for run no.20. The offset Δ is plotted for varying light-band distance. The box-plot shows 100 evaluations per box. Boxes represent the inter-quartile range of the data, while the horizontal bars inside the boxes mark the median values. The whiskers extends to the most extreme data points within 1.5 of the inter-quartile range from the box. The empty circles mark the outliers. 48

4.1 The Fitness during the evolution. The top thin line corresponds to the fitness of the best individual, while the dotted line refers to the average fitness of the population. 57

List of Tables

3.1 Post-evaluation. Performance of the ten best evolved controllers. The percentage of success (*Succ.* %) and the percentage of errors (*E1*, and *E2* in *Env.A*, and *E3*, and *E4* in *Env.B*,) over 100 trials are shown for both *Env.A* and *Env.B*. Additionally, the average offset Δ and its standard deviation (degrees) are shown for the environment type *Env.B*. 46

3.2 Robustness analysis. Performance of the twenty best evolved controllers. The average offset of 100 evolutionary runs is given for the mentioned distances between the circular band and the light. 47

3.3 Post-evaluation in SWARMBOTS3D. The percentage of success (*Succ.* %) and the percentage of errors (*E1*, and *E2* in *Env.A*, and *E3*, and *E4* in *Env.B*,) over 100 trials are shown for both *Env.A* and *Env.B*. Additionally, the average offset Δ and its standard deviation (degrees) are shown for the environment type *Env.B*. These values are displayed for various timesteps and initial robot orientations. 51

4.1 Post-evaluation in the Minimal Simulation environment. Performance of the ten best evolved controllers in *Env.A*. The percentage of success (*Succ.* %) and the percentage of errors *E1* and *E2* over 100 trials are shown for both robots. Robot 1 is initialised closer to the light source. 60

4.2 Post-evaluation in the Minimal Simulation environment. Performance of the ten best evolved controllers in *Env.B*. The percentage of success (*Succ.* %), the reaction time, the percentage of errors *E3*, *E4* and *E5* over 100 trials are shown for both robots. Additionally, we show the average offset Δ and its standard deviation (degrees) for the first robot that completes the loop. Robot 1 is initialised closer to the light source. 61
4.3 Post-evaluation in SWARMBOTS3D. Performance of one evolved controller in *Env.A*. The percentage of success (*Succ.* %) and the percentage of errors *E1* and *E2* over 100 trials are shown for both robots. Robot 1 is initialised closer to the light source. 62
4.4 Post-evaluation in SWARMBOTS3D. Performance of one evolved controller in *Env.B*. The percentage of success (*Succ.* %), the reaction time, the percentage of errors *E3*, *E4* and *E5* over 100 trials are shown for both robots. Additionally, we show the average offset Δ and its standard deviation (degrees) for the first robot that completes the loop. Robot 1 is initialised closer to the light source. 62

Chapter 1

Introduction

This work addresses the problem of defining the control system for a group of autonomous robots that have to deal with a non-reactive task. We aim to design neural controllers for a group of autonomous robots equipped with simple sensors. The robots integrate over time their perceptual experiences in order to initiate alternative actions. In other words, the behavior of the agents should change as a consequence of their repeated interaction with particular environmental circumstances. We are interested in exploiting a biologically-inspired evolutionary approach, based on the use of dynamical neural networks and genetic algorithms [5]. In general, 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 and is in fact limited by the designer's a priori intuitions. The designer must discover the rules that must be encoded into the controller in order to achieve a certain goal, and decompose the task into several subtasks. 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 entity that can take its own decisions depending on the current sensory input information, but also on its internal state. Furthermore, robots interact with each other, making the system

much more dynamic and complex. The designer must be capable of predicting the outcome of such interactions, which could be extremely difficult, even impossible. On the contrary, Artificial Evolution does not suffer from this problem since it is an automatic process that directly tests the behavior displayed by the robots embedded in their environment and selects out the bad-performing individuals. 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, being relatively unbiased. 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* [55, 40]. The *swarm-bot* is defined as an artifact composed of simple autonomous robots, called *s-bots*. An *s-bot* has limited acting, sensing and computational capabilities, but can create physical connections with other *s-bots*, therefore forming a *swarm-bot* that is able to solve problems the single individual cannot cope with. Up to now, in the project have been studied only reactive behaviors. We chose to study integration over time, a non-reactive task, that is a task that in order to be carried out by the robot, needs "memory". The robot's behavior will not only be affected by its current sensory status, but also by its internal dynamics. At this specific moment in the project, the study of efficient decision-making mechanisms is necessary in order to succeed in integrating different behaviors exhibited by the *swarm-bot*, for which efficient controllers have already been successfully evolved. The work described in this paper will tackle the problem of designing a controller which is able to integrate sensorial information over time and adjust its subsequent behavior accordingly. With the use of communication, we will expand its functionality for a group of robots. For more details on the significance of this work for the project, the reader is suggested to see Section 1.2.2. In the rest of this chapter, we first present the state-of-the-art, describing the research fields that constitute the starting point of this work (see Section 1.1). In Section 1.2, after some general information and state-of-the-art, we present in detail the SWARM-BOTS project and our contribution to it. Finally, Section 1.3 briefly summarizes the contents of this thesis.

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

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 tasks 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, multipurpose robot is a complex task, that can also prove to be very expensive. Another problem that might be experienced with the single-robot approach is that even small failures may prevent the accomplishment of the whole task. A group of simple and cheap robots may be able to efficiently accomplish many tasks that go beyond the capabilities of the individual robot. This idea is the cornerstone 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 development of controllers for autonomous robots (for a review see [45]). 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 act together in order to reach a common goal (for an overview of the field, see [49]). The main motivation behind the study of collective robotic systems lays in the possibility to decompose the solution of a complex problem into subproblems 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 [23] propose a dynamic task allocation method based on auction exchange in order to achieve cooperation in a group of robots. Agassounon *et al.* [2] use a scalable algorithm based on a threshold model for the allocation of robots in a puck collecting and clustering task. Melhuish [38] describes a clustering task collectively performed by a group of cooperating robots. Schenker *et al.* [56] summarize the robotics work being carried out at NASA Jet Propul-

sion Laboratory. They report on the development of RWC (a multi-Robot Work Crew), which consists of cooperating rovers controlled a decentralised behavior-based control architecture. It is an interesting approach since it is designed not only for cooperative group behaviors but also for tightly coordinated tasks such as the transporting of large payloads [51].

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 [48] 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 [24] 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 modelling the internal state of other agents, or for communicating the execution of a particular action to a teammate, as will also be the case in our work. Bonarini and Trianni [9] have shown that the communication of "cooperation proposals" can help learning cooperative behaviors. Matarić [37] 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 [3] 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.* [14] 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.* [69] 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.* [41] 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.* [34] have developed MTRAN, which got 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 and can self-reconfigure. Shen *et al.* [57, 13] with CONRO proposed another work that follows the above-mentioned directions. Robot morphology is ensured by modular identical 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.* [58]. 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 systems are only some examples of the possible ways of controlling a robot. Among these, Evolutionary Robotics is a very promising technique for the synthesis of robot controllers [45]. It is inspired by the Darwinian principle of selective reproduction of the fittest individual in a population. The process of searching the design space by mimicking natural evolution is generally referred to as Evolutionary Algorithms. In this thesis we will employ a particular type of Evolutionary Algorithms called genetic algorithms [31]. A genetic algorithm works as follows: 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*).

Evolutionary Robotics provides us with a unique opportunity to couple an agent's dynamical system with the environment's dynamical system, through sensory-motor interactions. By exhibiting both situatedness and embodiment, it evaluates a solution based on the agent's interaction with its enviroment [30].

Many difficult control problems have been easily solved relying on the evolutionary approach. For example, Nolfi [42] successfully evolved a controller for the Khepera robot [39] 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 did very well, while this task is very difficult to be solved by hand design—with a behavior-based controller. 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.* [29] 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 at the same time avoiding a rectangular one, guided by the vision system. Floreano and Mondada [19] 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.* [4] evolved group behaviors for simulated Khepera robots, which had to aggregate and navigate toward a light target. Quinn [52] evolved coordinated motion behaviors with two Khepera. On the same track, Quinn *et al.* [53] studied coordinated motion with three wheelchair robots.

1.2 Swarm Robotics and The Swarm-Bots Project

1.2.1 Swarm Robotics

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.

Sugawara *et al.* have studied different aspects of swarm robotic systems. In [59], they study the task of gathering pucks to a fixed point under different distributions of pucks in the environment. When a robot found a puck, it stopped and emitted light for a certain time duration, to broadcast its position. The emitted light served as an attraction field to other unladen robots. The performance of the swarm was measured through the percentage of collected pucks with respect to time. Among other things, the authors have also presented results for the aggregation of the robots, resembling that of amoebae. The authors also proposed an analytical model of the swarm robotic system, to explain some of the dynamics of the system.

Payton *et al.* [50] work on the Pheromone Robotics project and have built a swarm robotic system in order to study the coordination of robots for tasks such as surveillance, reconnaissance, hazard detection and path finding. The system consisted of a group of mobile robots, called *pherobots*, that can locally communicate with each other using infrared-based transceivers mounted on them.

DARPA (Defense Advanced Research Projects Agency) awarded a grant to Icosystem Corporation (http://www.icosystem.com) to apply swarm intelligence methods to the control of robotic swarms. The project is titled "Design of Control Strategies for Swarms of Unmanned Ground Vehicles" and it proposes to develop strategies to control swarms of robots carrying out indoor navigation and reconnaissance tasks. The underlying research goal of this project is to address a number of fundamental questions about swarm control [66, 22].

Bruemmer *et al.* [12] report on the use of social potential attractive and repulsive fields emitted by each robot, as a means to coordinate group

behavior and promote the emergence of swarm intelligence. They tackle the problem of spill finding and perimeter detection by a swarm of robots.

Gaudiano *et al.* [21] studied the control of a swarm of UAVs (Unmanned Air Vehicles). The work is done in simulation for the problems of search over a region (which can also be seen as an area coverage task). They simulated different strategies and analyzed their efficiencies.

1.2.2 The SWARM-BOTS project

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. *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).

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

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

In the *swarm-bot* formation, 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.

(a) (b)

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

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 definitely of utmost 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. 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 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*.

Up to now, the project's empirical work has focused on the study of Coordinated Motion (see [16]), Cooperative Transport (see [26, 27]), Chain Formation and Task Allocation(see [36, 35]). The final goal of the project is the successful realization of a scenario described in detail in [15]. Figure 1.3 gives an approximate idea about the settings of this scenario.

A swarm of up to 35 s-bots must transport a heavy object from its initial

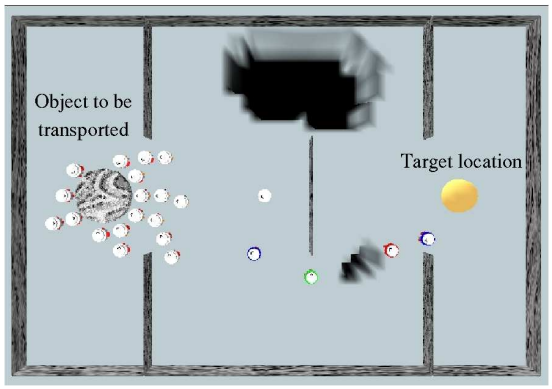

Figure 1.3: Picture of the scenario

location to a goal location. There are several possible paths from the initial to the goal location and these paths may have different lengths and may require avoiding obstacles and holes. The weight of the object is such that its transportation requires the coordinated work of at least n s-bots, where n is a parameter . Since the "building blocks" of the overall behavior exist, we need a strategy that will provide us with an effective decision making mechanism, for swapping between strategies. We also need a strategy that will allow each *s-bot* to realise its current status in the work it is carrying out. For example, robots that explore the enviroment in order to find the goal location, might encounter holes in it, as shown in the above picture. They need to be able to make a decision as to if the hole can be traversed by a single robot. If this is not the case, to call for help, resulting in a *swarm-bot* formation which might be more effective in passing over this gap. Definitely, it is a "cheaper" solution if one robot can solve the task alone, but we are interested in cases where this is not possible. Thus, the work demonstrated in this thesis addresses the problem of evolving neural network controllers which will be able to produce this decision-making mechanism, which in turn will contribute to the realization of the scenario. Our goal is to solve a simplified part of it, which will be the first step in the realization of the complete complex scenario, capturing the elements of a decision-making mechanism. More in detail, our work is focused on the design of controllers evolved to tackle non-reactive problems, problems where a simple reactive behavior is not enough to solve the task, but a kind of "memory" and internal dynamics are necessary. Consequently, we will result in controllers consisting in a very different structure compared to what has been used up to now in the project.

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 integration over time as a decision making mechanism for the *swarm-bot*. We present the work of Tuci *et al.* [65] which is the starting point of this research and discuss its limitations, discussing possible extentions and adaptations in order for this idea to fit into the SWARM-BOTS project context.

In Chapter 3, we present the setup used for a first set of experiments performed, the replication of the work of Tuci *et al.* in a physics-based 3D enviroment. We provide the motivation for the experiments performed and for the simulation model used, by introducing the notion of Minimal Simulation, as defined by Jakobi in [32]. We will show that since our experiments don't require dynamics and physics, they can be conducted within a Minimal Simulation environment. The latter is much faster than a 3D physics-based simulator. We also describe the simulation, controller and evolutionary algorithm we employed in all the performed experiments. Finally, we provide the results obtained and an analysis performed.

In Chapter 4, we present a second set of experiments performed, extending the work presented in 3, the results obtained and their statistical analysis. This time, the task is more oriented towards a Collective Robotics scenario, since it requires communication between two robots.

In Chapter 5, we draw the conclusions of this work, highlighting the important aspects of this research. Finally, we indicate the possible future research directions to be followed.

Chapter 2

Evolving Time-Dependent Structures

Several studies have described evolutionary simulation models in which time-dependent structures are evolved to control the behavior of agents required to make decisions based on their experiences. The aim of Section 2.1 is to present the related work in literature, introducing the distinction between ecological and non-ecological models. Section 2.2 gives an overview of the works introducing integration over time while in Section 2.3 we present in detail the Tuci *et al.* experiments, which are the starting point and inspiration of the work presented in this thesis.

2.1 Literature Review

It is useful to draw a line between two lines of research present in the literature, in a very simple way. These are non-ecological and ecological models [60, 61, 68, 63]. In ecological models, like the one by Tuci *et al.* [65], described in detail in the following section, and our experimental work, the agent's perception is brought forth by the agent itself through its actions. Contrary to that, in the non-ecological models the perceptual experience of the agents is determined by the experimenter. Obviously, the flow of perception provides the agents the cues to make the discrimination (for more on this issue see also [47]). The input to the network is not determined by the network's output at previous timestep. That is, the network does not bring forth the world which it experiences through its sensors. Moreover, some of the non-ecological models (see [68]) are further simplified by the presence of an explicit reinforcement signal—i.e., an input signal explicitly dedicated

to inform the agent's controller on the characteristics of the "environmental circumstances" in which it is currently situated by making available to the system any possible mismatch between the current agent's action and the correct response.

For example, in [63], populations of CTRNNs (Continuous-Time Recurrent Neural Networks) are evolved to solve the "Dowry Problem"—a sequential decision problem in which an agent has to maximise the expected payoff given by choosing a single item from among a population of sequentially encountered items. The latter appears to the agent in random order, and they are drawn from a population with parameters that are completely unknown ahead of time. The sequence of items presented to the network is not in any case affected by the response of the agent at previous time. The results of the simulations show that evolved CTRNNs are capable of sampling a certain proportion of the population of items that is currently experiencing to get an estimation of the distribution of values, and subsequently to exploit this information and make a choice. In [68], an "abstract" agent—i.e., a disembodied dynamic neural network—was responsible of solving the integration of reactive and non-reactive behaviors consisting mainly in generating the appropriate n-bit sequence chosen from either two, three, or four possible different sequences.

Other studies on the evolution of time-dependent structures for discrimination tasks share with ours and the Tuci *et al.* experiment a more ecological perspective, in which the nature of the agent's perception is determined by its own actions, and the reinforcement signals are part of the evolved structures (see [70, 64, 44, 8]). The evolution of time-dependent structures and decision-making mechanisms has been extensively studied on the T-maze problem (see [70, 8]). The robot is required to find its way to a goal location, placed at the bottom of any of the two arms of the maze. When at the T junction, the robot must decide whether to turn left or right. The correct decision can be made if the agent is capable of exploiting perceptual cues which were available to it while it was navigating down the first corridor, or by "remembering" something about previous trials in a similar T-maze. In [70], weight change mechanisms provide the agents the required plasticity to exploit the relationship between the location of light signals placed roughly at the middle of the first corridor, and the turn to make at the junction. Blynel *et al.* in [8] allow the agent to experience the environment in a first trial, in which the success or failure play the role of a reinforcement signal, in order to associate the position of the goal with respect to the T-junction.

In [64], evolved CTRNNs provide the agents the required plasticity to

discover the spatial relationship between the position of a landmark and the position of a goal. In this study, the spatial relationship between the goal and the landmark can be learnt by "remembering" from previous trials the relative position of the landmark with respect to the goal. The work illustrated by Nolfi in [44] investigates a discrimination task in which a robot, while navigating through a maze, must recognise if it is located in one room rather than another. Here, the agent exploits environmental cues, such as navigating through subsequent corners of the maze, and fine-tuned time-dependent structures to take the correct decision. Environmental structures (regularities) are at the basis of the recognition process performed by the agent's controller during the exploration of the maze.

The difference between the ecological models and our study is not as apparent as it was for the non-ecological ones described at the beginning of the section. However, it should be noticed that, in the ecological studies reviewed above, the discrimination is based on the recognition of distinctive environmental contingencies and the maintenance of these experiences through time, as a form of short term memory. On the contrary, in our study, the cue which allows the agent to make the discrimination has to deal with the persistence over time of a perceptual state common to both of the elements to be distinguished—i.e., *Env.A* and *Env.B*—rather than with the nature of the cue itself employed to make the discrimination. That is, in our case, due to the nature of the agent's sensory apparatus, the two types of environment can be distinguished solely because a perceptual state, present in both environments, might be perceived by the agent for a longer time in one than in the other.

2.2 Integration Over Time

A general problem common to biology and robotics concerns the definition of the mechanisms necessary to decide when it is better to pursue a particular action in a certain location and at which moment in time it is better to leave for pursuing a similar or a different activity in a similar or different location. This problem is not limited to foraging alone, but it extends to many activities a natural or artificial agent is required to carry out. Autonomous agents may be asked to change their behavior in response to the information gained through repeated interactions with their environment. For example, in a group of robots, although many individual actions might be simpler to carry out than a single coordinated activity, they might result less efficient (see [62]). Therefore, autonomous agents require adaptive

mechanisms to decide whether it is better to pursue solitary actions or to initiate cooperative strategies. Also, an agent might need to communicate to the other members of the group some information it has gathered. The scenario described in Section 1.2 requires such a behavior. For example, the robots should find a way to the goal area, and to do so they have to explore their enviroment, which might contain holes or areas dangerous to traverse. We need a strategy which will allow such agents to decide if alternative strategies are required and thus trigger cooperation through communication of their experience.

One way to deal with the challenge described above is the design of decision-making mechanisms for —in our case— an *s-bot*, which integrates over time its perceptual experience in order to initiate alternative actions. In other words, the behavior of the agent should change as a consequence of its repeated interaction with particular environmental circumstances.

Nolfi *et al.* define agents that exploit internal representations as well as information directly available from their sensors and that are able to extract their internal representations autonomously by interacting with the environment, as agents that are able to integrate sensory-motor information over time. They rely on a mixed strategy in which basic sensory-motor mechanisms are complemented and enhanced with additional internal mechanisms and tend to rely on partial, action-oriented, and action-mediated representations of the external environment [46].

In this thesis we call upon the notion of internal representation, a very controversial issue in the literature. It can be more properly characterized as a description that is in the eye of the observer rather than as a formal property of an agent. We do not want to get into details in this subject, therefore we will resort on the more general notion of internal state. By internal state we mean a state (e.g. the activation state of an internal neuron of the control system of a robot) that might be affected by the previous sensory-motor states experienced by the robot and that co-determine, together with the current sensory states, the robot's motor actions. By mediating between perception and actions, internal states might allow agents to produce behaviors that are decoupled from the immediate circumstances while still remaining sensitive to them. We will use the definition introduced by Nolfi *et al.* in [46]. Thus, a reactive robot is a robot that does not have any internal state and for which the current motor action is only dependent of the current sensory state. On the contrary, a robot that relies exclusively on its internal dynamics is a robot in which sensory information coming from the external environment is either missing or not taken into account once the robot motor actions are determined. A very important observation

to be made is that a robot with a non-reactive controller can also exhibit reactive behavior.

Most of the experiments in evolutionary robotics rely on neural controllers. In many cases feed-forward neural networks are used. These networks are effective in producing reactive behavior but cannot deal with time, always reacting in the same way to the same sensory state and therefore cannot integrate information over time. In other cases recurrent neural networks have been used (see [18]). Other attempts have been conducted by using Continuous Time Recurrent Neural Networks (CTRNNs) [7]. By relying on differential equations instead of being updated at fixed time steps these networks can produce continuous dynamics. These networks have been successfully applied to a variety of tasks (such as legged locomotion [33] and visually guided navigation [29]). However the extent to which they can be applied to tasks that have sequential components and their ability to scale up is unclear. Other attempts have been conducted by using synaptic plasticity. In some cases the synaptic weights were updated by using reinforcement learning (see [1]) or back-propagation (see [54]) on the basis of self-generated teaching signals. In other cases synaptic weights were updated on the basis of genetically encoded hebbian rules (see [20]). In general terms, integration of information over time can be accomplished both by modifying the synaptic weights (through some form of plasticity) and by means of recurrent connections. In both cases in fact, the way in which individuals react to the current sensory state might be affected by the previous experienced sensory states. Different methods however might have different characteristics. For instance, the former approaches, by relying on gradient descent techniques, tend to produce small and long term effects on the robot behaviors whereas the latter approach, based on hebbian learning, might produce significant effects in the short term [20].

So, we can expect the emergence of systems able to integrate sensory-motor information over time and later use this information to modulate their behavior accordingly under certain conditions. First of all, as we discussed above, the agent should be equipped with the appropriate neural controller, able to display non-reactive behavior. But then, how do we distinguish between the tasks that require integration over time and those that don't? The border between what can be accomplished by simple agents that only rely on their current sensory states or on their internal dynamics and what can be accomplished by more complex agents that are also able to integrate information over time is rather fuzzy and cannot be formally identified. However, problems that should be accomplished in varying environmental conditions tend to require agents able to integrate information over time [46].

2.2.1 CTRNNs

Continuous Time Recurrent Neural Networks (CTRNNs) have been introduced in Evolutionary Robotics by Beer [6], and they are the reflection of a theoretical approach to cognition which aims to exploit the mathematical tools of dynamical systems theory to investigate issues of interest in adaptive behavior research. According to Beer, there are two fundamental principles which justify the use of the formalism of dynamical systems theory within the context of adaptive behavior. Firstly, since the fundamental nature of adaptive behavior in natural systems is to generate the appropriate behavior at the appropriate time, dynamical systems theory provides the required mathematical formalisms for the description and the analysis of systems whose behavior unfolds over time. Far from being inessential details, issues of rate and timing fundamentally matter to an embodied agent. For an embodied agent, time can make all the difference between an adaptive behavior and an unsuccessful one. Secondly, since in nature qualitatively similar patterns of behavioral dynamics are given rise by different combinations of underlying biochemical mechanisms, it looks plausible to consider adaptive behavior as generated by causal mechanisms which result from the dynamical interactions of elementary units such as cells or molecules, rather than generated by the dynamics of the single elementary units. Thus, the explanatory focus on any investigation on the causal mechanisms of adaptive behavior must look at the structure of this internal dynamics, rather than the behavior of the single elementary unit. The theoretical concepts and formalism that can best do justice of this dynamical nature are those of the dynamical systems theory. Continuous Time Recurrent Neural Networks (CTRNNs) represent a particular convenient way of instantiating a dynamical system to control the behavior of autonomous robots. CTRNNs differ from the classic connectionist artificial neural networks because each node within a CTRNN has its own state: i.e., the activation level, whose rate of change is specified by a time constant associated with each node. Furthermore, the nodes within the network are self-connected, as well as interconnected in an arbitrary way with each other. These two features allow the network to develop dynamical behavior in which the state of nodes alters the behavioral output of the system even if the sensory input remains constant.

As we mentioned in the previous section, a prerequisite to achieve integration over time is that the agent is equipped with the appropriate neural controller, able to display non-reactive behavior and rich internal dynamics. According to Beer [6], CTRNNs are an obvious choice for this work because

(1) they are arguably the simplest nonlinear, continuous dynamical neural network model; (2) despite their simplicity, they are universal dynamics approximators in the sense that, for any finite interval of time, CTRNNs can approximate the trajectories of any smooth dynamical system on a compact subset of $\Re^n$ arbitrarily well ;(3) they have a plausible neurobiological interpretation, where the state y is often associated with a nerve cell s mean membrane potential and the output $s(y)$ is associated with its short-term average firing frequency. CTRNNs are also being applied to a wide variety of other problems, including associative memories, optimization, biological modeling and many others. Since these networks will be the ones used in our work to achieve agents displaying non-reactive as well as reactive behavior, it is useful at this point to give the mathematics that describe their behavior.

Continuous-Time Recurrent Neural Networks are networks of model neurons of the general form:

$$\frac{dy_i}{dt} = \frac{1}{\tau_i}\left(-y_i + \sum_{j=1}^{N} \omega_{ji}\sigma(y_j + \beta_j) + I_i\right), \imath = 1, 2, ..., N, \sigma(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

where, using terms derived from an analogy with real neurons, y_i represents the cell potential, τ_i the decay constant, β_j the bias term, $\sigma(y_j + \beta_j)$ the firing rate, ω_{ji} the strength of the synaptic connection from neuron j^{th} to neuron i^{th}, I_i the intensity of the sensory perturbation on sensory neuron i.

2.3 The Tuci *et al.* Experiment : "Evolving the "feeling" of time through sensory-motor coordination: a robot based model"

The starting point of our experiments is the paper by Tuci *et al.* (for details see [65]). They designed decision-making mechanisms for an autonomous robot equipped with simple sensors, which integrates over time its perceptual experience in order to initiate a simple signalling response. Contrary to other previous similar studies, in this work the decision-making was uniquely controlled by the time-dependent structures of the agent's controller, which in turn, are tightly linked to the mechanisms for sensory-motor coordination. The results of this work showed that a single dynamical neural network, shaped by evolution, makes an autonomous agent capable of "feeling" time through the flow of sensations determined by its actions. Further analysis

of the evolved solutions revealed the nature of the selective pressures which facilitate the evolution of fully discriminating and signalling agents.

Their experiments required an autonomous agent to posess both navigational skills and decision-making mechanisms. That is, the agent should prove capable of navigating in a boundless arena in order to approach a light bulb positioned at a certain distance from its starting position. Moreover, it should prove capable of discriminating between two types of environment: one in which the light can actually be reached, and another in which the light is surrounded by a "barrier" which prevents the agent from proceeding further toward its target. Due to the nature of the experimental setup, the agent could find out in which type of environment it was situated only if it proved capable of (i) moving coordinately in order to bring forth the perceptual experience required to discriminate between the two environments; (ii) integrating over time its perceptual experience in order to initiate a signalling behavior if situated in an environment in which the light cannot be reached.

The results of their simulations showed that a single Continuous Time Recurrent Neural Network controller shaped by evolution, makes an autonomous agent capable of "feeling" time through the flow of sensations determined by its actions. In other words, the controller allows an agent to make coordinated movements which bring forth the perceptual experience necessary to discriminate between two different types of environment and thus to initiate a simple signalling behavior. Low level "leaky-integrator" neurons, which constitute the elementary units of the robot's controller, provide the agent with the required time-dependent structures.

At this point we are going to present their work in detail, since as we already mentioned above, it is the starting point for our train of thought and experiments.

At the beginning of each trial, a simulated Khepera robot is positioned within a boundless arena, at about 100 cm west of a light bulb, with a randomly determined orientation chosen between north-east and south-east (see Figure 2.1 left). The light bulb is always turned on during the trial. The robot perceives the light through its ambient light sensors, positioned 45 degrees left and 45 degrees right with respect to its heading. Light levels alter depending on the robot's distance from the light. The colour of the arena floor is white except for a circular band, centered around the lamp, within which the floor is in shades of grey. The circular band covers an area between 40 cm and 60 cm from the light; the floor is black at exactly 40 cm from the light; the grey level decreases linearly with the distance from the light. The robot perceives the colour of the floor through its floor sensor,

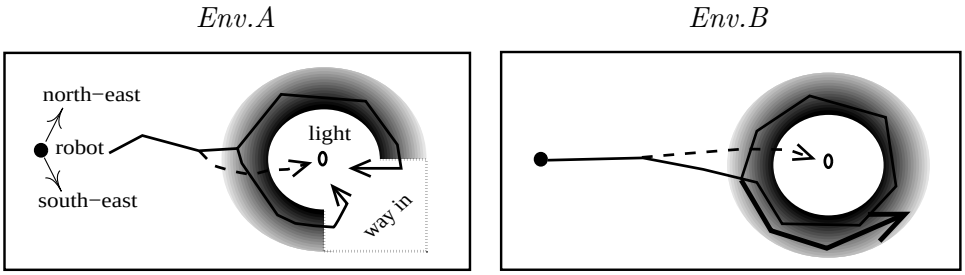

Figure 2.1: Depiction of the task. The small black circles represent the robot at starting position. The small empty circles represent the light bulb. The arena floor is white everywhere except within a circular band surrounding the light. The *way in* zone corresponds to the sector of the band, indicated by dotted lines, in which the floor is white. In both pictures, the continuous arrows are examples of good navigational strategies; the dashed arrows are examples of forbidden trajectories. In *Env.B*, the continuous arrow gets thicker to indicate that the robot emits a sound after having made a loop around the light.

positioned on its belly, which outputs a value scaled between 0—when the robot is positioned over white floor—and 1—when it is over black floor.

The robot can freely move within the band, but it is not allowed to cross the black edge. The latter can be imagined as an obstacle or a trough, that prevents the robot from further approaching the light (see dashed arrows in Figure 2.1). Whenever the robot crosses the black edge, the trial is unsuccessfully terminated. The area in shades of grey is meant to work as a warning signal which "tells" the robot how close it is to the danger—i.e., the black edge.

There are two types of environment. In one type—referred to as *Env.A*—the band presents a discontinuity (see Figure 2.1, left). This discontinuity, referred to as the *way in* zone, is a sector of the band in which the floor is white. In the other type—referred to as *Env.B*—the band completely surrounds the light (see Figure 2.1, right). The *way in* zone represents the path along which the robot is allowed to safely reach the light in *Env.A*. A successful robot should prove capable of performing phototaxis as well as looking for the *way in* zone to avoid to cross the black edge of the band. Such a robot should always reach the light in *Env.A*. On the contrary, in *Env.B* the robot should, besides avoiding to cross the black edge, signal the absence of the *way in* zone by emitting a tone. So, to summarise the

task the agent has to perform , he should distinguish between environments in which the band presents a discontinuity (i.e., *Env.A*) and environments in which the band does not presents any discontinuity (i.e., *Env.B*), while provided only with local information.

The cue the agent should use is a temporal one: that is, the *Env.B* can be "recognised" by the persistence of a particular perceptual state for the amount of time necessary to discover that there is no *way in* zone. For example, a successful agent might integrate over time the grey level sensed by its floor sensor to bring forth something similar to the "feeling" of being travelling within the band for as long as the time required to complete a loop. Such a strategy would allow the robot to make sure that there is no *way in* zone. Alternatively, the robot might simply react to the colour of the floor and integrate over time the perceived light intensity. In this case, the perception of the circular band is simply used to interrupt the phototaxis and to initiate a circular trajectory.

Notice that, whatever is the nature of the perceptual state that the robot integrates over time, the underlying mechanisms for the integration are strongly dependent on the way the robot moves within the environment. For example, let's assume that our robot, by circuiting around the light while remaining on the circular band, integrates over time the reading from the floor sensor. By employing this strategy, the amount of time required for our robot to perform a complete loop of the band depends on the dimensions of the band and on the way in which the robot moves within the band. The robot movements—e.g., its speed and trajectory—are determined by its controller. Thus, the latter should make the robot move in such a way that, if the perception of the band lasts for a certain amount of time, the following conclusions can be drawn: (i) the band does not present any discontinuity; (ii) the sound signalling must be activated. In other words, the agent should prove capable of moving in such a way that its flow of perception is informative enough to allow it to "feel" time and consequently to make a correct discrimination, through sound signalling, between *Env.A* and *Env.B*.

The difficulty of this experiment is twofold: on the one hand it resides in synthesising, through an evolutionary process, a robot's controller which must be capable of moving the robot coordinately so that it can integrate over time the flow of perception determined by the robot's actions. On the other hand, evolution must find a way to combine within a single—i.e., not modularised—controller the mechanisms required for sensory-motor coordination and discrimination through sound signalling.

At this point it would be beneficial to argue why this experiment requires integration over time, why it is a non-reactive task. The robot will have

to discriminate between the enviroments by "feeling" the time it has been travelling on the circular band, and then initiate a signalling behavior. So it has to encode in its internal state somehow this time travelling, therefore the task is non-reactive. And yet, we cannot guarantee that this task would never be solved by a purely reactive agent. Imagine an agent that can move in circles with gradually decreasing radius around the black band. It could signal, once it feels a certain grey level, information available by its floor sensor. This agent of course is considered a lucky one, and the case described here is so extreme that we can disregard it. After all, one has to a priori *design* this behavior.

In the following section, we will present the details concerning the robot-environment simulation model used by Tuci *et al.* to evolve the controllers (see section 2.3.1), the equation used to update the state of the neural network (see section 2.3.2), the parameters of the genetic algorithm (see section 2.3.3), the evaluation function used and a short analysis of the results of this first experiment they conducted. It is important to refer to in detail to all the above parameters, because most of them are going to be used in our experiments.

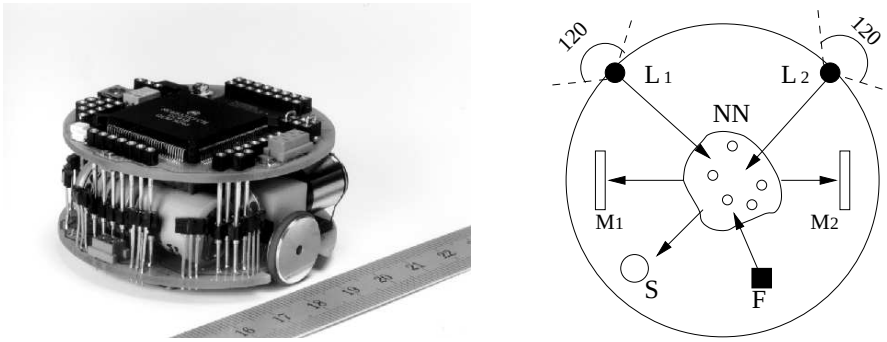

Figure 2.2: A picture of a Khepera robot on the left. Plan of the robot on the right, showing sensors and motors. The robot is equipped with two ambient light sensors (L_1 and L_2) and a floor sensor indicated by the black square F. The left and right motor (M_1 and M_2) are controlled by a dynamic neural network (NN). A simple sound signalling system, controlled by an output of the network, is referred to as S.

2.3.1 The simulation

The robot and its world were simulated using a modified version of the "minimal simulation" technique described by Jakobi in [32]. Jakobi's technique uses high levels of noise to guarantee that the simulated controller will transfer to a physically realised robot with no loss of performance. Their simulation models a Khepera robot, a 55 mm diameter cylindrical robot (see Figure 2.2). This simulated robot is provided with two ambient light sensors, placed at 45 degrees (L_1) and -45 degrees (L_2) with respect to its heading, and a floor sensor positioned facing downward on the underside of the robot (F). The light sensors have an angle of acceptance of 120 degrees. Light levels change as a function of the robot's distance from the lamp. The light sensor values are extrapolated from a look-up table which corresponds to the one provided with the Evorobot simulator (see [43] for further details). The floor sensor can be conceived of as a proximity infra-red sensor capable of detecting the level of grey of the floor. It produces an output which is proportional to the level of grey, scaled between 0—when the robot is positioned over white floor—and 1—when it is over black floor. The sound signalling system is represented by the binary output of one of the neurons of the robot's controller (see Section 2.3.2 for details).

The implementation of the simulator, as far as it concerns the function that updates the position of the robot within the environment, closely matches the way in which Jakobi designed his minimal simulation for a Khepera robot within an infinite corridor (see [32] for a detailed description of the simulator). The robot has right and left motors—respectively M_1 and M_2—which can move independently forward or backward, allowing it to turn fully in any direction.

2.3.2 The controller

Fully connected, eight neuron Continuous Time Recurrent Neural Networks (CTRNNs) are used. All neurons are governed by the state equation 2.1, with N = 8. Three neurons receive input (I_i) from the robot sensors. These input neurons receive a real value in the range [0,1], which is a simple linear scaling of the reading taken from its associated sensor[1]. The other neurons do not receive any input from the robot's sensors. The cell potential (y_i) of the 6^{th} neuron, mapped into [0,1] by a sigmoid function (σ) and then set to 1 if bigger than 0.5 or 0 otherwise, is used by the robot to control the sound

[1]Neuron N_1 takes input from the ambient light sensor L_1, N_2 from the ambient light sensor L_2, N_3 from the floor sensor F.

signalling system. The cell potentials (y_i) of the 7^{th} and the 8^{th} neuron, mapped into [0,1] by a sigmoid function (σ) and then linearly scaled into [-10,10], set the robot motors output. The strength of synaptic connections ω_{ji}, the decay constants τ_i, the bias terms β_j, and the gain factor g are genetically encoded parameters. Cell potentials are set to 0 any time the network is initialised or reset, and circuits are integrated using the forward Euler method with an integration step-size of 0.2 seconds.

2.3.3 The evolutionary algorithm

A simple generational genetic algorithm (GA) is employed to set the parameters of the networks [25]. The population contains 100 genotypes. Generations following the first one are produced by a combination of selection with elitism, recombination and mutation. For each new generation, the three highest scoring individuals ("the elite") from the previous generation are retained unchanged. The remainder of the new population is generated by fitness-proportional selection from the 70 best individuals of the old population. Each genotype is a vector comprising 81 real values (64 connections, 8 decay constants, 8 bias terms, and a gain factor). Initially, a random population of vectors is generated by initialising each component of each genotype to values chosen uniformly random from the range [0,1]. New genotypes, except "the elite", are produced by applying recombination with a probability of 0.3 and mutation. Mutation entails that a random Gaussian offset is applied to each real-valued vector component encoded in the genotype, with a probability of 0.15. The mean of the Gaussian is 0, and its standard deviation is 0.1. During evolution, all vector component values are constrained to remain within the range [0,1]. Genotype parameters are linearly mapped to produce CTRNN parameters with the following ranges: biases $\beta_j \in$ [-2,2], weights $\omega_{ji} \in$ [-6,6] and gain factor $g \in$ [1,12]. The genes which codify the decay constants are firstly linearly mapped onto the range $[-0.7, 1.7]$ and then exponentially mapped into $\tau_i \in [10^{-0.7}, 10^{1.7}]$.

2.3.4 The experiment - The evaluation function

In this section we illustrate the fitness function and the results of a first series of experiments in which they evolved agents capable of discriminating between *Env.A* and *Env.B*. The fitness function employed does not simply reward a robot for approaching the light bulb and for signalling anytime it is located in *Env.B*. A significant feature of this fitness function is that it rewards agents that make use of their sound signalling system at the point

where it is required.

During the evolution, each genotype is coded into a robot controller, and is evaluated 40 times—20 times in *Env.A* and 20 in *Env.B*. At the beginning of each trial, the neural network is reset—i.e., the activation value of each neuron is set to zero. Each trial differs from the others in the initialisation of the random number generator, which influences the robot starting position and orientation, the position and amplitude of the *way in* zone, and the noise added to motors and sensors. For each of the 20 trials in *Env.A*, the position of the *way in* zone is varied to facilitate the evolution of robust navigational strategies. Its amplitude is fixed to $\frac{\pi}{2}$. Within a trial, the robot life-span is 80 s (400 simulation cycles). A trial is terminated earlier if either the robot crosses the black edge of the band (see dashed arrows in Figure 2.1) or because it reaches an Euclidean distance from the light higher than 120 cm. In each trial t, the robot is rewarded by an evaluation function f_t which corresponds to the sum of the following four components:

$$R_{\text{motion}} = \frac{d_f - d_n}{d_f} \qquad R_{\text{error}} = -\frac{p_b}{t_b}$$

$$R_{\text{near}} = \begin{cases} p_c/t_c & Env.A \\ 0 & Env.B \end{cases} \qquad R_{\text{signal}} = \begin{cases} 0 & Env.A \\ p_a/t_a & Env.B \end{cases}$$

R_{motion} rewards movements toward the light bulb: d_f and d_n represent respectively the furthest and the nearest Euclidean distance between the robot and the light bulb. In particular, d_f is updated whenever the robot increases its maximum distance from the light bulb. At the beginning of the trial, d_n is fixed as equal to d_f, and it is subsequently updated every time step when (i) the robot gets closer to the light bulb; (ii) d_f is updated. In this latter case, d_n is set equal to the new d_f.

In *Env.A*, d_n is set to 0 if the robot is less than 7.5 cm away from the light bulb. In *Env.B*, d_n is set to 0 if the robot makes a complete loop around the light bulb while remaining within the circular band.

R_{error} is negative to penalise the robot for (i) signalling in *Env.A*, and (ii) signalling in *Env.B* before having made a loop around the light: p_b is the number of simulation cycles during which the robot has erroneously emitted a tone, and t_b is the number of simulation cycles during which the robot was not required to signal.

R_{near} rewards movements for remaining close to the light bulb: p_c is the number of simulation cycles during which the robot was no further than 7.5 cm away from the light bulb in *Env.A*, and t_c is the robot life-span. In *Env.B* the robot cannot get closer than 40 cm to the light, therefore, this component is equal to 0.

R_{signal} rewards signalling in *Env.B*: p_a is the number of simulation cycles during which the robot has emitted a tone after having made a loop around the light, and t_a is the number of simulation cycles during which the robot was required to emit a tone. In *Env.A*, this component is always set to zero. Recall that the robot is also penalised for crossing the black edge of the band and for reaching a distance from the light higher than 120 cm. In these cases, the trial is ended and the robot's fitness is computed by considering the current state of the system.

2.3.5 Results

Twenty evolutionary simulations, each using a different random initialisation, were run for 6000 generations. The best individual of the final generation from each of these runs was examined in order to establish whether they evolved the required behaviour.

During re-evaluation, each of the twenty best evolved controllers was subjected to a set of 100 trials in *Env.A* and a set of 100 trials in *Env.B*. At the beginning of each re-evaluation trial, the controllers are reset. Each trial has a different initialisation. During re-evaluation, the robot life-span is 120 *s* (600 simulation cycles).

Firstly, the navigational ability of the best evolved robot in an *Env.A* was analysed. A successful robot should reach the light bulb going through the *way in* zone, without signalling. The results prove that almost all the best evolved robots employ successful navigational strategies which allow them to find the *way in* zone, and to spend between 40% and 80% of their life-time close to the target. According to Tuci *et al.* , the fact that some runs resulted slightly less successful than others, is due to a tendency to cross the black edge of the band. A qualitative analysis of the robots' behavior that they performed shows that, when the best evolved robots are situated in an *Env.B*, their navigational strategies allow them (i) to approach the light as much as possible without crossing the black edge of the band, and (ii) to make a loop around the light, between 40 cm and 60 cm from the light, following a trajectory nearly circular.

The agents were not evolved just to navigate properly toward the light, but also for accurately discriminating between the two types of environment. Recall that the agents are required to make their choice by emitting a tone only if they "feel" they have been situated in an *Env.B*. None of the best evolved robots emited a tone if situated in *Env.A*. On the contrary, their success in evolving robots emitting sound when in *Env.B* was not as high, since only approximately half of the robots were behaving as expected.

The quality of the signalling behavior can be established with reference to the amount of error of type I (*Err.I*) and error of type II (*Err.II*) made by the successful robots. The *Err.I* refers to those cases in which the robot emits a tone **before** having made a loop around the light. The *Err.II* refers to those cases in which the robot emits a tone **after** having completed the loop. *Err.I* can be considered as a false positive error—i.e., signalling that there is no *way in* zone when there may be one. *Err.II* can be considered as a false negative error—i.e., not accurately signalling that there is no *way in* zone. Both types of error are calculated with respect to the angular displacement of the robot around the light from the starting position—the position at the time when the robot enters into the circular band—to the signalling position—the position at the time when the robot starts signalling.

If the robot makes no errors, this angle is 2π. It is obvious that the bigger the deviation from this value, the less reliable the signalling mechanism. Of course, a robot that signals less than $\frac{\pi}{2}$ radians before the full circle, is far more succesful than one that signals after the completion of the loop. This follows from the fact that the maximum distance on the black band a robot can cover in *Env.A* is $\frac{3\pi}{2}$, so having been travelling more on the band would mean that it is in *Env.B*. It is of course very difficult to make no errors—i.e, emitting a tone precisely at the time in which an entire loop around the light is made. Tuci *et al.* consider successful an agent that, in order to signal the absence of the *way in* zone, manages to reduce the amount of errors of both types. Most of the robots that manage to signal have average errors bigger than 20 degrees.

The mechanisms that the successful robots employ to solve the discrimination task are tuned to those environmental conditions experienced during evolution. So, they do not properly work if the environment changes. For example, in some complementary experiments they observed that both the reduction and the increment of the distance between the black edge of the band and the light disrupt the robot's performance: the smaller the distance, the bigger the *Err.II*—i.e., signalling after having made a loop around the light; the higher the distance, the bigger *Err.I*—i.e., signalling before having made a loop around the light. There was only one run that resulted in an agent integrating both the perception of the floor and the intensity of the light, but the relationship between these two sensory inputs had a bearing on the emission of the tone. So, for a given level of grey, the higher/lower is the intensity of the light the shorter/longer is the time it takes to the robot to emit a tone. Tuci *et al.* suggest that the artificial neural networks turned out to be capable of tracking significant variations in environmental conditions—i.e., the relationship between the intensity of the light and levels

of grey of the floor.

2.3.6 Discussion and Conclusions

Tuci *et al.* have indeed shown that a single dynamic neural network can be synthetised by evolution to allow an autonomous agent to make coordinated movements that bring forth the perceptual experience necessary to discriminate between two types of environments. The results illustrated in [65] are indeed of particular interest because, contrary to other previous similar studies, in this work the decision-making is uniquely controlled by the time-dependent structures of the agent's controller, which in turn, are tightly linked to the mechanisms for sensory-motor coordination.

The significance of their results is twofold: on the one hand, they bear upon the significance of CTRNNs as controllers for autonomous robots. That is, these results prove that, despite the complexity of the task, in which mechanisms for sensory-motor coordination and for discrimination must be tightly linked, CTRNNs can be easily shaped by evolution to bring forth complex reactive and non-reactive mechanisms within a single non-modularised controller. On the other hand, these results bear upon the significance of the evolutionary approach to robotics. That is, they suggest that the evolutionary approach to robotics is a suitable methodological tool to develop adaptive autonomous agents which, like natural systems, can cope with unexpected circumstances—that is, environments never encountered by the agents' ancestors during the evolutionary phase. From an engineering point of view, this is a particularly desirable property to observe in autonomous systems, since it represents a way to successfully overcome the limitations of other more classic approaches to robotics (for more on this issue see [10, 11, 28, 67]).

The reason we chose to give such a detailed presentation of this work is that it served as starting point and motivation to our work. It was very challenging to see if with the required modifications, we could port this work in the context of the SWARM-BOTS project. As mentioned, their work has been carried out with the Khepera robot and a simulator designed for it. If we want to make use of their results within the SWARM-BOTS project, significant work has to be done. Also, given the fact that these experiments can be extended in a lot of different ways, it must be clear why we chose to continue beyond them. Last but not least, it is always challenging to try to port the results of a simulation to a real robot—in our case the *s-bot*. But in order to do so, there remained a lot of work to be done concerning the reliability of the sensors and changes that we have to come up with in order to apply these methods to another robot than the Khepera.

Even though these experiments provided us with useful results and inspiration, there is an important deal of criticism to be addressed.

Firstly, as mentioned above, Minimal Simulation relies on the idea of applying big amounts of noise on the sensors to ensure portability of the evolved controller to a real-world enviroment. Obviously, it is very challenging to try to do so, and reality can be some times disappointing. The results of the experiments conducted were only tested on a 2D simulator that did not take into account forces and dynamics. It would be very interesting to see if there is any mismatch between the behavior of the evolved controllers in a 2D environment and a much-more realistic 3D physics-based simulator.

Secondly, although noise is added on sensorial information concerning the ambient light sensors, no noise was present on the floor sensor. Therefore, the information provided is idealistic and it is very probable that if we try to port these results to a realistic enviroment we will fail. Imagine a scenario where the robot is not allowed to touch the black edge and in case this happens, severe damage will be inflicted upon the robot. In order to ensure that the robot will always avoid critically approaching the black band, thus making it more robust, we have to introduce noise on the sensor encoding the status of the circular band.

Furthermore, the only variation in environmental circumstances encountered by the robot during the evolution, is a variation of the position of the *way in* zone in *Env.A*. This ensures that the robot will have to look for the way in and thus ensures the emergence of integration over time. On the contrary, there is no variation that would ensure adaptability of the controller in novel circumstances, not encountered by the robot's evolutionary ancestors. So when the robot was placed—during post-evaluation—at a distance significantly different than the distance to the light for which it had been evolved, the results were disappointing, since they found out that only one of the evolved controllers proved robust to variation in the environmental conditions without being explicitly evolved for this. We believe that the adaptability and the robustness of a controller is of utmost importance, so despite their encouraging results in this innovative experiment, some modification must be made in order to achieve more robust agents, that could potentially exhibit a good behavior even in circumstances not encountered by their evolutionary ancestors.

We believe that the cause of this potential lack of adaptability is primarily the design of the evaluation function, described in detail in Section 2.3.4. This function, being quite complicated and very explicit concerning the "score" it attributes to the different behaviors, does not allow evolution to freely explore the space of potential solutions. In our view it would be

better to employ a simpler fitness function that could lead us to more robust solutions. Also, the fact that they used the same number of evaluations for the two enviroments could be a factor that deteriorated the performance of the controllers. The number of evaluations in the two enviroments could be used as a parameter to tune the system. Given the fact that the robot travels around the band in a circular trajectory trying to find a *way in* zone is due to the *Env.A* conditions, it could be the case that if this enviroment was encountered more during the evolution, a better behavior could have been evolved. This argument is strengthened by the fact that a lot of non-succesful agents were crossing the black edge. Arguably, with a bigger proportion of *Env.A* during evolution, the robots could result being more prudent. Also, varying the width of the *way in* zone can result in more robust agents.

Finally, the signalling behavior could serve as a communication signal to other robots, that could alter their current status or action. It could even serve as a starting point for altering the current action of the signalling robot. This is an obvious extention to their experiment, since the way it is presented, the signalling behavior is just a sign that the robot has correctly discriminated between the two environments.

The critisisms formulated above will be incorporated as changes in our replication of their experiment, in order to put it in the context of the SWARM-BOTS project, described in the following chapter. These changes will also be used in Chapter 4, where we extend their experiments in a collective robotics scenario.

Chapter 3

Replicating Tuci *et al.* in SWARMBOTS3D

In this chapter we present in detail the setup we used for our experiments. In particular, we deal with the porting and replication of the Tuci *et al.* experiment for the *s-bot* simulator SWARMBOTS3D. We introduce the notion of Minimal Simulation and its applicability to our task. Finally, we present the results of the replication.

3.1 Methodological Issues

As we already mentioned in the previous chapter, the Tuci *et al.* experiment was designed for a Khepera robot. In order to use the results of this experiment for the SWARM-BOTS project, we need to rerun the experiments with the simulated *s-bots*, using the SWARMBOTS3D simulator, a 3D physics-based simulator described in detail in Section 3.2.

Of course, the successful transfer to a 3D physics-based enviroment will be guaranteed only if we **evolve** the controllers in this enviroment. This work is presented in Section 3.3. Although this methodology will ensure success, it introduces a time constraint, since this method is very slow. Our task does not require the use of dynamics to be carried out, that is the robot does not have to feel forces to perform the task successfully. Therefore, a very fast method to produce results would be to use once again the Minimal Simulation approach, as Tuci *et al.* did. So, all the parameters used for the Khepera robot simulator have to be changed and adapted for the *s-bot*. To be more precise, we had to use the look-up tables for sensorial inputs available for the *s-bot* and to adapt position update functions with respect

to the *s-bot* geometry. Also, their experiment was developped using a 2D simulator which did not take into account dynamics and forces. In order to make it more realistic, we had to **test** the evolved controllers on a simulator modelling 3D environments, thus bridging the gap between simulation and reality. This is the most basic step we have to follow if we want to test our controllers with real robots. This task was not trivial since we had to come up with a way to compensate the lack of dynamics in the evolution. As we will show later, the use of noise and more specifically pink noise[1] on all sensors, motor actuators and position produced robust solutions which were able to generalise in 3D enviroments with forces. As Jakobi states in [32], the robot does not have to move identically in simulation and reality, but it has to satisfy some criteria we define in order to be characterised as useful. Following the same rationale, we can use the criteria of signalling, avoiding errors, moving coordinately around the black band and distinguishing between the two environments to decide if the porting to the 3D simulator has been successful and avoid the demand of an identical behavior. Therefore, in all our subsequent experiments, we had to make some changes in the methods used by Tuci *et al.* in order to achieve a more robust and adaptable behavior. Namely, we introduced noise on the floor sensor, pink noise on all sensors, variance in the *way in* zone's amplitude during evolution, different proportions of the two enviroments and we changed the evaluation function to a simpler form.

3.2 The SWARMBOTS3D Simulator

In this section, we describe the simplified *s-bot* simulation model we used in order to run evolutionary experiments, but also to test controllers evolved within the simple 2D simulator.

The mobility of the real *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. The chassis can rotate with respect to the main body (turret) by means of a motorized axis. The *s-bots* have two different ways of creating physical interconnections, rigid and semi-flexible. In our simulated model, since it was not needed, we did not make use of either the grippers or the rotating

[1] Pink noise is a way of "adding noise to the noise", in other words a way to make the noise non-systematic. Applying pink noise ensures that the system will not "get used" to the noise characteristics.

chassis.

We have used 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 3.1). For this purpose, we relied on the Vortex^{TM} SDK, which offers the necessary functionalities to develop accurate 3D 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*.

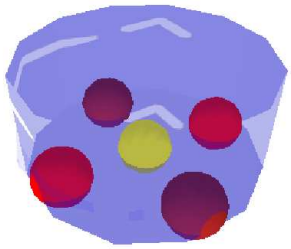

Figure 3.1: The simulated *s-bot* model. The body is transparent to show the chassis (center sphere), the motorized wheels (lighter spherical wheels) and the passive wheels (darker spherical wheels). The position of the virtual gripper is shown with an arrow painted on the *s-bot*'s body. On the contrary, 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. Furthermore, since we do not deal with tasks requiring connections between the robots, and in an effort to speed up the simulator as much as possible, we reduced all geometry by a factor of two, thus using a smaller and faster *s-bot* model (see Figure 3.2), which is able to capture all the properties of the normal-size model.

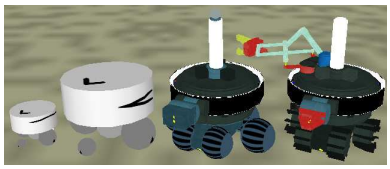

Figure 3.2: The 4 simulated *s-bot* models. From left to right we have the 4 models with ascending simulation detail. Our model is the left-most one and the most detailed one which models very closely the real *s-bot* is the right-most one.

3.2.1 Sensor, Actuator and Network 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. 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 6.5 rad/s. The maximum torque to be applied is set to 0.2 Nm. The desired angular speed is ω_t is defined as

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

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.

The first experiment we conducted was the replication of the experiment performed by Tuci *et al.* for the *s-bot*. To do so, we made use of the same sensors, actuators and neural architecture. We refer the reader to Sections 2.3.1 and 2.3.2 for details.

When defining the task for the SWARMBOTS3D simulator, we used as L_1 the average of the light sensors 0 and 1 of the *s-bot* model provided by SWARMBOTS3D and as L_2 the average of light sensors 6 and 7. To explain more in detail, the sensors 0 and 7 are placed at the left of the heading of the robot, at 22.5 and 70 degrees. Similarly, the sensors 6 and 7 are placed to the right, so at -22.5 and -70 degrees, respectedly. Therefore, we chose to average these values in order to create the equivalent of 2 sensors placed at 45 and -45 degrees, as used in Tuci *et al.*. This time, the light sensors values were extracted from a look-up table extrapolated from the SWARMBOTS3D simulator. To model the floor sensor we chose to introduce a 'false' sensor with the properties defined in Section 2.3.1. One obvious choice would be to use one of the infrared proximity sensors, that would be pointing to the floor, but this is not yet implemented and is possible future work which would render the porting of the evolved controllers on the real robot more feasible. The wheel actuators as well as the sound actuator were the ones provided by the SWARMBOTS3D model. The sound actuator was one of the three directional microphones mounted on the *s-bot* turret. Figure 3.3 presents them in detail. The sound signalling system is represented by the binary output of one of the neurons of the robot's neural controller. Once the neuron's output is bigger than 0.5, we consider this neuron activated. Concerning the update of the position of the robot, there was no need to use a look-up table since the update is done automatically by SWARMBOTS3D.

Noise of 5% was added as in [65] on top of the ambient light sensorial readings. This time, in order to make the controller more robust regarding the robot's capability of always efficienty avoiding the black edge, we also implemented a 10% noise on the readings of the floor sensor. This high amount of noise, although at first sight dangerous to fuzzify too much the evolution, would guarantee a more robust behavior. Imagine a setting where the robot will be destroyed if it touches the black band. It is thus of utmost importance to make the system as robust as possible. This was a definite drawback of the experiments conducted by Tuci *et al.*. We also added noise on the position of the robot, in order to minimize possible mismatches between its movement in the simple 2D enviroment and the complex physics-based enviroment of a 3D simulator. Finally, 10% noise was added on the motor actuators too, for the case of the Minimal Simulator approach.

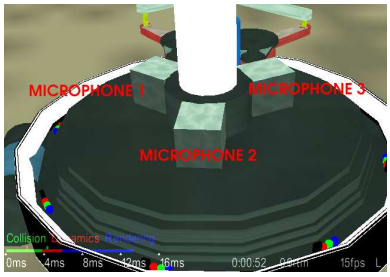

Figure 3.3: The three simulated microphones that an *s-bot* possesses

3.2.2 The Simulation and the new evaluation function

During the evolution, each genotype is coded into a robot controller, and is evaluated 15 times, 12 times in *Env.A* and 3 times in *Env.B*. We remind the reader that Tuci *et al.* in their experiment used a one-to-one proportion for the two environments. We introduced this change in order to obtain more robust controllers (see Section 2.3.6). At the beginning of each trial, the neural network is reset—i.e., the activation value of each neuron is set to zero. Each trial differs from the others in the initialisation of the random number generator, which influences the robot starting position and orientation, the position of the *way in* zone, and the noise added to motors and sensors. For each trial in *Env.A*, the position of the *way in* zone is varied to facilitate the evolution of robust navigational strategies. Its amplitude varies within the interval $[\frac{\pi}{6}, \frac{\pi}{2}]$. Within a trial, the robot life-span is 140 seconds (700 simulation cycles). The reason we increased this value in comparison to the 80 seconds used by Tuci *et al.* is that the *s-bot* due to different geometry and kinematics moves in a different way than the Khepera robot. A trial is terminated earlier if either the robot crosses the black edge of the band (see dashed arrows in Figure 2.1) or because it reaches an Euclidean distance from the light higher than 120 cm. For reasons we explained in Section 2.3.6, we changed the evaluation function to a simpler form. In each trial e, the robot is rewarded by function f_e which corresponds to the sum of the following two components:

1. R_{motion}—This component rewards movements toward the light bulb, and it is computed as:

$$R_{\text{motion}} = \frac{d_i - d_f}{d_i} \tag{3.2}$$

where d_i and d_f represent respectively the initial and the final Euclidean distance between the robot and the light bulb. In *Env.A*, d_f is set to 0 if the robot is less than 15 cm away from the light bulb. In *Env.B*, d_f is set to 0 as soon as the robot reaches the band in shades of grey.

2. R_{signal}—This component rewards agents that (i) do not signal anytime they are located in *Env.A*; (ii) emit a sound signal anytime they are located in *Env.B*. The component is computed as:

$$R_{\text{signal}} = \begin{cases} 1 & \text{if proper signalling} \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

An important feature of this evaluation function is that it simply rewards agents that make a proper use of their sound signalling system, without directly interfering with the nature of the discrimination strategies.

3.3 The Replication with SWARMBOTS3D

The first experiment conducted was evolving a controller able to perform integration over time, with SWARMBOTS3D, which means that the simulated enviroment also modelled forces between the enviroment and the robot and amongst the different counterparts of the robot itself. We aimed at replicating the Tuci *et al.* experiment. The main feature of this test was that the controller should be evolved within a physics-3D enviroment, while Tuci *et al.* evolved the robots in a 2D enviroment not taking into account dynamics.

Only one simulation was ran, and the desired behavior was evolved. The robot was most of the times able to successfully discriminate between the two environments, emitting a sound if situated in *Env.B*. Sometimes though it was crossing the black edge or not signalling at all. It would of course be interesting to have results with SWARMBOTS3D, and that for various reasons. Fistly, we could compare them to the ones acquired without using the latter simulator and possibly draw useful conclusions. Then, one could claim that evolving controllers in a realistic enviroment, much closer to reality than the one used in [65], would result in a more robust controller. Of course, this argument takes for granted that a simulator like SWARMBOTS3D can successfully be ported to the real *s-bots*, but there is not yet proof to support this argument. After all, we need to take into account time limitations and computational efficiency too, notions extremely important

in Evolutionary Robotics. One run of the experiment in SWARMBOTS3D demanded almost 3000 generations to evolve the required behavior which is translated to almost two weeks of six machines running in parallel. Considering the time limits for delivering this work, we decided to abandon this solution, and come up with a much faster solution to our problem. For these reasons, there will no results presented based on this method, except for just the fitness value during the evolution (Figure 3.4). The fitness value is scaled between 0 and 1. We can notice that at around 2700 generations the solution is found, but then there are a lot of fluctuations.

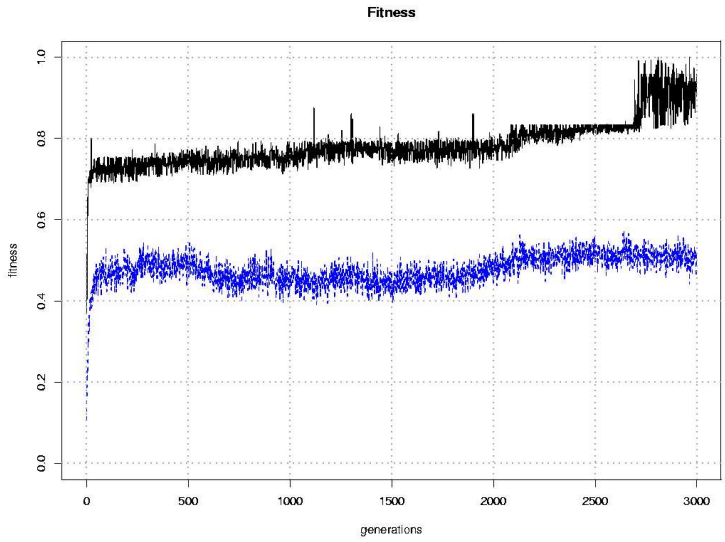

Figure 3.4: The Fitness during the evolution. The top thin line corresponds to the fitness of the best individual, while the dotted line refers to the average fitness of the population.

3.4 The Minimal Simulation Approach

Taking into account the fact that our task did not require use of dynamics, since no connections among dynamical bodies were needed, nor there was any dynamical interaction between robot and enviroment (i.e. rough terrain), we decided to use the Minimal Simulation approach for our experiment.

The robot and its world are once again simulated using a modified version of the "minimal simulation" technique described by Jakobi in [32]. Jakobi's technique uses high levels of noise to guarantee that the simulated controller will transfer to a physically realised robot with no loss of performance. Our hypothesis is that the controller will initially successfully transfer to the SWARMBOTS3D enviroment, when tested there. What we want to achieve is evolve the controller in a simple minimal environment, thus avoiding complicated dynamics relations which are not required according to the definition of our task, acquiring the solution quickly and easily. Our hypothesis will be confirmed if the robot behaves in a satisfactory manner, fulfilling the criteria of correctly signalling and finding the *way in* zone, avoiding to make errors of any type—like crossing the band, when the evolved controller is downloaded and tested on a physics-based simulator as SWARMBOTS3D. We will not demand that the robot moves identically in the two simulators, since that would be an extremely severe criterion, especially if the robot is able to carry out the task.

Therefore, we adapted the simulation of Tuci *et al.*, modelling the Khepera robot, so that it can model the fast *s-bot* model described in section 3.2.

The initial results were promising, but there was a big discrepancy in the behavior of the robot in the SWARMBOTS3D enviroment and in the 2D minimal simulator. Therefore, we had to come up with a way to compensate the lack of dynamics in the evolutionary environmant. Jakobi claims that high levels of noise are required in order to expect a satisfactory tranfer of the controller to reality. Having already used noise on all sensorial information and on the robot's position, we decided to also implement pink noise on them. Pink noise is a way of avoiding the case where a system would "learn" the characteristics of the noise and thus its effect on the robustness is reduced. In fact, what we do is ensure that the noise comes from different windows of the uniform distribution for each sensor and position. Also, there is a provision for long or short-term change to the sensor's or position's value.

Another big change we had to implement was to adapt the position updating to the *s-bot* geometry. Initially we made use of the same look-up table described in [32]—also used by Tuci *et al.*, just altering the parameters which refer to the robot's geometry and dimensions. Unfortunately the initial results were not that satisfying, so we decided to use the kinematics of a differential drive robot, as described by Dudek and Jenkin in [17].

The results were satisfactory in the sense that the robot was moving and behaving very similar to the Minimal 2D Simulator case. Of course, it is again of utmost importance to stress that we do not require the robot

to behave identically in the two simulators, especially for what concerns its movement. This is almost impossible to achieve since a physics-based environment contains interactions and dynamics that cannot be predicted.

The main lesson that we learned with the experimentation presented above is that if our task does not need physics to be achieved, we can use a Minimal simulation which can easily and fastly produce results, that can successfully transfer to a physics-based environment, maybe even reality, with the use of noise.

The experiment we ran with the methodology described above is a replication of the Tuci *et al.* experiment, with the Minimal Simulation approach. This allows us to spot possible differences in the results obtained, but is also very encouraging since the experiment was successful, in our effort to proceed and expand their experiment. The post-evaluation is done in both the Minimal 2D simulator and SWARMBOTS3D. Thus we will be able to infer some conclusions on the success of the porting to the physics-based 3D simulator.

3.5 Results

3.5.1 The Replication

We made twenty replications of the experiments. Figure 3.5 shows the fitness of the best individual and the mean population fitness plotted against the generation number (5000) and averaged over the 20 replications. We can notice that in all replications of the experiment a successful behavior was evolved[2]. The 100% success rate can be accounted for by recalling that the fitness function, not rewarding any specific action except phototaxis and the signalling behavior, has positively influenced the development of successful behaviors. In fact, evolution was left free to search for a strategy that could be effective for the achievement of the final goal. This was not the case with the evaluation function used by Tuci *et al.*, described in detail in Section 2.3.4. Their results report—after post-evaluation—12 out of 20 successful controllers. As we will show in Section 3.5.3, we obtained 18 out of 20 successful controllers, that is controllers able to perform the discrimination.

[2]The maximum fitness value is almost 2, since for some of the controllers, the final generation's achieved fitness value is very close to 2 but not exactly 2

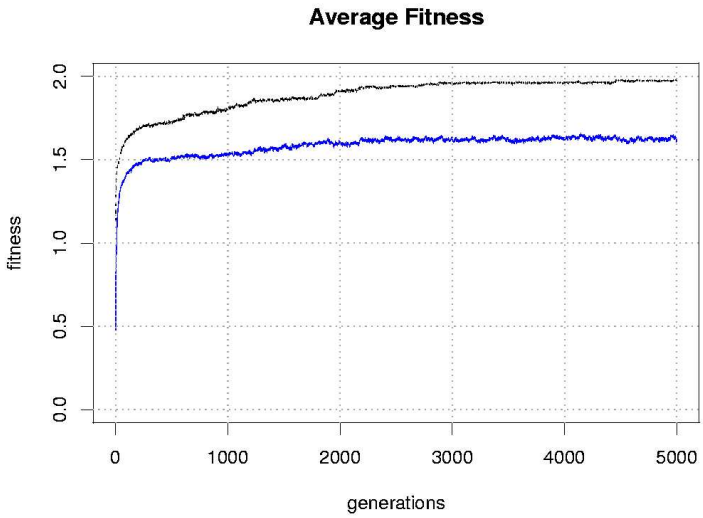

Figure 3.5: Average fitness during the evolution. All plots are the average over the 20 replications of the experiment. The top thin line corresponds to the average fitness of the best individual, while the dotted line below refers to the average fitness of the population.

3.5.2 Analysis of the evolved behavioral strategies

A qualitative analysis of the evolved controllers confirms that a number of different behavioral strategies have been obtained. However, some constant characteristics can be recognised. At the beginning of a trial, all robots perform phototaxis until they reach the circular band. When the grey level on the floor overcomes a certain threshold, the robots start circuiting around the light bulb with an approximately constant angular speed. Whenever the robots are placed in *Env.A* and the *way in* zone is detected, phototaxis starts again and the light bulb is reached. On the contrary, in *Env.B*, after travelling on the band for a given time without detecting the *way in* zone, the robots initiate a signalling behavior.

An example of this behavior is shown in Figure 3.6: in both *Env.A* and *Env.B*, it is possible to notice that, when the circular band is detected—see continuous line F at about simulation cycle 90—the robot starts moving on the circular band maintaining a constant distance from the light bulb. This behavior is indicated by the constant readings of the light sensors $L1$ and

$L2$ and of the floor sensors F. In *Env.A*, the *way in* zone is encountered shortly before simulation cycle 500, as indicated by the sudden drop in the floor sensor F. At this point, the robot performs phototaxis again, rapidly reaching the light bulb, as indicated by the high activation of the light sensors $L1$ and $L2$ at the end of the simulation.

The constant angular speed on the circular band is the basic mechanism exploited for discrimination between *Env.A* and *Env.B* by successfully evolved robots. In fact, this constant motion allows the robots to experience a constant perceptual state (the grey level of the floor and the light intensity that impinges on their sensors), which roughly corresponds to the constant flow of time. In Figure 3.6, one can notice that the persistence of a particular perceptual state, corresponding to the robot circuiting around the light and over the band, makes the output S, which controls the sound, increase linearly. This perceptual state triggers the sound signalling through an efficient integration mechanism which is based on the "feeling" of being travelling long enough over the circular band without having encountered the *way in* zone. In fact, if the *way in* zone is encountered, as in the upper part of Figure 3.6, the activation of the neuron S decreases below the threshold level 0.5. This response makes the robot capable of avoiding to initiate the signalling behaviour when it is not required. The situation is different in *Env.B*: the absence of the *way in* zone let the output of neuron S reach and overcome the threshold level 0.5—see bottom part of Figure 3.6, simulation cycle 550. This response makes the robot capable of correctly signalling that it is located in *Env.B*. It is also worth noticing that the output of the neuron S is rising in both enviroments, and around the same time in *Env.A* drops—manifestating the detection of a *way in* zone—and in *Env.B* rises above the threshold level. Finally, we can notice that the output of neuron S is initialised a bit below 0.5 and then drops linearly until the circular band in shades of grey is discovered, when it starts rising again. This proves that the feeling of time is dependent of the previous experience of the robot, that is of the distance it covered to reach the band. So in case it has to cover, for instance, a bigger distance to reach it, the value of the neuron's output will decrease further on, possibly causing the robot to signal later the absence of the *way in* zone, than it does for the initial distance.

In summary, the behavioral analysis revealed that the evolved controllers produce the required sensory-motor coordination that brings forth a perceptual state that is integrated over time and exploited for discrimination through sound signalling.

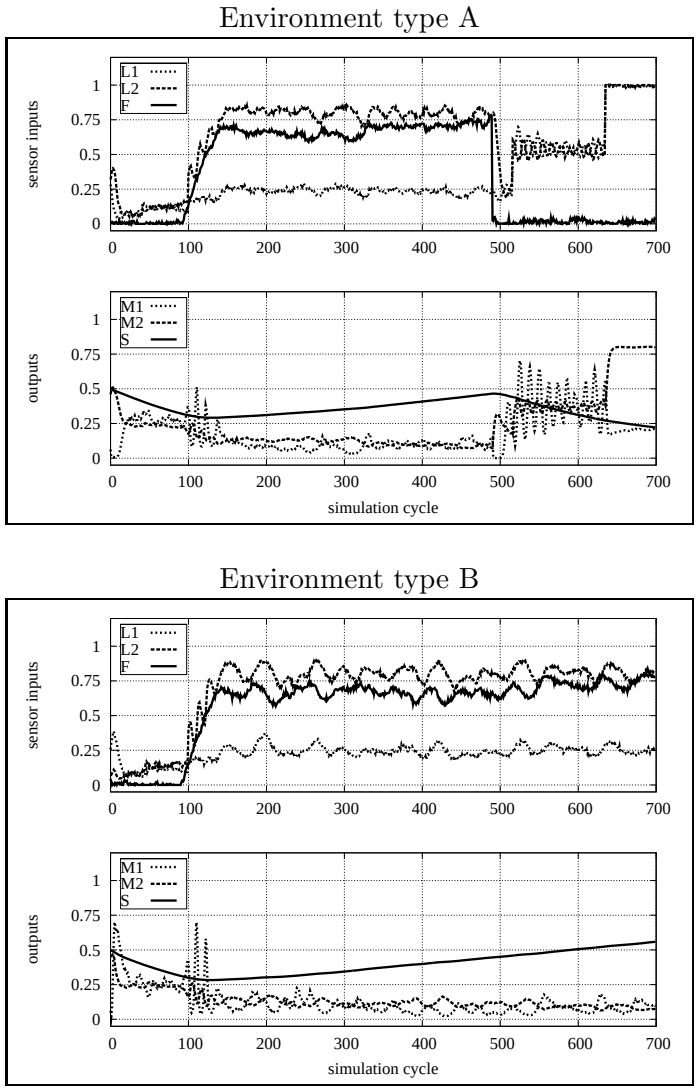

Figure 3.6: Behavioral analysis. The sensor activity and the corresponding motor output are plotted for 700 simulation cycles. L1 and L2 refer to the light sensors, while F refers to the floor sensor. M1 and M2 correspond to the motors of the two wheels, and S refers to the sound signalling. When S is bigger than 0.5, the robot emits a signal.

3.5.3 Post Evaluation in the Minimal Simulator Environment

In order to test the performance of the controllers evolved, it is required to perform a post evaluation. This is due to the possibility that the fitness of an

evolved individual is overestimated. If this is the case, the post evaluation will reveal it. Thus, we performed further analyses, by re-evaluating each of the best evolved final generation individuals for 100 trials in each type of environment (i.e., *Env.A* and *Env.B*). In each trial performed in *Env.A*, we look at the robot's capability to reach the light bulb (*Succ.*), without incurring in any error. Errors can be of two types: *E1* refers to the emission of a sound signal, while *E2* refers to crossing the black edge of the band. Similarly, in *Env.B*, we look at the performance of the robot on properly signalling the absence of the *way in* zone (*Succ.*), without committing any error. Also in this case, two error types are possible: *E3* refers to the lack of sound signalling, and *E4* refers to the robot crossing the black edge of the band. Furthermore, in *Env.B* we also compute the offset between the entrance position of the robot in the circular band and the position in which the robot starts to signal. This measure, called offset Δ, is computed as follows:

$$\Delta = |\alpha(t_e, t_s)| - 2\pi, \tag{3.4}$$

$$\alpha(t_1, t_2) = \sum_{t=t_1}^{t_2-1} \widehat{\mathbf{AOB}}, \qquad \mathbf{A} = \mathbf{X}_t, \mathbf{B} = \mathbf{X}_{t+1} \tag{3.5}$$

where $\mathbf{O}$ corresponds to the position of the light, and α is the angular displacement of the robot around the light from the starting position—the position at time t_e when the robot enters into the circular band—to the signalling position—the position at time t_s when the robot starts signalling. Angular displacement α is computed summing up all the convex angles $\widehat{\mathbf{AOB}}$ comprised between two consecutive position of the robot $\mathbf{X}_t$, taking into account that an angle is negative if the robot moves clockwise. This measure accounts for the capability of a robot for searching the *way in* zone. Offset Δ takes value 0 if the robot signals exactly after covering a complete loop of the circular band. Otherwise, it gives the angular displacement from this position. Negative values of the offset Δ suggest that the robot signals before having performed a complete loop, while positive values correspond to the situation in which the robot has performed more than one loop around the light, waiting too long to signal.

Table 3.1 refers to the post-evaluation results. As we can see, 17 out of the 20 controllers perform well, having a very high success rate in both *Env.A* and *Env.B*. In general, 18 out of the 20 controllers are able to perform the discrimination task. It is worth noting that two of the controllers that do not perform well, both fail to signal in *Env.B* (replications 12 and 18). The errors of the rest of the controllers are mostly signalling errors (*E1* and

E3), while some replications of the experiments have a higher error rate in crossing the black edge of the circular band. This is due mainly to a tendency of the robots to approach the black edge while circuiting on the band. Especially the controller in replication 11 has incured in this error lots of times, making this controller almost unsuccessful in *Env.B*. Concerning the offset Δ, most evolved controllers have a negative value, in general lower than 80 degrees, meaning that all robots signal before having completed one loop of the circular band. However, this offset can enough to discriminate between *Env.A* and *Env.B*, as the *way in* zone is up to 90 degrees wide. Only in one case, in replication 1 and 13, the robot is "prudent": that is, it signals only after having completed a loop around the light bulb. The only times the robot signals at an angle smaller than -90 degrees are replications 11 and 18, which as we mentioned above are not successful. It is important here to remind the reader that our fitness function described in detail in Section 3.2.2 does not reward signalling exactly after a full loop, as the one in Tuci *et al.* did. Yet, we managed to produce more successful controllers, with the changes we implemented on some methodological issues.

3.5.4 Robustness of the evolved solutions

It is very important to acquire controllers that are able to adjust to varying enviromental circumstances, that are able to display a satisfactory behavior even in circumstances not encountered in evolution. As we criticized in Section 2.3.6, Tuci *et al.* fail to produce controllers able to do so. Only one out of the twenty controllers is able to display some generalization. Therefore, we decided to test the best evolved controllers of the twenty runs in an experimental setup where the distance between the light source and the circular band varies between 20 and 60 cm, in *Env.B*. We measure the offset of the robot as the forementioned distance varies. The results are diplayed in Table 3.2. In general we can say that the closer to the light source is the circular band, the earlier the robot emits the sound. The distance that the robot has been travelling performing phototaxis until it reaches the black band definitely plays a role in its subsequent behavior. The feeling of time travelling in the band is **not** independent of the robot's previous experience, that is the time it travelled to arrive to the band. This is a result of network configuration and choice. Furthermore, if the band is positioned closer to the light, the light sensors of the robot once upon the band are more active. We expect thus a different behavior if the robot is placed in an enviroment different than the one in which it has been evolved. It is important to stress the fact that the robot has not been evolved with a fitness function that

Table 3.1: Post-evaluation. Performance of the ten best evolved controllers. The percentage of success (*Succ.* %) and the percentage of errors (*E1*, and *E2* in *Env.A*, and *E3*, and *E4* in *Env.B*,) over 100 trials are shown for both *Env.A* and *Env.B*. Additionally, the average offset Δ and its standard deviation (degrees) are shown for the environment type *Env.B*.

Post-Evaluation Results in the Minimal Simulation Environment								
run	*Env.A*			*Env.B*				
	Succ.	*E1*	*E2*	*Succ.*	*E3*	*E4*	*Offset* Δ	
	(%)	(%)	(%)	(%)	(%)	(%)	*Avg.*	*Std*
n. 1	78	0	1	100	0	0	16.40	45.03
n. 2	100	0	0	87	13	0	-8.15	47.90
n. 3	100	0	0	100	0	0	-47.86	9.83
n. 4	100	0	0	100	0	0	-78.99	14.58
n. 5	99	0	0	98	2	0	-28.89	17.44
n. 6	99	0	1	100	0	0	-71.84	25.91
n. 7	100	0	0	93	7	0	-19.97	15.62
n. 8	100	0	0	100	0	0	-28.41	15.49
n. 9	100	0	0	89	5	6	36.42	23.28
n. 10	100	0	0	100	0	0	-39.85	10.44
n. 11	100	0	0	54	8	38	-107.49	71.70
n. 12	89	0	0	0	100	0	-2.55	12.25
n. 13	100	0	0	98	2	0	30.24	19.72
n. 14	100	0	0	100	0	0	-21.40	12.55
n. 15	100	0	0	100	0	0	-1.16	22.41
n. 16	99	0	1	98	2	0	-0.62	37.90
n. 17	100	0	0	99	1	0	-58.19	25.39
n. 18	96	0	3	0	100	29	-102.57	57.72
n. 19	100	0	0	100	0	0	-36.10	7.95
n. 20	94	0	0	99	1	0	-68.15	19.32

would favor its signalling with zero offset.

We can notice that there are a lot of controllers that are able to display a good behavior, for a lot of distances, contrary to the results of Tuci *et al.* Of course, Tuci *et al.* present in their results smaller errors for the "bad performing" controllers, but we remind the reader here that our fitness function did not reward exact signalling. Another general remark is that in our results they almost all fail when the band is positioned very close to the light,

Table 3.2: Robustness analysis. Performance of the twenty best evolved controllers. The average offset of 100 evolutionary runs is given for the mentioned distances between the circular band and the light.

Robustness Analysis									
run	20	25	30	35	40	45	50	55	60
1	495.88	375.90	183.21	62.28	18.13	-29.89	-149.03	-216.52	-228.68
2	-336.70	-346.56	-42.39	37.12	-12.90	-54.54	-41.04	-64.81	-83.73
3	-258.39	-219.49	-62.47	-18.30	-53.66	-78.09	13.97	-9.04	-27.20
4	-297.15	-343.73	-193.25	-32.21	-82.33	-134.39	-69.30	-66.60	-85.06
5	-346.36	-340.60	-101.55	11.76	-26.96	-49.27	-68.22	-81.20	-91.74
6	-43.15	20.76	15.44	-33.93	-72.28	-62.15	-74.20	-98.83	-118.44
7	-57.75	-79.16	72.67	22.70	-15.93	-46.20	-56.97	-73.77	91.81
8	-159.55	-182.58	-47.90	24.34	-30.34	-67.88	-32.55	-50.43	-65.23
9	432.50	312.10	167.09	83.69	35.01	1.09	11.06	-15.64	-38.37
10	82.22	69.18	43.91	-1.05	-36.29	-42.02	-57.71	-80.44	-99.29
11	-322.44	-332.44	-124.27	-68.14	-106.32	-150.58	-36.48	-55.62	-73.28
12	-356.13	-355.99	-355.74	-167.97	-3.38	-34.69	-79.94	-175.94	-200.85
13	-175.53	-242.47	-155.63	79.48	30.53	-10.77	17.36	-25.29	-54.62
14	-253.24	-41.58	57.42	12.09	-22.38	-41.10	-90.80	-103.97	-115.43
15	-74.19	-126.56	56.72	32.08	-0.30	-28.22	-36.15	-56.88	-96.15
16	263.80	225.88	163.90	57.61	-2.86	-47.91	-123.25	-139.07	-151.85
17	-348.21	-345.60	-282.34	-11.84	-63.76	-106.50	-138.68	-180.32	-203.69
18	8.37	-118.91	-65.88	-92.18	-108.91	-130.78	-169.82	-337.55	-341.41
19	-344.55	-344.31	-318.16	-1.67	-43.29	-80.16	-32.54	-51.80	-68.30
20	-308.60	-318.03	-13.27	-30.96	-69.84	-87.39	-50.85	-66.83	-79.52

namely 20 or 25 cm away from it. On the other hand, they seem to perform very well in distances bigger than the one for which they were evolved (40 cm). Runs no. 2, 5, 7, 8, 10, 15, 19, 20 seem to perform well, especially for distances bigger than 40 cm. It is possible that when the band is positioned very close to the light, the light sensors activation is too high, affecting in a negative way the performance of the robot. What is also surprising is that sometimes the controllers perform better for distances other than 40 cm than for the latter distance. Finally, in Figure 3.7, we show the boxplot for run 20, one of the most successful ones.

3.5.5 Post Evaluation in SWARMBOTS3D

In this section, we present the results obtained by post-evaluating in SWARMBOTS3D the best individual of the final generation of run no. 14, one of the most successfull ones, as we can see in Table 3.1. The robot was 100% successful in both enviroments and had an average offset Δ of -21.40 degrees. As we mentioned in Section 3.1, the criteria for judging the quality of the porting from the one simulator to the other must be relevant to the

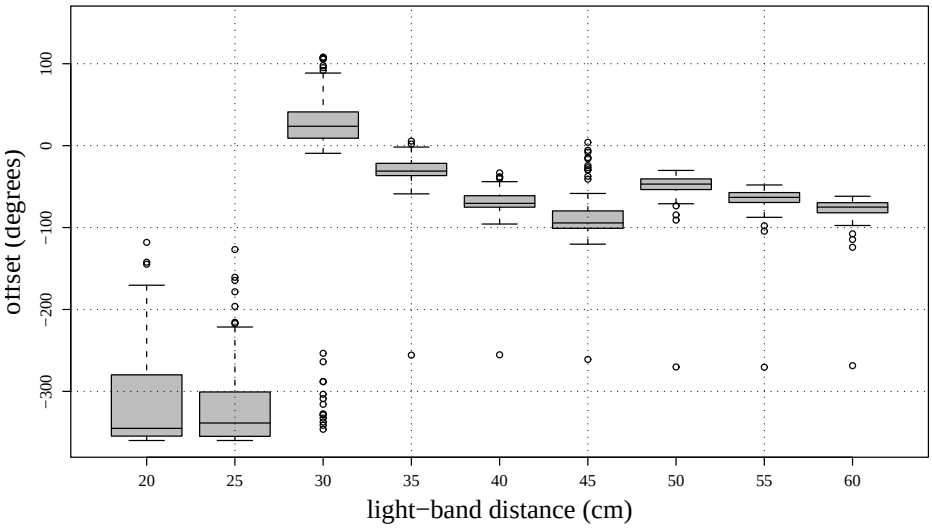

Figure 3.7: Robustness analysis for run no.20. The offset Δ is plotted for varying light-band distance. The box-plot shows 100 evaluations per box. Boxes represent the inter-quartile range of the data, while the horizontal bars inside the boxes mark the median values. The whiskers extends to the most extreme data points within 1.5 of the inter-quartile range from the box. The empty circles mark the outliers.

task the robot has to carry out. We must not demand the robot to have an identical behavior in the two very different simulators. Therefore, we will once again perform a post-evaluation where we extract the same information as in Section 3.5.3, namely in *Env.A* the robot's capability to reach the light bulb (*Succ.*), without incurring in any error, errors *E1* and *E2* and for *Env.B*, we look at the performance of the robot on properly signalling the absence of the *way in* zone (*Succ.*), without committing any error and the two possible error types, *E3* and *E4*. Furthermore, in *Env.B* we once again compute the offset Δ.

The initial evaluations were performed with exactly the same settings used in the Minimal Enviroment were the controller was also evolved. Therefore, we used the same initial orientation for the robot, thus between -120 and 120 degrees (see Figure 2.1) and the same amount of timesteps for the lifetime of the robot, so 700. The initial results were quite disappointing. What we found out was that due to acceleration, inertia and friction, the

robot lost a lot of time trying to orient itself correctly towards the light and then perform phototaxis. In fact, the robot moved towards the light bulb in both simulators with his back turned to the light. This is the solution evolution found for our problem. Therefore, placing the robot in a position *facing* the light, even with some variability, caused a spinning of the robot in order for the correct orientation to be found and then for phototaxis to be performed. Of course, this was also the case in the Minimal Simulator enviroment, but in this case, the acceleration, inertia and friction seemed to cause a very annoying delay in the beginning of the robot's lifetime. In order to confirm our hypothesis that the initial orientation matters, we conducted the same evaluation for three different configurations:

- orientation π: the robot is placed in a position facing the light, the most inappropriate for phototaxis
- orientation π+[-120,120]: the robot is placed in the original orientation
- orientation 0: the robot has its back turned to the light, having the most appropriate orientation for phototaxis.

As we can see in Table 3.3, our hypothesis was confirmed. The robot performed well only in the latter of the three cases, behaving extremely bad for the π orientation.

Looking at the results for the robot facing the light, we see that it was not able to signal in any of the 100 trials. Looking at the behavior of the simulated robot, it was obvious that because of the totally unfavorable initial orientation and the disruptive effect of dynamics in the phase where the robot tried to find a suitable orientation to perform phototaxis, the robot's lifetime was not enough for it to perform a complete loop around the light and thus discriminate between the two enviroments, thus emitting a sound. Therefore, we decided to perform some more experimentation, this time varying the timestep value, from the initial 700 value up to its double, 1400. As we can see in Table 3.3, the performance of the robot improves significally if its lifetime is increased above 800 timesteps, even for unfavorable initial orientations.

The conclusions we can draw from this post-evaluations is that it is possible to compare the results obtained in the two enviroments, but we have to be very careful because the influence of parameters that are not so important in the Minimal Simulation case, like the initial orientation, is affected by dynamics. Since that must be a value that is identical when we make a comparison between the two series of results, we have to give

the robot a longer lifetime if evaluated in the SWARMBOTS3D simulator. We can see that the performance is definitely comparable to the one in the Minimal Simulator case. What is also astonishing is the fact that the robot **never** crosses the black band in this case too, which is due to the noise on the floor sensor, the bigger proportion of *Env.A* encountered during evolution and the varying amplitude of the *way in* zone. Finally, the average offset Δ value is almost identical to the one obtained in the post-evaluation with the Minimal Simulator. This is almost a surprising result which proves that indeed the only discrepancy present between the two simulated behaviors lies in the initialisation of the robot and the effect of 3D dynamics and kinematics in its repositioning—which can be minimized by allowing the robot to live longer in the 3D case.

Table 3.3: Post-evaluation in SWARMBOTS3D. The percentage of success (*Succ.* %) and the percentage of errors (*E1*, and *E2* in *Env.A*, and *E3*, and *E4* in *Env.B*,) over 100 trials are shown for both *Env.A* and *Env.B*. Additionally, the average offset Δ and its standard deviation (degrees) are shown for the environment type *Env.B*. These values are displayed for various timesteps and initial robot orientations.

Post-Evaluation Results in SWARMBOTS3D									
orientation	*timestep*	*Env.A*			*Env.B*				
		Succ.	*E1*	*E2*	*Succ.*	*E3*	*E4*	*Offset* Δ	
		(%)	(%)	(%)	(%)	(%)	(%)	*Avg.*	*Std*
0	700	93	0	0	88	12	0	-22.59	34.90
0	800	100	0	0	95	3	0	-21.07	34.52
0	900	100	0	0	96	3	0	-20.95	34.56
0	1000	100	0	0	95	0	0	-20.95	34.56
0	1100	100	0	0	92	1	0	-20.95	34.56
0	1200	100	0	0	82	0	0	-20.95	34.56
0	1300	100	0	0	79	3	0	-20.95	34.56
0	1400	100	0	0	71	0	0	-20.95	34.56
π	700	62	0	0	0	100	0	-	-
π	800	65	0	0	59	41	0	-15.52	46.97
π	900	93	0	0	92	4	0	4.13	25.24
π	1000	96	0	0	98	1	0	5.72	18.19
π	1100	97	0	0	94	0	0	5.89	17.48
π	1200	99	0	0	94	0	0	5.89	17.48
π	1300	99	0	0	87	0	0	5.89	17.48
π	1400	99	0	0	86	1	0	5.89	17.48
π+[-120,120]	700	94	0	0	58	40	0	-7.43	38.94
π+[-120,120]	800	100	0	0	93	5	0	0.47	33.5
π+[-120,120]	900	100	0	0	98	0	0	1.93	31.87
π+[-120,120]	1000	100	0	0	95	0	0	1.93	31.87
π+[-120,120]	1100	100	0	0	93	0	0	1.93	31.87
π+[-120,120]	1200	100	0	0	87	0	0	1.93	31.87
π+[-120,120]	1300	100	0	0	80	1	0	1.93	31.87
π+[-120,120]	1400	100	0	0	72	2	0	1.85	31.72

Chapter 4

Evolving communicating agents that integrate information over time

In this chapter we present a second set of experiments we conducted, of a much more complicated nature. Two *s-bots* have to integrate over time their perceptual experience and communicate the result to the other robot, resulting in a cooperative behavior. The robot that has first finished its integration of its perceptual state must inform the other member of the group—in case of absence of a *way in* zone, by signalling. Therefore, the actions of each robot are now affected by the perceptual state of the other member of the group. Finally, the robots are required to adjust their strategy after a signalling behavior has been triggered by either robot and go away from this inaccessible light source. It is important to notice that the latter behavior is reactive, since the robots simply react to an enviromental signal.

Evolution must find a solution by evolving a single neural network that controlls two robots, which must successfully alternate between reactive and non-reactive behaviors. Furthermore, the controller should be able to display even contradictory behaviors, that is going towards and away from the light source. Finally, the robots should quit their current activity and pursue another, once the communication signal is transmitted. These points illustrate the difficulties and challenges of the task.

We present in detail the simulation, the evolutionary algorithm, the neural controller and the evaluation function used for this experiment and in the end we present and analyse the results obtained. Furthermore, we compare once again the post evaluation results obtained with the Minimal Simulation

approach with the ones obtained in SWARMBOTS3D.

4.1 The Task

This task is inspired by two distinct sources. The first one is the final scenario depicted in Figure 1.3 and described in [15]. A desired property of an explorer robot is to be able to communicate to the other members of the swarm the results of his search. More specifically, in case a trough is detected by a robot, this robot should inform the swarm about the nature of the trough, so if it can be traversed by a single *s-bot* or if there is a *way in* zone, in order to pass to the other side. Since we have to allocate our resources in the best possible way to achieve an optimal use of the number of *s-bots* available, we want to avoid having more than one robots reaching to the same conclusions about the same part of the explored enviroment. The second source of inspiration is biology. Animals that forage in a heterogeneous environment, where resources are distributed in patches, are required to make "complex decisions" such as the patch in which to forage, and at which moment in time it is better to leave and travel to another patch. To make such decisions, animals need to acquire relevant information from their environment. Afterwards, through stigmergic or direct communication they communicate their decision to the other members of the colony. A general problem common to biology and robotics concerns the definition of the mechanisms necessary to decide when it is better to pursue a particular action in a certain location and at which moment in time it is better to leave for pursuing a similar or a different activity in a similar or different location.

Once again, we are going to make use of the Minimal Simulation approach. This time though, we have to overcome certain obstacles. Since two robots are present in the enviroment, we are probably going to face collisions between them. A physics-based 3D simulator can take care of the collisions since the two robots are modelled like dynamical bodies. In our Minimal Simulation approach we are not using any forces, therefore we have to evolve robots that can perform obstacle avoidance, thus avoiding the other robot. In order to be able to do this, we have to use simulated proximity Infra-Red sensors, along with the other sensors used also in the Tuci *et al.* experiment, so that the robots can "sense" each other. The robots perceive the light through their ambient light sensors, positioned 45 degrees left and 45 degrees right with respect to their heading. Light levels alter depending on the robots distance from the light. The colour of the arena floor is once again white, except for a circular band, centered around the lamp, within which

the floor is in shades of grey. The circular band covers an area between 40 cm and 60 cm from the light; the floor is black at exactly 40 cm from the light; the grey level decreases linearly with the distance from the light. Again the robots perceive the colour of the floor through their floor sensor, positioned on their bellies, which outputs a value scaled between 0—when the robots are positioned over white floor—and 1—when they are over black floor. The robots are also equipped with a sound sensor, which is binary and is set to 1 if any robot is signalling (including the robot itself), and to 0 if no robot is signalling. This sensor will implement communication.

Concerning the robot's controller, this time we decided not to use more neurons than the sum of inputs and outputs of the network, as was done by Tuci *et al.*. The reason is that with the use of IR proximity sensors and the extra communication sound sensor, the network has become already quite big. Therefore, it will be harder for evolution to find the solution, given that the search space has singifically grown. So, our genotype is a vector comprising 144 real values (121 connections, 11 decay constants, 11 bias terms and a gain factor).

At the beginning of each trial, two simulated *s-bots* are positioned within a boundless arena, one of them at about 85 cm west of the light bulb, with a randomly determined orientation chosen between north-east and south-east (see Figure 2.1 left), and one at about 115 cm west of the light bulb, with the same random orientation.

Similar to Tuci *et al.*, there are two types of environment. In *Env.A* the band presents a discontinuity—a *way in* zone (see Figure 2.1, left). In *Env.B* the band completely surrounds the light (see Figure 2.1, right). The *way in* zone represents the path along which the robots are allowed to safely reach the light in *Env.A*. Successful robots should prove capable of performing phototaxis as well as looking for the *way in* zone, avoiding to cross the black edge of the band. Such robots should always reach the light in *Env.A*, avoiding crashing with each other. On the contrary, in *Env.B*, the robots should, besides avoiding to cross the black edge, signal the absence of the *way in* zone by emitting a tone. Furthermore, they have to react to the latter signal—no matter from which robot it came from— and move away from this light source which does not present a *way in* zone.

4.2 The Simulation

During the evolution, each genotype is coded into a robot controller, and is evaluated 15 times, 12 times in *Env.A* and 3 times in *Env.B*. We use

this proportions once again, as they led to more robust controllers in the replication of the Tuci *et al.* experiment. At the beginning of each trial, the neural network is reset—i.e., the activation value of each neuron is set to zero. Each trial differs from the others in the initialisation of the random number generator, which influences the robots' starting position and orientation, the position of the *way in* zone, and the noise added to motors and sensors. For each trial in *Env.A*, the position of the *way in* zone is varied to facilitate the evolution of robust navigational strategies. Its amplitude varies within the interval $[\frac{\pi}{6}, \frac{\pi}{2}]$. Once, again we choose a varying amplitude since it contributed to obtaining more robust controllers in our replication of Tuci *et al.*. Within a trial, the robot life-span is 280 seconds (1400 simulation cycles). A trial is terminated earlier if both robots cross the black edge of the band (see dashed arrows in Figure 2.1) or because both reach an Euclidean distance from the light higher than 140 cm. In each trial e, the robots are rewarded by the average of the values of the fitness function of the two robots. This way we ensure that in order to result in a successful behavior, both robots must perform well.

In case the robots are situated in *Env.A*, they are each rewarded by function f_{e1} which corresponds to the sum of the following two components:

1. R_{motion}—This component rewards movement toward the light bulb, and it is computed as:

$$R_{\text{motion}} = \frac{d_i - d_f}{d_i} \tag{4.1}$$

   where d_i and d_f represent respectively the initial and the final Euclidean distance between the robot and the light bulb. In *Env.A*, d_f is set to 0 if the robot is less than 15 cm away from the light bulb.

2. R_{signal}—This component rewards agents that do not signal. The component is computed as:

$$R_{\text{signal}} = \begin{cases} 1 & \text{if no signalling} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

In case the robots are situated in *Env.B*, they are each rewarded by function f_{e2} which corresponds to the sum of the following two components:

1. R_{motion}—This component rewards movement toward the light bulb, if none of the robots have signalled, and moving away from the light bulb if any of the two robots has signalled. It is computed as:

$$R_{\text{motion}} = \begin{cases} \frac{d_i - d_f}{d_i} & \text{if no robot has signalled} \\ \frac{d_f}{dmax} & \text{if any robot has signalled} \end{cases} \quad (4.3)$$

where d_i and d_f represent respectively the initial and the final Euclidean distance between the robot and the light bulb and d_{max} is the maximum distance the robot is allowed to go away from the light. This will ensure that the robots will go away from the light after one of them has emitted a sound communicating that there is no *way in* zone, and reach a distance from it equal to 140 cm for this simulation. D_f is set to 0 when the robot is on the circular band in shades of grey.

2. R_{signal}—This component rewards agents that signal. The component is computed as:

$$R_{\text{signal}} = \begin{cases} 1 & \text{if any robot signalled} \\ 0 & \text{otherwise} \end{cases} \quad (4.4)$$

Notice that the evaluation function used is identical to the one used in Section 3.2.2 (equation 3.2), when the robots are located in *Env.A*. On the other hand, if the robots are situated in *Env.B*, R_{motion} is different in the sense that if any of the two robots has signalled, it changes and instead of rewarding phototaxis, it rewards the opposite behavior. Also, R_{signal} is different in the sense that each robot does not only get rewarded by its own action of emitting a sound, but also by the other group member's action. The evaluation function described above is very simple, being the least explicit possible. Even though component R_{signal} for *Env.B* could be made more explicit punishing robots that both finish the loop and signal, ignoring the communication signal, we let evolution free to explore the search space, which for this experiment is very big.

In order to evolve robot-robot avoidance, we punish the robots whenever they approach each other below a critical distance. The more the robots fall into thar error during their lifetime, the more they are punished. The maximum number of virtual crashes we allow is set to 3 and above this value the robots die being punished very severely, rewarded very poorly by the evaluation function. Since they are initialised quite far from each other in order to ensure that one of them will be able to finish its integration earlier than the other and signal first—in *Env.B*—, the only time the robots really interact is when they are located in *Env.A* and very close to the light—having both found the *way in* zone. We let the robots interact in this area for a long interval in order to ensure that they evolve robot-robot avoidance.

4.3 Results

We ran 10 evolutions, each with a different random initial seed, for 5000 generations. In eight of them, the maximum fitness value of around 2 was achieved. Two runs did not find the solution, and evolution was stuck at a point where the robots fail to signal the absence of a way in zone. Figure 4.1 shows the fitness of the best individual and the mean population fitness plotted against the generation number (5000) and averaged over the 10 replications. We can notice that the average fitness of the best individuals of all replications is slightly over 1.9, which is normal if we take into account the fact that the two unsuccessful controllers had a fitness of 1.8. Of course, post-evaluation will reveal once again if some controllers were lucky in obtaining the maximum fitness value and if there are drawbacks in their behavior.

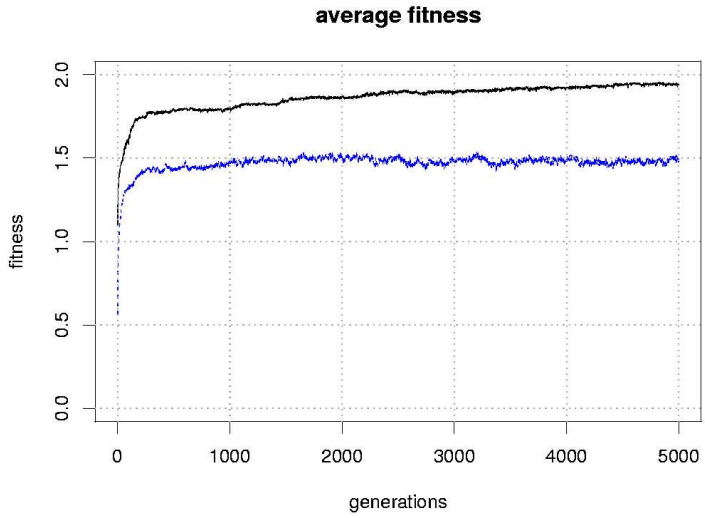

Figure 4.1: The Fitness during the evolution. The top thin line corresponds to the fitness of the best individual, while the dotted line refers to the average fitness of the population.

4.3.1 Post-Evaluation in the Minimal Simulator

In order to test the performance of the controllers evolved, it is required to perform a post evaluation. This is due to the possibility that the fitness of an

evolved individual is overestimated. If this is the case, the post evaluation will reveal it. Especially in our case, sometimes the robots might have been lucky without ever crashing against each other during their lifetime, or without ever crossing te black edge. Thus, we performed further analyses, by re-evaluating each of the best evolved final generation individuals for 100 trials in each type of environment (i.e., *Env.A* and *Env.B*). In each trial performed in *Env.A*, we look at the two robots' capability to both reach the light bulb (*Succ.*), without incurring in any error. Errors can be of two types, for each robot: *E1* refers to the emission of a sound signal and *E2* refers to crossing the black edge of the band. The results are displayed in Table 4.1. In *Env.B*, we look at the performance of the two robots in the following way: We measure the times the complete task is successful, that is the first robot to reach the band properly signals the absence of the *way in* zone and then both robots leave the grey zone and end up in a distance of 140 cm away from the light source, without committing any errors (*Succ.*). We also measure the reaction time for each robot (*react*), that is the number of timesteps required by each robot to get out of the circular band in shades of grey in order to go away from this light source. In this case, three error types are possible: *E3* refers to the lack of sound signalling by the first robot (we do not care if the second robot signals or not), *E4* refers to the robots crossing the black edge of the band and *E5* for each robot refers to the times they end up at a different distance from the light source than the desired value of 140 cm. Finally, in *Env.B* we also compute the offset between the entrance position of the robot arriving to the circular band first and the position in which this robot starts to signal. This measure, called offset Δ, is computed as in equation 3.4. The results are displayed in Table 4.2.

We can see from the results that it was not an easy task for evolution to find a solution for our task. Especially the fact that we require the robots to perform both phototaxis and go away from the light, and those contradictory behaviors to be both displayed by the same controller makes the task difficult. Evolution has to shape one single neural network for two robots that manage to display a very robust transition between reactive and non-reactive behaviors. Robots are also required to rapidly quit their current action and switch to another once they hear a communication signal emitted by the other member of the group. Nevertheless, evolution managed to find four controllers, namely the ones produced in runs no. 3, 7, 8 and 10 that perform very well. The errors are very few and the *Succ.* rate is high and could have been almost maximal if the robots did not crash sometimes when located in the vicinity of the light source in *Env.A*. Of course, these crashes are virtual and it would be better to use the expression of approaching each

other more than a critical distance. It is worth noting at this point that in almost all runs we have some crashes, but it is rather difficult to evolve robots that are required to be both in a distance smaller than 15 cm from the light source, given that their diameter is already 5.8 cm. Reaction time for the latter controllers is quite small and the average value of the offset Δ is excellent for runs no. 7 and 8, good for run no. 3 but big for run no. 10, since the *way in* zone's amplitude varies up to 90 degrees. Also, the robots almost always end up in the desired distance from the light, that is the number of *E5* is very low. Runs no. 2 and 6 perform fine, but the number of *E4* and *E5* is rather high. Run no. 4 is unsuccessful, but looking at the evolved behavior in our simulated 2D environment we found out that evolution found a solution that under some circumstances was evaluated with the maximum score, despite the fact that it was not optimal. The robots signal **before** reaching the black band, and then immediately go further away from the light source, reaching the distance they are supposed to, in all trials in *Env.B*. Also, in *Env.A*, we can see that in many trials the robots erroneously emit sound. Finally, the fact that it is hard for evolution to shape a network able to display two contradictory behaviors—going towards and away from the light—is illustrated in the results of run no. 5. The performance is optimal, except for the fact that the robots in all cases fail to reach the desired distance from the light (140 cm). On the contrary, they start spinning at a smaller distance to it and afterwards the going away behavior is disrupted.

Concerning communication, it is evident that in the vast majority of the cases the robots react very quickly to the signalling. Notice that we measure the simulated timesteps they spend to go out of the circular band in shades of grey. An average of 45 timesteps corresponds to 4.5 seconds to leave the black band, which is a very fast reaction. We remind the reader that the robots are required to disrupt their action of looping around the light staying inside the grey circular band and then signalling the absence of a *way in* zone. They must pursue a very different action from that point on, which is not based on constant light readings anymore (see Section 3.5.2 and Figure 3.6).

Finally, regarding the times the robots cross the black band, we notice that in our experiment, this number is higher than in the replication of Tuci *et al.*. One explanation for this is that the robots are initialised with an average distance of 30 cm between them. As we saw in Section 3.5.4, the distance to the lights for which the robots are evolved is crucial, and when placed in varying distances, their performance—in general—drops. In our case, one single network should control two robots that must perform the

same task, but under very different conditions. The distance the one robot must cover in order to reach the circular band in shades of grey is much bigger than the one the other has to cover.

Table 4.1: Post-evaluation in the Minimal Simulation environment. Performance of the ten best evolved controllers in *Env.A*. The percentage of success (*Succ.* %) and the percentage of errors *E1* and *E2* over 100 trials are shown for both robots. Robot 1 is initialised closer to the light source.

Post-Evaluation Results in the Minimal Simulation Environment					
Env.A					
run	*Succ.*	*E1*		*E2*	
		r1	r2	r1	r2
	(%)	(%)	(%)	(%)	(%)
n. 1	85	0	0	0	0
n. 2	92	0	0	0	0
n. 3	88	0	0	1	0
n. 4	62	31	31	3	4
n. 5	94	0	0	0	0
n. 6	98	0	0	1	1
n. 7	76	0	0	0	0
n. 8	94	0	0	1	0
n. 9	94	0	0	0	0
n. 10	92	0	0	0	0

4.3.2 Post-Evaluation in SWARMBOTS3D

As we did in Chapter 3 and with the replication of the Tuci *et al.* experiment, we are interested in re-evaluating the controllers evolved in the Minimal Simulation environment in SWARMBOTS3D. To do so, we will extract the same information as in Section 4.3.1. We re-evaluate 100 times again in *Env.A* and 100 times in *Env.B* the best controllers of run no. 10, one of the most successful runs over all. In Table 4.3 we can see the results of the post-evaluation in *Env.A*, and in Table 4.4 the results for *Env.B*.

As we can see, in *Env.A*, the controller controllers behaves in a very similar manner in both simulators (see also Table 4.1). The success rate is high in both cases and the number of errors is almost identical. Again, the cases where the robots do not succeed usually are caused by crashes around

Table 4.2: Post-evaluation in the Minimal Simulation environment. Performance of the ten best evolved controllers in *Env.B*. The percentage of success (*Succ.* %), the reaction time, the percentage of errors *E3*, *E4* and *E5* over 100 trials are shown for both robots. Additionally, we show the average offset Δ and its standard deviation (degrees) for the first robot that completes the loop. Robot 1 is initialised closer to the light source.

Post-Evaluation Results in the Minimal Simulation Environment												
Env.B												
run	*Succ.*	*react*				*E3*	*E4*		*E5*		*Offset* Δ	
		r1		r2			r1	r2	r1	r2	*Avg.*	*Std*
		Avg.	*Std*	*Avg.*	*Std*							
	(%)	(%)	(%)	(%)	(%)	(%)	(%)	(%)	(%)	(%)		
1	0	—				100	0	0	—			
2	50	96.9	51.7	104.13	50.02	0	21	30	2	2	-86.71	54.86
3	95	65.23	25.98	67.02	28.98	0	2	3	0	0	-71.30	89.29
4	0	—				0	7	8	5	4	—	
5	0	113.17	35.62	118.13	29.12	0	0	0	100	100	-28.45	67.95
6	67	60.61	48.26	61.68	41.81	6	4	16	9	3	0.59	119.68
7	96	45.04	83.97	38.30	13.14	1	0	0	2	1	-30.34	122.50
8	90	46.31	11.29	45.98	10.53	1	3	5	1	0	-27.36	90.08
9	0	—				100	0	1	—			
10	100	74.1	17.67	73.8	16.58	0	0	0	0	0	-113.74	98.64

the light source, after the *way in* zone is found by both robots.

In *Env.B* though, things are not that similar (see also Table 4.2). The success rate is again 100% no errors are made, the reaction time is almost identical, but we see a very big discrepancy in the value of the offset Δ. When re-evaluated in the minimal simulator environment, the robot initialised closer to the band was signalling on average too early. On the other hand, when re-evaluated in SWARMBOTS3D, it is signalling too late. For reasons discussed in Section 3.5.5, the initial orientation of the robots, being π+[-120,120], is disrupting phototaxis and causes a lot of problems in the physics-based 3D environment. Also, as we saw in Section 4.3.1, the fact that the robots are initialised with an average distance of 30 cm between them, affects their performance. More specifically it is more difficult for the robot to reach the band first, to signal at the right moment the absence of a *way in* zone, since one single network should control two robots that must perform the same task, but under very different conditions. If we take into account the disruptive effect of dynamics in the phase where the robots have to adjust their orientation and perform phototaxis, we can explain the very big discrepancy in the values of the offset Δ in the two simulated envi-

ronments. Finally, as mentioned in Section 3.5.5, the feeling of time is not independent of the previous experience, that is the robot's memory while on the band is affected by the time it has been moving to get there.

It is clear that this time it is much more challenging to acquire similar results when testing an evolved controller in the two simulated environments. In this complex task we have to use parameters that worsen the performance of the controller in the physics-based 3D environment, like the difference in the initial distances. It is possible that modifying some parameters—like we did in Section 3.5.5—might lead to more consistent results. But this study is out of the scope of this work and was not performed.

Table 4.3: Post-evaluation in SWARMBOTS3D. Performance of one evolved controller in *Env.A*. The percentage of success (*Succ.* %) and the percentage of errors *E1* and *E2* over 100 trials are shown for both robots. Robot 1 is initialised closer to the light source.

Post-Evaluation Results in SWARMBOTS3D					
Env.A					
run	*Succ.*	*E1*		*E2*	
		r1	r2	r1	r2
	(%)	(%)	(%)	(%)	(%)
n. 10	89	0	0	0	0

Table 4.4: Post-evaluation in SWARMBOTS3D. Performance of one evolved controller in *Env.B*. The percentage of success (*Succ.* %), the reaction time, the percentage of errors *E3*, *E4* and *E5* over 100 trials are shown for both robots. Additionally, we show the average offset Δ and its standard deviation (degrees) for the first robot that completes the loop. Robot 1 is initialised closer to the light source.

Post-Evaluation Results in SWARMBOTS3D												
Env.B												
run	*Succ.*	*react*				*E3*	*E4*		*E5*		*Offset* Δ	
		r1		r2			r1	r2	r1	r2	*Avg.*	*Std*
		Avg.	*Std*	*Avg.*	*Std*							
	(%)	(%)	(%)	(%)	(%)	(%)	(%)	(%)	(%)	(%)		
10	100	69.86	1.03	58.40	8.50	0	0	0	0	0	72.62	14.00

Chapter 5

Conclusions

In this work we addressed the problem of defining the control system for a group of autonomous robots that have to deal with a non-reactive task. We successfully designed neural controllers for robots that integrate over time their perceptual experiences in order to initiate alternative actions. We made use of techniques derived from Artificial Evolution, and we showed how they can produce simple but effective and robust solutions. One of the interesting points that came up during our experiments was the problem of choosing the nature of the simulator in which we evolved our controllers, having to choose between a 3D physics-based environment taking into account dynamics and forces, and a Minimal Simulator—a much faster method. The latter solution was chosen since our task did not require forces in order to be solved and since we proved that with the correct modifications, the controllers it produces can perform equally well in a realistic simulated enviroment.

5.1 Results

We ran two sets of experiments. The first one was a replication of the experiment of Tuci *et al.*, in order to port its results in the SWARM-BOTS project context. In this experiment, a robot must integrate over time its perceptual experience, thus "feeling" the flow of time. Changes in various methodological aspects were made in order to produce more robust solutions and better-performing controllers. Initially, we tried to evolve the controllers in a physics-based 3D enviroment, but since this method was not efficient enough as far as time and computational power are concerned, it was abandoned. We turned to the Minimal Simulation approach, which was able to rapidly produce robust solutions. These solutions were able to perform

equally well when tested in a realistic simulated environment, revealing the power of the Minimal Simulation approach.

The second experiment was an extension of the latter experiment, more suited for a collective robotics scenario. Two robots were able to integrate over time their perceptual experience and communicate the result to the other member of the group, resulting in an alternative action. The simulated robots managed to display a very robust transition between reactive and non-reactive behaviors. Once again, we used the Minimal Simulation approach.

One of the major achievements of this work is the fact that we managed to design **one** controller able to display a very complex behavior. The controller can trigger even contradictory behaviors, namely phototaxis and going away from the light source, depending on a decision it has to make. Robots can also rapidly quit their current action and switch to another once they hear a communication signal emitted by the other member of the group. Of course, there are other ways of designing a control system for robots having to perform the task we described, like hand-crafting parts of the solution. We cannot prove that our approach is better, but we proved that our approach was successful, even though it is time-consuming and not trivial to synthesize an integrated (not modularised) neural network through an evolutionary process. Our contribution to the literature is that we showed that evolution can produce robust solution for a complex task like the one we described.

This experiment finally is a very important first step towards the realization of a complex SWARM-BOTS project scenario. We have designed a decision-making mechanism that with the use of communication will be very useful to the *swarm-bot*, since the *s-bots*will often be required to make complex decisions based on their perception of the environment.

5.2 Future Work

Our work can be extended in various directions. The first and most obvious is a collective robotics one, like the task the *swarm-bot* has to carry out in the scenario described in [15]. It would be of particular interest to have robots that are able to aggregate or self-assemble after one of them realises that the trough it discovered cannot be traversed by one robot. Also, if the knowledge every robot gathers about its environment is communicated, we will have a better allocation of resources, exploiting the *s-bots* more efficiently. Another direction is more biology-inspired, specifically by patch

foraging. It would be interesting to see if robots that leave a light source that is not accessible can locate and approach other, accessible light sources, by exploring their environment. The inspiration is animals that leave a food source they cannot exploit and look for others in their environment. Finally, we could also move in the direction of exploring the use of communication. If the sound sensor is made distance or direction-dependent, we can be able to coordinate the movement of the two robots in various manners, enriching the repertoire of behaviors they can exhibit.

Nevertheless, as in any robotics study, the ultimate challenge is to be able to port the results obtained in simulation on real robots. In order to be more confident that we will be successful, we have to perform some further experimenting and apply several changes. Sensorial information must become more realistic, especially concerning the sensor that encodes the status of the enviromental cue used for discrimination between different enviroments, that is the floor sensor, and the sound sensor that implements communication between the robots. The floor sensor can be one of the 15 infra-red proximity sensors mounted on the real *s-bot*. The sound sensor can be one or a combination of the four sound sensor mounted on the *s-bot*. Furthermore, the simulated robot model used has to be closer to reality. Since the SWARMBOTS3D simulator provides a detailed simulated *s-bot* which models very closely the real robot, we will run our experiments with that model. Finally, methodological aspects during evolution might need reconsideration in order to achieve more robustness. Specifically, we would like to make the "feeling of time" independent of previous—irrelevant to the task—experience, but only dependent of the environmental cue used for discrimination.

Bibliography

[1] D.H. Ackley and M.L. Littman. Interaction Between Learning and Evolution. In *Proceedings of the Second Conference on Artificial Life*, pages 487–507, New York, USA, 1991. Addison-Wesley.

[2] 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, Piscataway, NJ, October 2001. IEEE Press.

[3] T. Balch and R.C. Arkin. Communication in Reactive Multiagent Robotic Systems. *Autonomous Robots*, 1(1):27–52, 1994.

[4] 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.

[5] R. D. Beer. A Dynamical Systems Perspective on Agent-Environment Interaction. *Artificial Intelligence*, 72:173–215, 1995.

[6] R. D. Beer. On The Dynamics of Small Continuous-Time Recurrent Neural Networks. *Adaptive Behavior*, 3(4):471–511, 1995.

[7] R.D. Beer and J.C. Gallagher. Evolving Dynamical Neural Networks for Adaptive Behavior. *Adaptive Behavior*, 1:91–122, 1992.

[8] J. Blynel and D. Floreano. Exploring the T-Maze: Evolving Learning-Like Robot Behaviors using CTRNNs. In G. R. Raidl et al., editor, *Applications of Evolutionary Computing, EvoWorkshops2003*, volume 2611 of *LNCS*, pages 598–609. Springer-Verlag, Berlin, Germany, 2003.

[9] A. Bonarini and V. Trianni. Learning Fuzzy Classifier Systems for Multi-Agent Coordination. *Information Science*, 136:215–239, 2001.

[10] R. Brooks. Intelligence Without Representation. *Artificial Intelligence*, 47:139–159, 1991.

[11] R. A. Brooks. Intelligence Without Reason. In J. Mynlopoulos and R. Reiter, editors, *Proceedings of the* 12^{th} *International Joint Conference on Artificial Intelligence*, San Mateo, 1991. CA: Morgan Kaufmann.

[12] D. J. Bruemmer, D. D. Dudenhoeffer, M. O. Anderson, and M. D. McKay. A Robotic Swarm for Spill Finding and Perimeter Formation. In *Proceedings of International Conference on Nuclear and Hazardous Waste Management (Spectrum)*, Reno, NV, USA, 2002.

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

[14] 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, Los Alamitos, CA, USA, May 1994. IEEE Computer Society Press.

[15] M. Dorigo. Wp5: Project Demonstration Scenario. Technical report, SWARM-BOTS PROJECT, 2004.

[16] M. Dorigo, V. Trianni, E. Şahin, R. Groß, T. H. Labella, G. Baldassarre, S. Nolfi, J.-L. Deneubourg, F. Mondada, D. Floreano, and L. M. Gambardella. Evolving Self-Organizing Behaviors for a Swarm-bot. *Autonomous Robots*, 2004.

[17] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.

[18] J. Elman. Finding Structure in Time. *Cognitive Science*, 14:179–211, 1990.

[19] 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.

[20] D. Floreano and F. Mondada. Evolution of Plastic Neurocontrollers For Situated Agents. In *From Animals to Animats 4, Proceedings of the International Conference on Simulation of Adaptive Behavior*. MIT Press, 1996.

[21] P. Gaudiano, B. Shargel, E. Bonabeau, and B. T. Clough. Swarm Intelligence: a New C2 Paradigm with an Application to Control of Swarms of UAVs. In *Proceedings of the* 8^{th} *International Command and Control Research and Technology Symposium*, 2003.

[22] B. Gerkey, R. T. Vaughan, and A.Howard. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proceedings of the* 11^{th} *International Conference on Advanced Robotics, (ICAR'03)*, pages 317–323, 2003.

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

[24] 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.

[25] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc., 1989.

[26] R. Groß and M. Dorigo. Cooperative Transport of Objects of Different Shapes and Sizes. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence – Proceedings of ANTS 2004 – Fourth International Workshop*, volume 3172 of *Lecture Notes in Computer Science*, pages 107–118. Springer Verlag, Berlin, Germany, 2004.

[27] R. Groß and M. Dorigo. Group Transport of an Object to a Target that Only Some Group Members May Sense. Technical Report TR/IRIDIA/2004-4, Université Libre de Bruxelles, Belgium, 2004. Submitted to the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), September 18-22, 2004, Birmingham, UK.

[28] I. Harvey, P. Husbands, and D. Cliff. Issues in Evolutionary Robotics. In J.-A. Meyer, H. Roitblat, and S. Wilson, editors, *Proc. of the* 2^{nd} *Int. Conf. on Simulation of Adaptive Behavior*, pages 364–373, Cambridge MA, 1992. MIT Press.

[29] 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, Cambridge, MA, 1994.

[30] Tuci E. Harvey I., Di Paolo E.A. and Wood R. Evolutionary Robotics: A New Scientific Tool for Studying Cognition. *Artificial Life*, 2004, forthcoming.

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

[32] N. Jakobi. Evolutionary Robotics and The Radical Envelope of Noise Hypothesis. *Adaptive Behavior*, 6:325–368, 1997.

[33] K.S. Espenschied J.C. Gallagher, R.D. Beer and R.D. Quinn. Application of Evolved Locomotion Controllers to a Hexapod Robot. *Robotics and Autonomous Systems*, 19:95–103, 1998.

[34] 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. press.

[35] M. Labella, T.H.. Dorigo and J.-L. Deneubourg. Self-Organised Task Allocation in a Group of Robots. In R. Alami, editor, *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems (DARS04)*, Toulouse, France, June 23–25 2004.

[36] T.H. Labella, M. Dorigo, and J.-L. Deneubourg. Efficiency and Task Allocation in Prey Retrieval. In A.J. Ijspeert, D. Mange, M . Murata, and S. Nishio, editors, *Proceedings of the First International Workshop on Biologically Inspired Approaches to Advanced Information Technology (Bio-ADIT2004)*, Lecture Notes in Computer Science, pages 32–47. Springer Verlag, Heidelberg, Germany, 2004.

[37] M.J. Matarić. Using Communication to Reduce Locality in Distributed Multiagent Learning. *Journal of Experimental and Theoretical Artificial Intelligence,*, 10(3):357–369, 1998.

[38] 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 Verlag, Heidelberg, Germany, 1999.

[39] 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, Oct. 28-30 1993.

[40] 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 2002. University of Zurich.

[41] 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, Los Alamitos, CA, USA, May 1994. IEEE Computer Society Press.

[42] 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.

[43] S. Nolfi. *EvoRob 1.1 User Manual*. Institute of Psychology, National Research Council (CNR), 2000. Available at http://gral.ip.rm.cnr.it/evorobot/simulator.html.

[44] S. Nolfi. Evolving Robots Able to Self-Localize in The Environment: The Importance of Viewing Cognition as The Result of Processes Occurring at Different Time Scales. *Connection Science*, 14(2):231–244, 2002.

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

[46] S. Nolfi and D. Marocco. Evolving Robots Able to Integrate Sensory-Motor Information over Time. *Theory in Biosciences*, 120:287–310, 2001.

[47] D. Parisi, F. Cecconi, and S. Nolfi. Econet: Neural Networks that Learn in an Environment. *Network*, 1:149–168, 1990.

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

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

[50] D. Payton, R. Estkowski, and M. Howard. Compound Behaviors in Pheromone Robotics. *Robotics and Autonomous Systems*, 44(3-4):229–240, 2003.

[51] P. Pirjanian, C. Leger, E. Mumm, B. Kennedy, M.Garrett, H. Aghazarian, S. Farritor, and P.Schenker. Distributed Control for a Modular, Reconfigurable Cliff Robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, 2002.

[52] 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.

[53] 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.

[54] Nolfi S. and Parisi D. Auto-Teaching: Networks that Develop their Own Teaching Input. In *Proceedings of the Second European Conference on Artificial Life*, Brussels, Université Libre de Bruxelles, 1993.

[55] 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, 2002. Piscataway, NJ: IEEE Press.

[56] P. S. Schenker, T. L. Huntsberger, P. Pirjanian, E. T. Baumgartner, and E. Tunstel. Planetary Rover Developments Supporting Mars Exploration, Sample Return and Future Human-Robotic Colonization. *Autonomous Robots*, 14(2-3):103–126, March - May 2003.

[57] 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, Barcelona, Catalonia, Spain, June 2000. ACM Press.

[58] 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, Marina del Rey, CA, Mar 25-27 2002. IOS Press.

[59] K. Sugawara and M. Sano. Cooperative Acceleration of Task Performance: Foraging Behavior of Interacting Multi-Robots System. *Physica D*, 100:343–354, 1997.

[60] P. M. Todd and G. F. Miller. Exploring Adaptive Agency II: Simulating the Evolution of Associative Learning. In J.-A. Meyer and S. A. Wilson, editors, *From Animals to Animats I: Proceedings of the* 1^{st} *International Conference on Simulation of Adaptive Behavior*, pages 306–315. MIT Press, Cambridge, 1991.

[61] P. M. Todd and G. F. Miller. Exploring Adaptive Agency III: Simulating The Evolution of Habituation and Sensitization. In H. P. Schwefel and R. Männer, editors, *Proceedings of the* 1^{st} *Workshop on Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*. Springer, Berlin, 1991.

[62] V. Trianni, E. Tuci, and M. Dorigo. Evolving Functional Self-Assembling in a Swarm of Autonomous Robots. In *From Animals to Animats 8. Proceedings of the Eight International Conference on Simulation of Adaptive Behavior (SAB04)*. MIT Press, Cambridge, 2004. to appear.

[63] E. Tuci, I. Harvey, and P. M. Todd. Using a Net to Catch a Mate: Evolving CTRNNs for the Dowry Problem. In B. Hallam, D. Floreano, J. Hallam, G. Hayes, and J.-A. Meyer, editors, *From Animals To Animats VII: Proceedings of the* 7^{th} *International Conference on the Simulation of Adaptive Behavior (SAB'02)*. MIT press, Cambridge, MA, 2002.

[64] E. Tuci, M. Quinn, and I. Harvey. An Evolutionary Ecological Approach to the Study of Learning Behaviour Using Robot-Based Model. *Adaptive Behavior*, 10(3-4):201–221, 2002.

[65] E. Tuci, V. Trianni, and M. Dorigo. Evolving The "Feeling" of Time through Sensory-Motor Coordination: a Robot-Based Model. Technical Report TR/IRIDIA/2004-7, Universitè Libre de Bruxelles, 2004. To appear in PPSN'04.

[66] R. T. Vaughan, B. Gerkey, and A. Howard. On Device Abstractions For Portable, Resuable Robot Code. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems, (IROS2003)*, 2003.

[67] M. Wheeler. From Robots to Rothko: The Bringing Forth of World. In M. A. Boden, editor, *The Philosophy of Artificial Life*, chapter 7, pages 209–236. Oxford University Press, Oxford, 1996.

[68] B. M. Yamauchi and R. D. Beer. Sequential Behavior and Learning in Evolved Dynamical Neural Networks. *Adaptive Behavior*, 2(3):219–246, 1994.

[69] 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.

[70] T. Ziemke and M. Thieme. Neuromodulation of Reactive Sensorimotor Mappings as a Short-Term Memory Mechanism in Delayed Response Tasks. *Adaptive Behavior*, 10(3-4):185–199, 2002.