Emergence of internal representations in evolutionary robotics : influence of multiple selective pressures

Charles Ollion

▶ To cite this version:

Charles Ollion. Emergence of internal representations in evolutionary robotics : influence of multiple selective pressures. Other. Université René Descartes - Paris V, 2013. English. NNT : 2013PA05S023 . tel-00948029

HAL Id: tel-00948029
https://theses.hal.science/tel-00948029
Submitted on 17 Feb 2014

PhD Thesis from Paris-Descartes University

Major : **Computer Science (EDITE)**

M. Charles Ollion

Emergence of Internal Representations in Evolutionary Robotics: influence of multiple selective pressures

Émergence de représentations internes en robotique évolutioniste en présence de pressions de sélection multiples

October 7, 2013

Jury:

Dr. Tom Ziemke	University of Skövde	Reviewer
Dr. Hervé Luga	University Toulouse 1 IRIT	Reviewer
Dr. Patricia A. Vargas	Heriot-Watt University	Examiner
Dr. Amine Boumaza	University of Loraine, LORIA	Examiner
Dr. Benoît Girard	CNRS, ISIR, Univ. Pierre et Marie Curie	Examiner
Dr. Stéphane Doncieux	ISIR, Univ. Pierre et Marie Curie	PhD supervisor

Contents

1 Introduction **1**

1.1 Context . . . 1

1.1.1 Artificial Intelligence and Cognitive Science . . . 1

1.1.2 Evolutionary Robotics . . . 3

1.2 Problem . . . 4

1.3 Outline . . . 6

2 Background **7**

2.1 Evolutionary Algorithms . . . 7

2.1.1 History . . . 7

2.1.2 Principle . . . 8

2.1.3 Multi-Objective . . . 11

2.2 Evolutionary Robotics . . . 21

2.2.1 Introduction . . . 21

2.2.2 Behavior . . . 22

2.2.3 Fitness Functions . . . 22

2.2.4 Conclusions . . . 29

2.3 Artificial Neural Networks . . . 29

2.3.1 Neuron models . . . 29

2.3.2 Neural Networks . . . 31

2.3.3 Evolution and Artificial neural networks . . . 33

2.3.4 Evolution of Artificial Neural Network Topology . . . 34

2.3.5 Conclusions . . . 42

3 Internal Representations **45**

3.1 Internal Representations and Cognitive Behaviors . . . 46

3.1.1 Internal Representations . . . 46

3.1.2 Examples of Reactive and Cognitive Behaviors . . . 47

3.1.3 Internal Representations in Computational Neuroscience . . . 49

3.2 Internal Representations in Evolutionary Robotics . . . 51

3.2.1 Representation-less ER tasks . . . 52

3.2.2 Representation-Hungry Tasks . . . 52

3.2.3 Tasks with unclear Representation needs . . . 54

3.2.4 Conclusion . . . 56

3.3 Testing Characteristics of Internal Representations . . . 56

3.3.1 Measuring Representations in the literature . . . 56

3.3.2 Properties of Internal Representations . . . 57

3.3.3 Quantitative tests . . . 58

3.3.4 Qualitative tests . . . 61

3.4 Analysis over Existing Networks . . . 62

3.4.1 Representations in classic networks . . . 63
3.4.2 Visual Attention . . . 65
3.4.3 ER task: Hard Ball-Collecting Task . . . 67
3.4.4 Conclusion . . . 71

4 Behavioral Consistency Method 73
4.1 Introduction . . . 74
4.2 Working Memory Experiment . . . 74
4.2.1 T-Maze navigation task . . . 75
4.2.2 Methods . . . 76
4.2.3 Results . . . 78
4.2.4 Conclusion . . . 82
4.3 Behavioral Consistency Method . . . 85
4.3.1 Related Work . . . 87
4.3.2 Behavioral Consistency Method . . . 87
4.4 Attention Focus . . . 90
4.4.1 Experimental Setup . . . 90
4.4.2 Results . . . 93
4.5 Action Selection . . . 94
4.5.1 Experimental setup . . . 94
4.5.2 Results . . . 96
4.6 Conclusion . . . 97

5 Environmental Pressures and emergence of Internal Representations 99
5.1 Selective Pressures for the emergence of Internal Representations . . . 100
5.1.1 Task and World Complexity . . . 100
5.1.2 Helper Objectives and Specific Pressures . . . 101
5.2 Circular Maze Evolutionary Robotics Protocol . . . 102
5.2.1 Goal-oriented and Novelty Objective . . . 105
5.2.2 Behavioral Consistency Objective and Environmental Pressures 108
5.2.3 Representation Tests . . . 111
5.2.4 Setups and Selective Pressures . . . 113
5.3 Results: Emergence of Internal Representations under Environmental Pressures . . . 114
5.3.1 Impact of Selection Pressures . . . 114
5.3.2 Resulting Architectures . . . 121
5.3.3 Lineage and selection for Internal Representation . . . 126
5.4 Conclusion . . . 129

6 Discussion and Perspectives 131
6.1 Contributions . . . 131
6.1.1 Definition of Internal Representations . . . 131
6.1.2 Behavioral Consistency Method . . . 132

6.1.3 Emergence of Internal Representations under multiple selective pressures . . . 132
6.2 Discussion and Perspectives . . . 133
6.2.1 Classification of representations and tests . . . 134
6.2.2 Behavioral Consistency Method versus standard evaluation . 134
6.2.3 Emergence of Internal Models . . . 136
6.2.4 New Selective Pressures . . . 137
6.2.5 Neural Network Encodings . . . 138

7 Conclusion 141

A Appendix 143
A.1 DNN and EvoNeuro2 Parameters . . . 143
A.2 Additional Lineage . . . 144
A.3 Neural Networks . . . 145
A.4 Chronograms . . . 146
A.5 Comparison between BCO and Standard Evaluation . . . 147

Bibliography 149

Chapter 1

Introduction

Contents

1.1 Context 1
1.1.1 Artificial Intelligence and Cognitive Science 1
1.1.2 Evolutionary Robotics 3
1.2 Problem 4
1.3 Outline 6

1.1 Context

1.1.1 Artificial Intelligence and Cognitive Science

Nature is filled with examples of animals displaying complex behaviors, ranging from bacteria to mammals. *Cognitive science* is the interdisciplinary field aiming at studying the control system of animals (for instance, nervous systems) and their processes, with a particular emphasis on the human brain. Cognitive science includes fields such as psychology, neuroscience, and in particular artificial intelligence.

The term *Artificial Intelligence* (AI) is originally referred as “the science and engineering of intelligent agents” (McCarthy et al., 1955). While Artificial Intelligence is still being the long term goal, the problem was broken down into simpler sub-problems: building systems with the ability to navigate autonomously, to learn, to make decisions, etc. Early AI works were based on the belief that human intelligence could be reduced to symbolic manipulation. Despite the enthusiasm of its pioneers[1], the progress of artificial intelligence research slowed down in the 1970s; and many researchers believed that these sub-problems were beyond reach.

The symbolic approach to AI is based on mathematics and logic. In contrast, some approaches focus on mimicking the control systems and processes found in animals. Specifically, the “animat” approach considers the development of autonomous artificial agents simulated or real, by drawing inspiration from the behavior of real animals (Wilson, 1991).

Cognition in the animal kingdom (Shettleworth, 2009) can be studied at various levels of abstraction: from the precise study of the nervous systems, to the study

[1]For instance, in 1965, H. A. Simon stated: “machines will be capable, within twenty years, of doing any work a man can do.”

of the behavior of the animals. The control systems found in the animal kingdom include very diverse nervous system structures (Shettleworth, 2009). Several cognitive science sub-fields, such as Computational Neuroscience, aim at finding the underlying *neural models* explaining the behaviors of the animals.

Early theories in cognitive science have related the behavior of cognitive agents and internal models (Craik, 1943; Johnson-Laird, 1983). Those models are described as mental maps that incorporate symbolic representations. For instance, Tolman (1948) describes cognitive maps, that take part in animal navigation and path planning. These maps, called **internal representations** (Tolman, 1948; Chomsky, 1965; Marr, 1982), are the basis of models and cognitive functions. These internal structures are the central notion of traditional theories in cognitive science (see e.g. Johnson-Laird (1980) for a review). In the case of Tolman's maps, or Craik's models, the representations were *symbolic*, and their activity was expected to be easily interpreted.

However, in the late 1980s, Harnad (1990) introduced the symbol grounding problem, which questioned the presence of symbolic representations and their links with the world. In the 1990s, cognitive science saw a new emphasis on the embodied and situated intelligence, including works by Brooks (1991b); Varela et al. (1991), rejecting the symbolic-only views. It was argued that "the world is its own best model" (Brooks, 1991b), and emphasized intelligence without representation. This led to the development of agents which could solve tasks and have seemingly complex behaviors with reactive controllers. A good illustration of this concept lies in the work of Braitenberg (1986): wheeled robots, with a very simple reflex control architecture, seem "intelligent" or to have emotions, such as fear. The complexity does not lie in their internal structure, but rather in their environment and their interactions with it.

This anti-representationalist view has created debates in the cognitive science and artificial intelligence communities. However, Clark and Toribio (1994) argue that there might not be a representational/non-representational dichotomy, but rather different degrees and types of possible representations, from no representations at all to sophisticated models. In embodied views of cognition, cognition can rely on prediction and internal states, however the representations[2] are based on sensorimotor codes rather than symbols (Wolpert et al., 1995; Keijzer, 2002; Pfeifer and Bongard, 2006).

In embodied (and *situated*) views of cognition, the interaction between the agents and the external world is emphasized. The artificial agents, for instance robots, are connected to their environment through their bodies, motors and sensors. The behavior of the agents depends on all of these parameters — not only on the controller of the robot (Pfeifer and Bongard, 2006). From a human point of view, it may be challenging to understand the dynamics between the sensors, motors and environment of the agent (Brooks, 1991b,a). Brooks hypothesizes that

[2]The term representation can then be misleading and has been criticized (Keijzer, 2002). The exact scope of the notion of representation in this thesis will be defined in Chapter 3.

modeling what a robot perceives and which information it can exploit efficiently is very challenging. Therefore, the human design of a robotic controller in a complex environment is very challenging, even though some simple sensorimotor interaction loops may exist.

Unfortunately, there is no consensus on the systematic design principles within this view of cognition. Consequently researchers rely on adaptive approaches in order to build artificial agents: optimization techniques or learning algorithms may find solutions that humans can hardly find. In particular, inspiration may be drawn from the natural adaptation and emergence of the nervous systems through natural evolution. *Evolutionary Robotics* (ER) draws such inspiration to automatically design and optimize embodied agents. It has received a lot of attention in the situated and embodied views of cognition (Brooks, 1991b; Pfeifer and Bongard, 2006), and will be the basis of this thesis.

1.1.2 Evolutionary Robotics

Evolutionary Robotics (ER) is an embodied and situated Artificial Intelligence approach which appeared in the early 1990s with the publication of two promising works by Floreano (1994) and Harvey and Husbands (1992). The aim was to automatically design controllers of autonomous agents using artificial evolution. The principle of ER relies on the optimization of the control architecture of real or simulated agents with an evolutionary algorithm (Meyer et al., 1998; Floreano and Nolfi, 2000). Therefore the researchers do not program directly the robot: ER is a bottom-up design method.

Evolutionary algorithms (EAs) (Goldberg, 1987) are stochastic optimization algorithms loosely inspired by the Darwinian theory of evolution (see e.g. Eiben and Smith (2003)). They rely on the evaluation of a population of candidate solutions (see section 2.1 for more details). In order to evaluate the solutions, a *fitness function* is defined by the researchers, from which a ranking of the solutions is possible. The fitness function is one of the most important aspect of evolutionary algorithms as it drives the evolutionary process towards the desired solutions.

When optimizing artificial agents, the researchers often have more than one conflicting criteria. For instance an agent should be as efficient as possible in solving a task, but consume as little energy as possible. In multi-objective evolutionary algorithms (MOEAs) (Fonseca and Fleming, 1993; Srinivas and Deb, 1994; Deb et al., 2000), the fitness function consists of several “objectives” all of which influence the selection process. The optimization process may then result in a collection of solutions which are compromises between the different objectives.

In natural evolution, any cause that constrains the reproduction in a population exerts a *selective pressure* (or selection pressure). Selective pressures may include climate change, predation, diseases, or even noise in the animal sensors, etc. The individuals of the population that cope best with these pressures are more likely to survive and reproduce, and are therefore naturally selected. The Malaria parasite provides an example of selective pressure. The parasites has led to natural selection for erythrocytes carrying the sickle cell hemoglobin gene mutation because it grants some resistance to Malaria. As a result, Malaria exerts a selective pressure towards the sickle cell hemoglobin gene mutation even though it causes sickle cell anaemia [3].

By analogy with natural evolutionary processes, *selective pressures* can be added to artificial evolution experiments. A selective pressure may be applied by adding an objective which drives the evolutionary process alongside other objectives.

Besides fitness functions, a crucial aspect of Evolutionary Robotics lies in the nature of the evolved controllers. Different types of controllers can be optimized through evolution: artificial neural networks (Floreano and Mondada, 1996), genetic programming (Koza, 1992), fuzzy logic (Hoffmann and Pfister, 1996), or parametrized models. Neural networks are one of the most common architecture for controllers in ER (Nelson et al., 2009), and will be focused on in this thesis. These controllers have been tested on real or simulated robots, on various tasks: obstacle avoidance (Floreano and Mondada, 1996), walking gaits (Hornby et al., 2000), foraging (Mouret and Doncieux, 2012a) and swarm robotics (Großet al., 2006).

To summarize, this work deals with the evolution of artificial neural networks controlling robots (simulated in this case). The focus is on the study of selective pressures driving the evolution, rather than the encoding of the neural network.

1.2 Problem

Biologists have tried to characterize the level of cognition in the animal kingdom (Shettleworth, 2009). For instance, behaviors purely based on reflexes are called *reactive behaviors*; behaviors involving complex computations and integration of information over time are *cognitive behaviors* (Shettleworth, 2009). Navigation provides a very practical framework to study cognition, due to the large number of studies on animals such as insects and rodents. In natural organisms, navigation can be achieved through reactive behaviors (taxis, depending only on local clues), or through cognitive path planning and other strategies (Trullier et al., 1997). In particular, a cognitive ability such as path planning is hypothesized to rely on an internal representation of the world, a cognitive map (Tolman, 1948).

While early navigation works in ER showed promising results (Harvey and Husbands, 1992; Floreano, 1994), most of the controllers display reactive behaviors (Nolfi, 2002; Ziemke et al., 2004; Nelson et al., 2009). In most ER experiments, representations do not spontaneously appear, most likely because they are not needed to solve the tasks. A task in which a representation is required for the agent to solve it is referred as *representation-hungry* (Clark and Toribio, 1994). Some ER experiments include *representation-hungry* tasks to study the emergence of simple internal representations, such as memory units (Ziemke and Thieme, 2002; Capi and Doya, 2005). They show a successful resolution of the task, but also that the emergence of such a representation is not straightforward.

This thesis relies on the intuition that more complex tasks may become

[3]`http://sickle.bwh.harvard.edu/malaria_sickle.html`, Kenneth R. Bridges, 2002

representation-hungry, and reaching higher levels of cognition will require some form of internal representation.

Consequently, the aim is to answer the following question:

$$\boxed{\text{Under which conditions do internal representations emerge in artificial agents ?}}$$

This problem can be addressed in several ways. One approach is to work on the network structure (Elman, 1990) or encoding (Stanley and Miikkulainen, 2002a), in order to encourage the emergence of representations. Another way is to study the selection process which drives the evolutionary search — and thus look for the selective pressures that facilitate the emergence of such representations. Both approaches are complementary, and many works have studied the influence of the structure or encoding (Gruau, 1995; Stanley and Miikkulainen, 2002a; Stanley et al., 2009). However, evolutionary pressures will be focused on in this work. The reason behind this choice is twofold: 1) selective pressures are independent from any particular encoding or network structure; 2) it was argued that selective pressures were more critical when solving difficult and deceptive problems (Mouret and Doncieux, 2012a).

This work deals mostly with a specific selective pressure, which is induced by the perception of the environment. Autonomous agents rely on their perceptions of the world to decide on their actions. In biological, as well as artificial control, the variability of the world induces perception issues in a real world environment:

- animals as well as robots usually have very noisy raw inputs;
- perceptions can be blocked (i.e. an obstacle can block the agent's vision) or hampered (i.e. day/night cycle for vision).

These two perception issues are referred to as *perception noise* and *perception occlusions*. These issues can be seen as selective pressures: in order to perform well in its environment an agent has to cope with these perception issues. Perception noise and occlusions are denominated as *environmental pressures* due to their natural presence in any real-world environment. In order to have a consistent behavior in this variable environment, some have argued that having an internal representation robust to these variabilities could be a major factor (Gigliotta et al., 2011).

Hypothesis The main hypothesis in this work is that the environmental pressures play a critical role in the emergence of internal representations for artificial agents. Additionally, other selective pressures, such as diversity or novelty (Mouret and Doncieux, 2012a; Lehman, 2008), are hypothesized to have a strong influence on the evolutionary runs.

To address this hypothesis, several tools and protocols are defined:

- a protocol to test neural networks for internal representations;

- a new method for designing efficient selective pressures.

The definitions of these protocols and tools are contributions on their own which can be used in other evolutionary robotic studies.

1.3 Outline

Chapter 2 details the necessary background about evolutionary algorithms and robotics. It emphasizes the use of multi-objective evolutionary algorithms, behavioral diversity, and specific neural network encodings.

Chapter 3 gives a definition of the internal representations which will be considered. It also introduces a protocol to test whether a neural network includes one of these representations or not.

Chapter 4 is separated into two parts: the first one uses the previous protocol to design a *memory helper objective* in an evolutionary robotics experiment. The second part defines a new method for designing objectives: the "Behavioral Consistency Method". This method will be used to add selective pressures to evolutionary experiments.

Finally chapter 5 investigates the main hypothesis. Using the tools and definitions introduced in the previous chapters, a new experiment is defined: the Circular Maze Evolutionary Robotics Protocol. This experiment is used to study the conditions of the emergence of internal representations in the robotic controllers.

These chapters are followed by a discussion in Chapter 6, in which the work is reviewed and ideas for further works are detailed; and a brief conclusion in Chapter 7.

Chapter 2

Background

Contents

2.1 Evolutionary Algorithms . . . 7
2.1.1 History . . . 7
2.1.2 Principle . . . 8
2.1.3 Multi-Objective . . . 11
2.2 Evolutionary Robotics . . . 21
2.2.1 Introduction . . . 21
2.2.2 Behavior . . . 22
2.2.3 Fitness Functions . . . 22
2.2.4 Conclusions . . . 29
2.3 Artificial Neural Networks . . . 29
2.3.1 Neuron models . . . 29
2.3.2 Neural Networks . . . 31
2.3.3 Evolution and Artificial neural networks . . . 33
2.3.4 Evolution of Artificial Neural Network Topology . . . 34
2.3.5 Conclusions . . . 42

2.1 Evolutionary Algorithms

2.1.1 History

The Origin of Species lays the foundations of the Theory of Evolution (Darwin, 1859). Almost a century later, the synthetic theory of evolution (Mayr, 1942) integrates Darwin's theories with Mendelian inheritance (Mendel, 1865) and population genetics (Haldane, 1932). This theory sets the mechanisms of differentiation of individuals among a population, as well as the transmission of characteristics of the parents to their offspring. Many of these characteristics are transmitted through the genome of the individuals, which is subject to random mutation, and recombination through crossover — in case of sexual reproduction.

Evolutionary Algorithms are an abstraction of this principle which can be seen as an optimization process. The use of the theory of evolution as an inspiration for computer programs has several independent origins. At the end of the 1950s, some of the first evolutionary mechanism are simulated by computer programs (Fraser, 1957). The use of Darwinian principles for problem solving was later introduced under three different concepts:

- Evolutionary programming (Fogel, 1962, 1991)
- Evolution strategies (Rechenberg, 1965, 1973; Schwefel, 1981)
- Genetic algorithms (Holland, 1975; Mitchell, 1998)

Evolutionary Algorithms (Bäck, 1996; Fogel, 2000; De Jong, 2006) regroup these three classes, as well as other later introduced methods such as genetic programming (Cramer, 1985; Koza, 1992), or estimation of distribution algorithms (Larranaga and Lozano, 2002). They became popular through their use as optimization tools (De Jong, 1975), and by the works of Goldberg (Goldberg, 1989). All of them present many common attributes: they make use of a population of solutions; solutions are modified through a random process; a selection process determines which solutions are kept or discarded.

2.1.2 Principle

An Evolutionary Algorithm (EA) is a population-based metaheuristic optimization algorithm. It relies on the few following key concepts:

Definition 1 (Genotype) *The genotype of an individual is the information transmitted to offspring, on which mutation and crossover operators apply.*

Definition 2 (Phenotype) *The phenotype of an individual is the observable characteristics of an individual.*

The expansion of the genotype to the phenotype is called the *development* phase. The mapping between the genotype and phenotype is called the *encoding*.

Definition 3 (Fitness) *The fitness function or cost function is an evaluation of the performance of the individuals.*

An EA can be summarized in a few steps (See figure 2.1 for illustration). First a population is initialized with random genotypes. This population is the current population of the first generation. Then, for each generation:

1. Compute the fitness of each individual in the population and sort the population on this basis. The fitness depends on the phenotype of the individuals.
2. Apply a selection process to select the parents of the next generation. The offsprings are generated from the parents, through copy, mutation and crossovers.
3. The new population is drawn from the offspring population and, eventually, parent population.

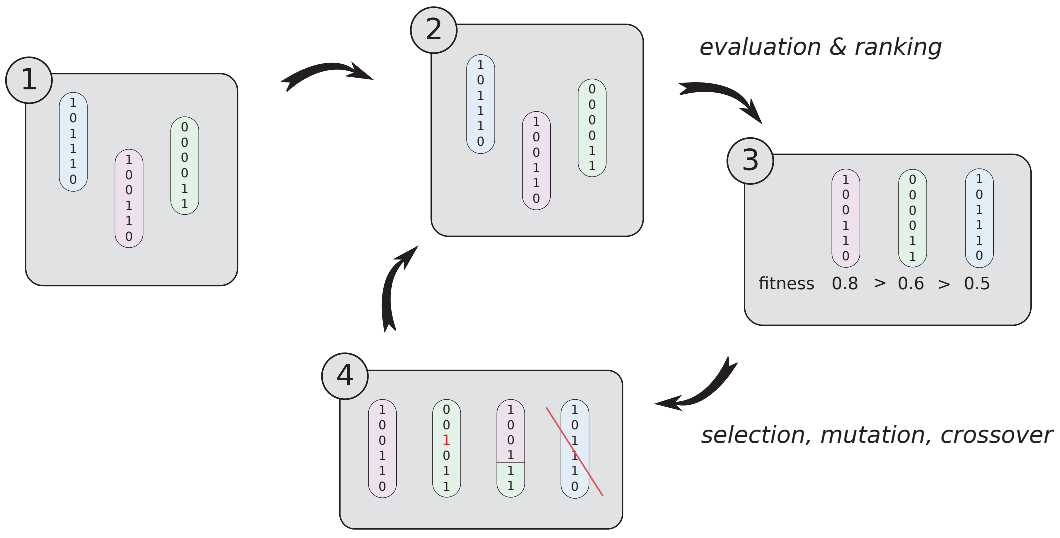
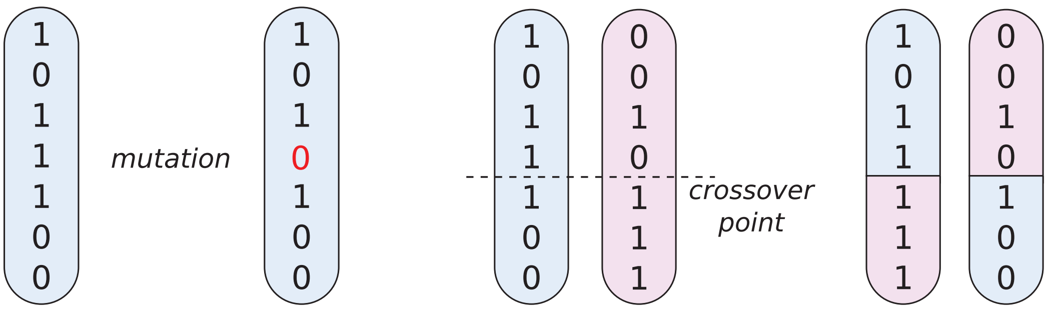


Figure 2.1: General principle of an Evolutionary Algorithm in 4 steps: 1) A population is initialized at random. 2) The current population, modified generation after generation. 3) Each individual of the current population is evaluated and ranked. 4) A selection process is performed based on the fitness of the individuals; the new population is generated by mutation and crossover of the selected individuals.

As a consequence of random mutations and crossovers, the offsprings of solutions can have better as well as worse fitness. However, the worst solutions found are likely to disappear while the better ones are likely to be selected as new parents. This usually results in a global increase of the fitness. This process continues until a stopping criterion is reached — for instance the fitness reaches a threshold or after a given number of generations. The result of the evolutionary process is usually the most fit individual of the last generation.

Genotype The genotype of an individual can be represented as a binary string $(0, 1^N)$, a real-valued vector $(G = [0,1]^N, \mathbb{R}^N)$, a graph, etc. Each sub-part of the genome is a *gene*. Variability operators applied in this genotypic space are inspired by biology: mutations (Bäck and Schwefel, 1993; Eiben and Schippers, 1998) and crossovers (Eiben et al., 1998; Jansen and Wegener, 1999).

Mutations A mutation modifies the genotype of a solution. For a binary string genotype, a common mutation operator is a bit inversion (figure 2.2, left). Each bit has a probability to be inverted, so that the offspring has one or more inverted bits. For real-valued genotypes, common mutation operator include gaussian variation: the operator adds a Gaussian distributed random value to the chosen gene. A graphical representation can have several mutation operators such as addition/removal of nodes or connections.



(a) Mutation (b) Crossover

Figure 2.2: (a) Mutation operator of a binary string genotype: the offspring has a random bit inverted. (b) One point crossover operator of a binary string genotype: a random pivot point is chosen, and two children are generated.

Crossover Another way to add variability in the offspring is to use crossovers. A crossover can be applied to generate offspring from two or more parents. The resulting offspring receives a combination their parents' genome. For binary string genotypes, the most straightforward crossover is the one-point crossover displayed in figure 2.2, (b). However the definition of the crossover mainly depends on the chosen encoding: the definition of a general crossover operator in a graph space is not straightforward (Eiben and Schippers, 1998; Stanley and Miikkulainen, 2002b).

Some evolutionary work only include one of these two variability operators, and there has been some debates on which one is the most important (Fogel, 2006). Some works only use crossovers, for instance in genetic programming Banzhaf et al. (2000), while others rely solely on mutations (Bäck et al., 1991; Mouret and Doncieux, 2012b). There is no clear consensus on whether one of these operator is crucial, as this question strongly depends on the genotype representation, size of the search space, as well as population size. However one is necessary at least, because otherwise no variability would occur in the population.

Phenotype The development phase expands the genotype into the phenotype. For instance a graph can be expanded into a neural network. Other common phenotypes include models or functions which are parametrized by the real values of the genotype.

The phenotype is evaluated in order to get the fitness of the individual. This evaluation range from simple application of a mathematical function, to a full simulation in a 3-dimensional environment of the phenotype of the individual.

Selection The selection of solutions among the population is one of the key points of evolutionary algorithms. While variation operators provide exploration of the search space, the selection process allows for convergence towards the best solutions. In that way, there exists a well-known trade-off between exploration and exploitation. An algorithm not exploiting enough will not converge, while an algorithm not exploring enough will have premature convergence into some local optima.

Several selection methods have been proposed:

- tournament selection (Brindle, 1981; Miller and Goldberg, 1995) in which a predefined number of tournaments are organized between randomly chosen groups of $n \geq 2$ solutions, and the most performing solution is chosen. The larger n, the higher the selective pressure. If $n = 2$, two poor solutions might be chosen for a tournament, one of them getting selected by the algorithm; if n is large, this is less likely to happen;
- fitness proportionate selection (or roulette-wheel selection) (Baker, 1987): Each solution has a chance to be selected proportional to its performance;
- ranking selection: the population is sorted according to fitness. The chance of selecting a solution is proportional to its rank.

Elitism and population replacement Keeping some unchanged parents in the new population is called elitism. This enables faster convergence as very good solutions are then less likely to disappear or to suffer from negative mutations.

In evolution strategies (ES), two parameters define how much elitism is applied, and how many replacement there will be. In a (μ, λ)-ES, the best μ solutions among the λ offspring are used to generate the next generation (no elitism). In a $(\mu+\lambda)$-ES, the μ best solutions are kept into the population, as well as some of the λ offspring.

Choice of Evolutionary Algorithms There are many evolutionary algorithms, some of which are fine tuned to solve certain types of problems. There is no consensus on a “best” evolutionary algorithm, different problems may require different algorithms. However, in black-box continuous optimization, one of the most popular methods is Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 2001; Hansen and Koumoutsakos, 2003). It relies on modifying the variation process in order to adapt to the specificities of the problem.

2.1.3 Multi-Objective

Several optimization problems cannot be reduced to a single objective optimization. In many cases, concurrent objectives are optimized (for instance, energy cost and speed of a walking robot), and many trade-off solutions exist.

Pareto optima (Pareto, 1897) are trade-off solutions in which any objective cannot be improved without worsening at least one others.

Pareto defines a domination relationship between two solutions x_1 and x_2: a solution x_1 **dominates** x_2 if the two following conditions are fulfilled:

1. x_1 is not worse than x_2 over all objectives;
2. x_1 is strictly better than x_2 over at least one objective.

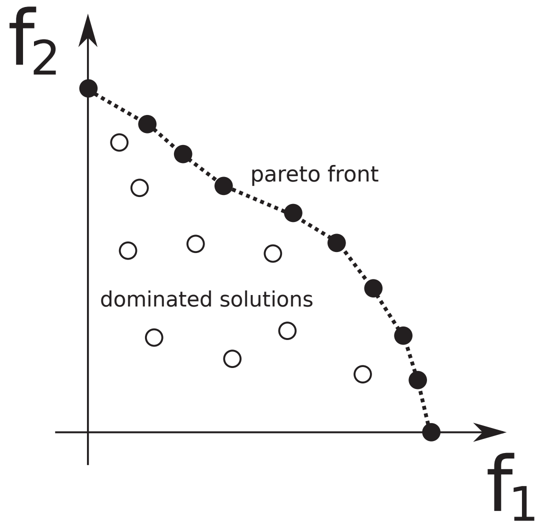


Figure 2.3: The two objectives f_1 and f_2 are being maximized. Non dominated solution (full dots) are situated on the Pareto front (dashed line). Other solutions are dominated, and thus not on the Pareto front.

For n objective functions f_i, x_1 **dominates** x_2 can be formalized as follows:

$$\begin{cases} \forall i \in [1,\ k],\ f_i(x_1) \geq f_i(x_2) \\ \exists i \in [1,\ k],\ f_i(x_1) > f_i(x_2) \end{cases}$$

The Pareto front is the set of all non-dominated solutions (figure 2.3).

Two classes of approaches have been proposed to optimize multi-objective problems:

1. Find a single trade-off solution by reducing the multi-objective problem into a single one
2. Find multiple trade-off solutions

2.1.3.1 Reducing the multi-objective problem into a single one

Aggregate methods These methods consist in grouping objectives into a single one, for instance through linear combinations or minimization of the distance to an ideal solution. Then the problem is reduced to a single objective optimization problem and can be solved with single-objective methods.

The weighted sum method is the most straightforward way to aggregate objectives. Each objective i is given a weight $w_i \geq 0$ which denotes the relative importance of the objectives to others.

$$maximize \sum_{i=1}^{k} w_i f_i(x)$$

with $\sum_{i=1}^{k} w_i = 1$.

The results of this method strongly depend on the choice of the weights w_i. By choosing different combinations of weights multiple times, the Pareto front can be approximated, provided it is convex. Concave parts of a Pareto front cannot be attained by such method. The choice of the weights is not straightforward, and methods with dynamic weights have been proposed (Jin et al., 2001).

Another method consists in trying to minimize the distance between the solutions and an ideal goal point (Charnes and Cooper, 1961). The goal point $\boldsymbol{f^*}$ is a usually a point of the search space which has a maximal possible value for each objective. The optimization problem is then reduced to finding x such that:

$$min \sum_{i=1}^{k} |f_i(x) - f_i^*|$$

Variation around this method include the min-max method (Coello et al., 1995) in which the maximum difference between an objective and its associated goal is minimized:

$$\text{min max}_i \left(\frac{f_i(x) - f_i^*}{f_i^*} \right) \text{ with } i, \in [1, k]$$

ε-constraints The ε-constraints method is different from other aggregate methods because only one of the objectives will be maximized while others are treated as constraints (Ranjithan et al., 1992; Quagliarella and Vicini, 1997).

On the $n - 1$ objectives treated as constraints, ε lower bounds are defined. The algorithm then searches for the solution maximizing the remaining objective in the objective subspace thus defined. Assuming f_1 is optimized while each other objective f_i has a lower bound ε_i, the optimization process can be formulated in the following way:

$$\begin{array}{cc} \max\limits_{\boldsymbol{x} \,\in\, \mathbb{X}} & f_1(\boldsymbol{x}) \\ under\, constraints & f_i \geq \varepsilon_i \quad i \in [2,\ k] \end{array}$$

The chosen objective is the one to be optimized first, but the method can be run multiple times with different objectives to optimize.

Drawbacks There are many drawbacks in using aggregate optimization methods. First the choice of parameters (such as weights) requires *a priori* knowledge on the problem, and a set of parameters is not guaranteed to work on different problems. Then, numerous tests have to be performed in order to determine the relative importance of objectives (Laumanns et al., 2006). Furthermore, these methods find only one trade-off solution per run, whereas the user might want a set of Pareto optimal solutions.

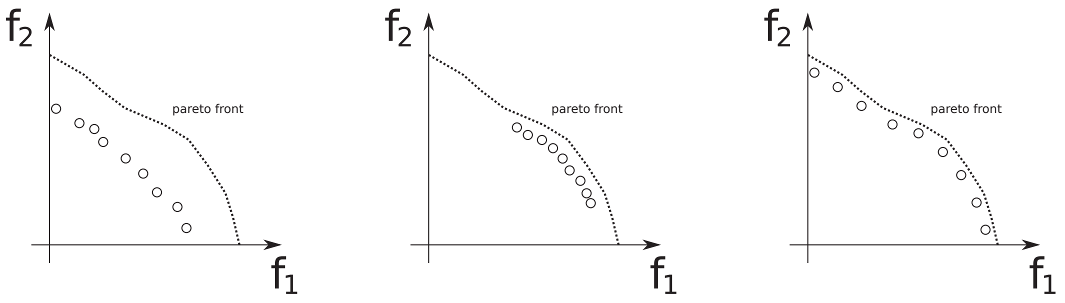


Figure 2.4: Example of a Pareto front approximation using a fictive two-objective approach. From left to right, 1) bad convergence 2) bad dispersion on the front 3) good convergence and dispersion.

2.1.3.2 Pareto dominance based methods

This section describes Evolutionary Multi-Objective algorithms that approximate the Pareto front instead of just finding one trade off solution.

Dispersion and Convergence These two terms are useful for the approximation of a Pareto front in a multi-objective problem. Convergence measures how far the solutions are from the true Pareto front. Dispersion indicates how well-distributed are the solutions on the approximation of the Pareto front (see figure for illustration).

Early approaches A few early methods were developed on the basis of Pareto dominance:

- In Multiple Objective Genetic Algorithm (MOGA) (Fonseca and Fleming, 1993), each solution is ranked depending on the number of solutions that dominates it. A sharing objective is added to improve the dispersion of the Pareto Front.

- In Non Dominated Sorting Genetic Algorithm (NSGA) (Srinivas and Deb, 1994), the population is separated into groups depending on their Pareto domination degree: non-dominated solutions will be on group 1, then solutions only dominated by group 1 will be on group 2, etc. Selection is then defined as in a traditional genetic algorithm, with a stochastic selection and heuristics such as tournament selection.

- In Niched Pareto Genetic Algorithm (NPGA) (Horn et al., 1994), a tournament selection is applied. Two solutions and a small sub-population are chosen at random in the current population. If only one of the two solution dominates the sub-population, then it is selected for next generation. Otherwise, a sharing function is applied in order to chose the individuals.

Those early methods share a common problem of slow convergence. This is partially due to the lack of elitism in the selection process. The dispersion of the resulting population is also not guaranteed.

Modern approaches Those methods focus on a fast convergence to the Pareto front, while maintaining a good dispersion. The following list describes the main algorithms (even though many more exist) and each of the following will be described briefly.

- Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele, 1999; Zitzler et al., 2001);
- Pareto Archived Evolution Strategy (PAES) (Knowles and Corne, 2000)
- Pareto Envelope based Selection Algorithm (PESA) (Corne et al., 2000)
- Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2000, 2002);
- ε-Multi-objective Evolutionary Algorithm (ε-MOEA) (Laumanns et al., 2002; Deb et al., 2005);
- Algorithm based on hypervolume measure (Knowles, 2002; Fleischer, 2003).

Strength Pareto Evolutionary Algorithm (SPEA) In SPEA (Zitzler and Thiele, 1999) the set of all non-dominated solutions are stored in an archive. The fitness of each solution depends the archive, and the solutions of the archive itself also have a fitness value. The fitness of an archive solution depends on the number of solutions it dominates in the population. The fitness of a solution in the population is the sum of the fitness of solutions in the archive that dominate it. The lower the fitness of an individual, the more it is selected. This has two major consequences:

- Non-dominated solutions have lower fitness than dominated ones, thus they are kept in the optimization process.
- Solution that dominate or are dominated by few individuals are more likely to be selected, which means they are in potentially interesting areas of the objective space. This maintains dispersion in the objective space.

As solutions are gradually added to the archive, clustering methods are employed to limit the archive size. If the archive size exceeds the limit of N solutions, only N clusters are kept in the objective space. Only the centroid of each of these cluster is taken into account for computing the dominance relationships (the centroid of a cluster is the point with minimal distance to all solutions in the cluster). This guarantees the high dispersion of solutions in the objective space without having problems due to a too large archive.

However the clustering algorithm used in SPEA has a high complexity and is to be taken into account in the global complexity of the algorithm. Furthermore, the limit size N of the archive can have a huge influence over the results, it should neither be too large nor too small. One of the features of SPEA-II (Zitzler et al., 2001) is that the clustering algorithm has a lower complexity.

Pareto Archived Evolution Strategy (PAES) PAES (Knowles and Corne, 2000) is inspired from evolution strategy (1+1) (Rechenberg, 1973). Consequently it is based on a single solution instead of a population. The solution is then compared to an archive of non dominated solutions. If the new solution is not dominated by a member of the archive, it is added to the archive and all dominated members of the archive are deleted. If the archive exceeds a maximal size N, new solutions are accepted on the basis of a density measure in the objective space. This ensures a good dispersion of solutions.

Pareto Envelope based Selection Algorithm (PESA) The PESA (Corne et al., 2000) population consists of a small internal population and a larger external archive. A crowding measure is applied as the MOEA runs, to maintain selection diversity. This measure is used to copy solutions to the archive in a way similar to PAES previously described.

In PESA-II (Corne et al., 2001) selection is region-based and the subject of selection is now a k-dimensional volume (k being the number of objectives), not just an individual. In other words, it first selects a volume, and then it selects an individual within that volume. The motivation behind this approach is to reduce the computational cost associated with Pareto ranking.

ε-Multi-objective Evolutionary Algorithm, (ε-MOEA) ε-MOEA (Laumanns et al., 2002; Deb et al., 2005) relies on the ε-dominance (Laumanns et al., 2002; Grosan and Oltean, 2004; Deb et al., 2005), a modified version of Pareto dominance.

A solution $\boldsymbol{x_1}$ ε-dominates $\boldsymbol{x_2}$ under the following conditions:

$$\begin{cases} \forall i \in [1,\ n],\ f_i(\boldsymbol{x_1}) + \varepsilon_i \geq f_i(\boldsymbol{x_2}) \\ \exists i \in [1,\ n],\ f_i(\boldsymbol{x_1}) + \varepsilon_i > f_i(\boldsymbol{x_2}) \end{cases}$$

where $\boldsymbol{\varepsilon}$ is a real-valued vector, so that ε_i is defined for each objective.

A solution then dominates all solutions that are not better enough than it on at least one objective. In other words, a solution can now dominate solutions that have a less than ε_i advantage on objective i (see 2.5).

The ε-dominance relationship itself ensures good theoretical properties. The ε-approximated Pareto front has a good convergence and dispersion of solutions with regards to the classic Pareto front defined by the Pareto dominance. The ε-approximated Pareto front also has a bounded size (a classic Pareto front can have

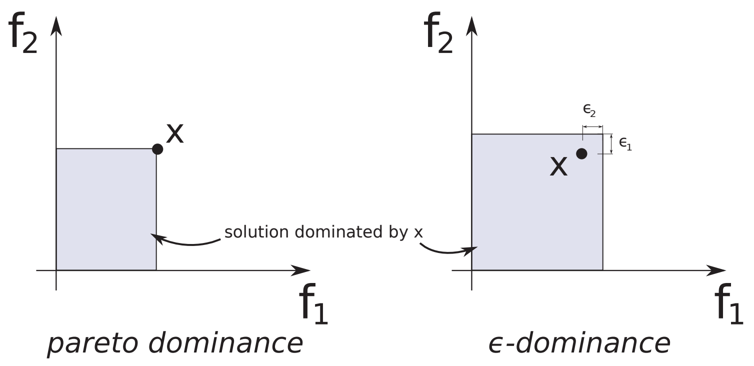


Figure 2.5: (left) Classic Pareto dominance in a 2 objective space. (right) ε-dominance

an infinite number of trade-off solutions), depending on the ε_i used. Note that the ε-approximated Pareto front is not unique.

ε-MOEA (Laumanns et al., 2002; Deb et al., 2005) is an algorithm which uses this new relationship as well as the Pareto dominance. As in PESA, the algorithm relies on a population and an archive of non dominated solutions found. Each step, one solution from the archive and one from the population are selected and used to generate offsprings. Classic dominance relationship is used to add new individuals into the population, while ε-dominance is used for the archive. This ensures that only representative solutions are kept along the front in the archive, while a good dispersion is maintained in the objective space.

Consequently, the ε-MOEA algorithm is efficient with good convergence and dispersion, while having a lower computational cost than SPEA or NSGA-II. However, the choice of ε_i has a strong influence on the final approximation of the Pareto front. Furthermore, the use of constant ε_i for each objective can result in under-sampling in some areas of the objective space, because the density of a Pareto front can vary.

2.1.3.3 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II algorithm was introduced in 2000 (Deb et al., 2000, 2002). It is a fast multi-objective optimization algorithm, that ranks individuals based on their convergence towards the Pareto front and dispersion in the objective space. To do so, NSGA-II divides the population into several fronts based on the pareto dominance relationship. NSGA-II also takes into account the dispersion of the solutions by selects individuals based on the density of their neighborhood. This algorithm will be the basis of all evolutionary experiment in this thesis, and thus will be fully described.

The algorithm starts by initializing a population of N solutions, then for each generation i the following operations are performed:

1. Mix the parent population P_i and the children population C_i of the previous

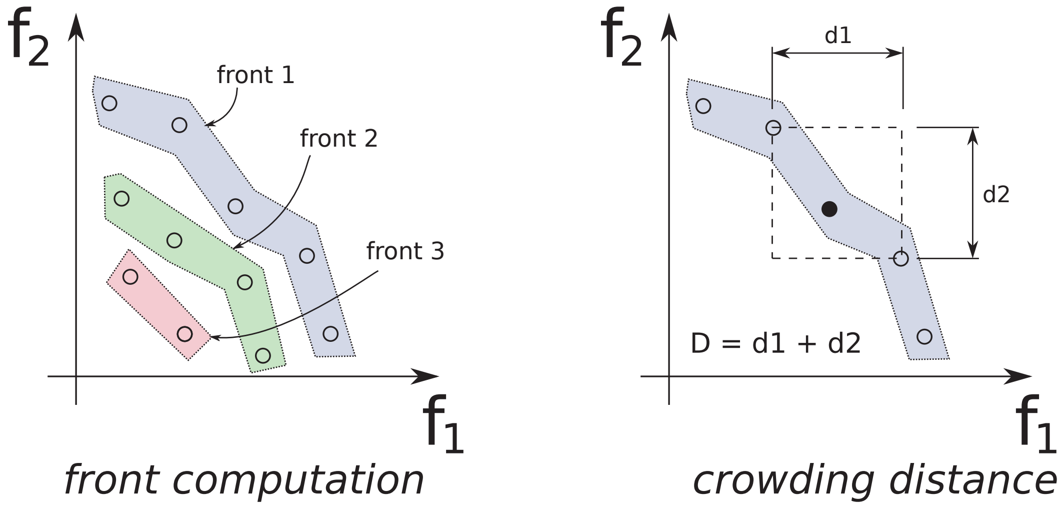


Figure 2.6: Successive front ranking and crowding distance computation in a fictive two-objective NSGA-II optimization.

generation, to obtain a population $R_i = P_i \cup C_i$. of size $2 \times N$. The population is distributed into successive fronts $\mathcal{F}_j$, $j = 0$ being the Pareto front.

2. The new parent population P_{i+1} is first empty and will be filled with the successive fronts. Each successive front is added if the size of the population and the added front does not exceed N. Let j be the last fully added front.

3. Use the crowding distance computation (see below) to include the $M - |P_{i+1}|$ remaining individuals from the front $\mathcal{F}_{j+1}$

4. From the new parent population P_{i+1}, generate the new child population C_{i+1} by tournament, mutation, and cross-over.

The computation of successive fronts is made by finding the Pareto front of the solutions (front 0), and removing them to successively compute the next fronts (2.6, left). Each solution also has a crowding distance which indicates if it is in a dense of sparse region. The crowding distance of the $i - th$ solution on a sorted front $\mathcal{F}_j$ is the Manhattan distance between the solution $i - 1$ and the solution $i + 1$ (2.6, right for a two-dimensional example of the distance). Solutions at one end of the front always are given a maximal crowding distance.

Each solution is then associated with a pair $(i,\ d)$, where i is the rank of the front it is in, and d its crowding distance. In order to generate the child population C_{i+1} from the parent population P_{i+1}, a Crowded Tournament Selection Operator is used. Two solutions $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ (with respective pairs $(i_1,\ d_1)$ and $(i_2,\ d_2)$), are compared on the basis of their front rank and crowding distance. $\boldsymbol{x_1}$ wins over $\boldsymbol{x_2}$ if:

$$\begin{cases} i_1 < i_2 \\ \text{or} \\ (i_1 = i_2) \text{ and } (d_1 > d_2) \end{cases}$$

The original implementation of NSGA-II has a complexity in $O(MN^2)$, M being the number of objectives, and N the size of the population. Using binary trees for the ranking of fronts (Jensen, 2003) can reduce the complexity of the algorithm to $O(Nlog^{M-1}N)$, but the implementation is not straightforward for $M > 2$.

2.1.3.4 Algorithms based on the Hypervolume Indicator

The hypervolume (Zitzler and Thiele, 1999; Knowles, 2002) is an indicator which measures the volume of the dominated portion of the objective space. The Hypervolume is computed relatively to an arbitrary point (easily accessible and dominated). The hypervolume covered between this reference point and non-dominated solutions is the hypervolume indicator (See figure 2.7).

The Hypervolume indicator possesses the feature of strict Pareto compliance. If an approximation of a Pareto front dominates another, then the hypervolume indicator will be larger (Bader and Zitzler, 2011). It was first used as a performance measure to compare different MOEA (Zitzler and Thiele, 1999), but was then used in many optimization algorithms:

- ESP (Huband et al., 2003)
- IBEA (Zitzler and Künzli, 2004)
- SMS-EMOA (Beume et al., 2007)
- SIBEA (Brockhoff and Zitzler, 2007)
- HypE (Bader and Zitzler, 2011)
- MO-CMA-ES (Igel et al., 2007) which is the Multi-Objective version of previously described CMA-ES.

The computational cost of hypervolume is one of its major limiting factor: computing the hypervolume on k solutions with n objectives has an approximate complexity of $O(k^{n+1})$ (Knowles and Corne, 2003). This makes this method infeasible with large populations and several objectives. Different methods have been proposed to reduce this computation cost (Fonseca et al., 2006; Beume and Rudolph, 2006; Yang and Ding, 2007).

Other approaches include the reduction of the number of objectives (Brockhoff and Zitzler, 2007), or an approximation of the hypervolume using Monte Carlo methods (Everson and Fieldsend, 2002; Bader and Zitzler, 2011).

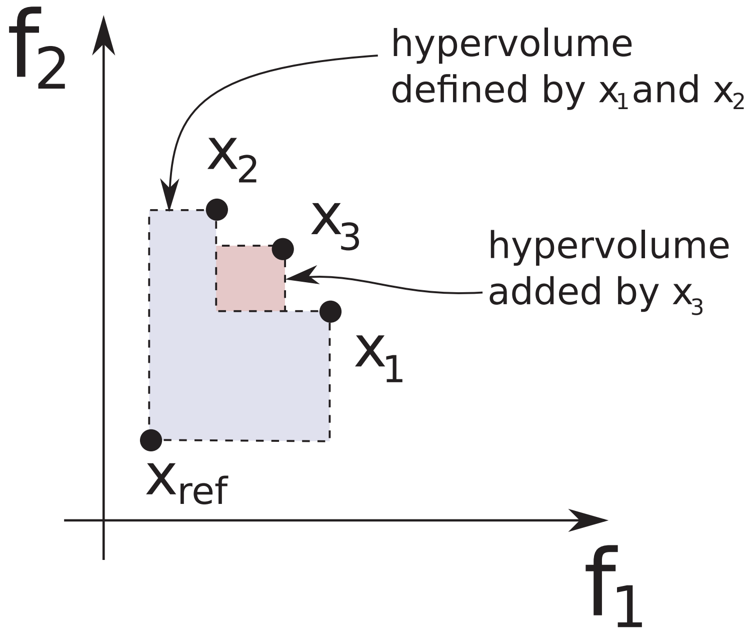


Figure 2.7: Hypervolume computation for 2 objectives. The grey surface represents the hypervolume for the two solutions x_1 and x_2 relatively to x_{ref}. Adding a non dominated solution (here x_3) always increase the hypervolume.

2.1.3.5 Discussion

In the following work, the algorithm used is NSGA-II. There are several reasons for this choice:

- **Performances** It has a low computational cost compared to many other algorithms (including most of hypervolume based ones). This point is of crucial importance because of the rather large populations and number of evaluations required for the evolution of robotic controllers. The algorithm also shows a good convergence and dispersion, as well as a fast convergence.

- **Flexibility** It has no archive, and therefore supports the use of dynamic objectives, whose value do not only depend on the evaluated individual. This includes objectives such as diversity which will be detailed in the later sections 2.2.3.1.

- **Parametrization** It is simple to implement and requires no a priori knowledge on parameters, in contrast with algorithms such as ε-MOEA.

- **Usage** NSGA-II is one of the most used MOEAs (Coello and Lamont, 2004; Coello, 2006).

Furthermore, the influence of different algorithms is beyond the scope of this thesis, and we postulate that our results have little dependence on the choice of the EA.

Limitations One of the major problem of NSGA-II (as well as other Pareto-based MOEAs) is that having more than 3 or 4 antagonistic objectives damages their efficiency (Khare et al., 2003; Ishibuchi et al., 2008). The number of pareto optimal solutions grows exponentially with the number of objectives. The size of the population needed to approximate the Pareto front is then growing exponentially as well. Furthermore, when the size of the population is too small, any potential solution might be non dominated. This results in having a much worse selection process, as all solutions are then considered as equivalent. The selection process is then only based on the crowding distance. However, evolutionary algorithms are, in general, very practical and effective optimization tools for multi-objective problems with 2 or 3 objectives.

Objectives as multiple selective pressures Multi-objective evolutionary algorithm can be useful to simulate the effects of multiple *selective pressures*. In biology, selective pressures are factors fostering selection, such as limitation of resources, threats (predators, diseases, weather).

We assume here that a selective pressure is a force that drives the evolutionary process in a particular direction. In single objecitve EAs, the selective pressure relies on the fitness function. By using a MOEA, several of such pressures can be added to an evolutionary experiment. Each selective pressure can be added as an additional objective in a straightforward way (no extra parameter are needed).

2.2 Evolutionary Robotics

2.2.1 Introduction

Evolutionary Robotics (ER) is the application of Evolutionary Algorithms to automatically design robotic controllers, morphology or both. It was developed in the 1990s (Harvey and Husbands, 1992; Meyer et al., 1998; Floreano and Nolfi, 2000) in order to automatically build controllers for autonomous robots in uncertain environments. Evolutionary algorithms can be seen as a black-box optimization method which suits well the difficult optimization of controllers in complex or dynamic environments.

In ER, the controllers can exploit any specificities of the environment in which they are evaluated, or their control architecture. They can thus find simple and original solutions to robotic problems, without any human intervention (Cliff et al., 1993; Meyer et al., 1998). The following work will not include the design of the morphology of robots. Furthermore, the design of controllers will be limited to artificial neural networks (see section 2.3).

Because of the closed loop between the robot controller and its environment (See figure 2.8), complex dynamics can emerge, even for simple controllers. This was demonstrated in a study by Braitenberg (1986) in which seemingly complex behaviors (associated with fear, love, etc.) emerged from agents with only two sensor neurons and two motor neurons.

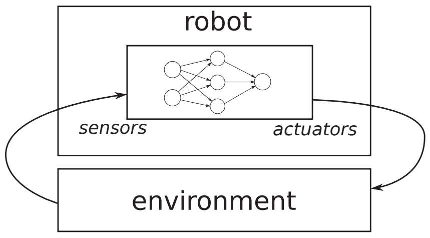


Figure 2.8: Closed sensorimotor loop of a robot in its environment. Even very simple controllers can produce complex and dynamic behaviors in complex environments.

2.2.2 Behavior

Evolutionary robotics experiments rely on the evaluation of performance (fitness) of controllers which are conducted on real robots or in simulations. During this evaluation, the controller is iterated over N time-steps, taking as input the sensors of the robot. The output of the controller represents the motor commands that the robot will perform on the environment, and the resulting behavior of the robot will emerge, as shown in figure 2.9. This figure shows the three classic description levels of the individual (genotype, phenotype and fitness spaces) as well as a new description level: the behavioral space.

The mapping between a robot controller and its behavior is therefore very complex. For instance two very different neural networks might produce similar behaviors, while two very similar networks could produce opposite behaviors (see figure 2.10). For a given problem, understanding the mapping between genotype/phenotype space and behavioral space is an important challenge (Mouret and Doncieux, 2012b; Ollion and Doncieux, 2011). Some recent ER studies use such descriptions to improve the evolutionary process (Lehman, 2008; Trujillo et al., 2008; Gomez, 2009; Doncieux and Mouret, 2010a).

2.2.3 Fitness Functions

Evolutionary algorithms aim at finding controllers that solve best the given task in a given environment (Davidor, 1991; Nolfi et al., 1994). Therefore, by modifying the fitness function, the task or the environment, the researcher can strongly affect the evolutionary process. Optimally, a fitness function in ER should reveal how well the behavior of the controller solves the given task. In practice, many different fitness function types are used in ER experiments, which can be categorized by the quantity of a priori information on the controller that the designer integrates to the evolutionary process (a review of such fitness function was performed in Nelson et al. (2009)). The following list describes a few examples of fitness functions with different *a priori* knowledge incorporated (categories are taken from Nelson et al. (2009)):

- Jakobi (1998) defines *a priori* some characteristics of the desired behavior in a gait evolution for an octopod robot. This corresponds to a *Behavioral*

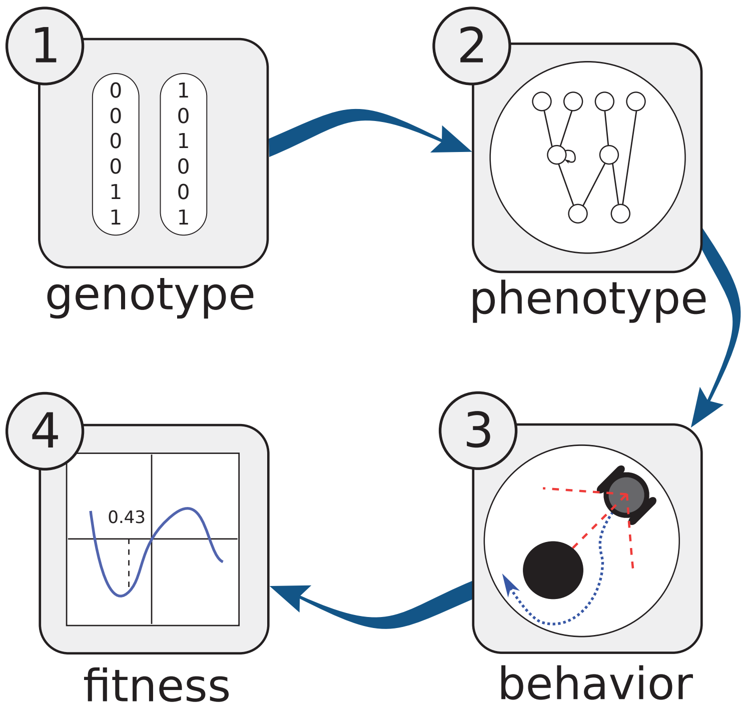


Figure 2.9: Full evaluation of the genotype of an individual in ER. [1 $\Longrightarrow$ 2] The genotype is developed into the phenotype, which is here an artificial neural network. [2 $\Longrightarrow$ 3] The neural network is embodied in the robot (in real robots or simulation) and controls it in a given environment. [3 $\Longrightarrow$ 4] The behavior of the robot is evaluated depending on the tasks and outputs the final fitness score.

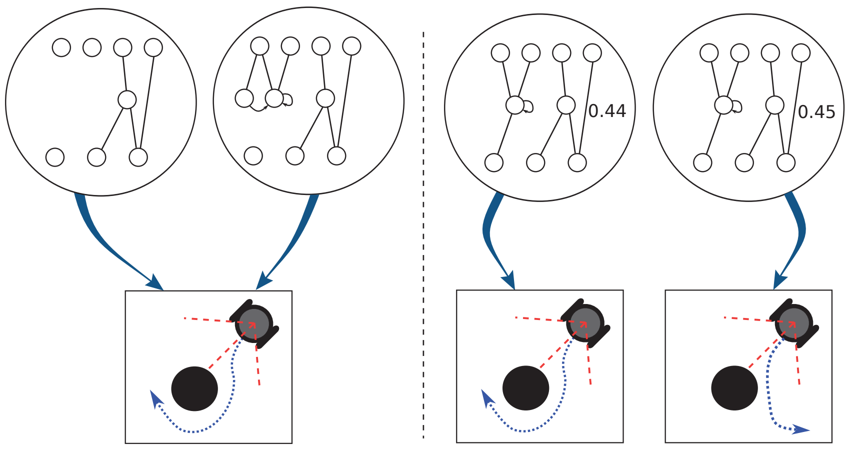


Figure 2.10: Mapping between phenotype and behavior. (left) Two different controller architectures produce the same output behavior. In this figure, this is because a sub-part of the neural network is not connected to the output neurons, but the effect could happen with fully connected networks. (right) Two similar neural architecture (only differing on one weight) produce very different behaviors, due to the interaction with the environment.

Fitness, with high *a priori* knowledge on how to solve the problem.

- Bongard (2011a) uses an environmental incremental fitness: The learning of a behavior in complex environments separated into several sub-parts. First the robot has to solve a task in a simplified environment, then in more and more complex ones. This is a process called *shaping* used in a robotic grasping task. This corresponds to an *incremental fitness*, with moderate *a priori* knowledge incorporated.

- Mouret and Doncieux (2012b) uses an *Aggregate Function* which only rewards partial completion of the task, here only when a robot puts a ball in a basket. That type of fitness gives very little information on how to solve the task.

- Lehman (2008); Lehman and Stanley (2010) have developed a method named *Novelty Search* in which there is no explicit fitness function. Only novel behaviors are rewarded in the considered task, and as more and more original behaviors are found, one might also be a very efficient one. This method can incorporate some *a priori* knowledge into the behavioral description, but it usually ensures very open-ended evolution.

For more exhaustive reviews on ER tasks, see (Mataric and Cliff, 1996; Harvey et al., 1997; Meyer et al., 1998; Meeden and Kumar, 1998; Floreano and Nolfi, 2000; Pratihar, 2003; Walker et al., 2003; Pfeifer et al., 2003; Jin, 2005; Teo and Abbass, 2005; Nelson and Grant, 2007; Doncieux et al., 2011). The work by Nelson et al. (2009) emphasizes that the lower the level of *a priori* knowledge incorporated in ER experiments, the more freedom is left to evolution. In order to find original solutions, or to solve deceptive problems, this might be a crucial factor (Lehman and Stanley, 2010).

The fitness function is associated with a fitness landscape. This landscape represents the mapping between genotypes and fitness scores. Evolutionary algorithms are adapted to find (rather) good solutions even if this landscape is not convex, but struggle if the landscape is too discontinuous, complex and rugged (See figure 5.1). In ER, this problem is even more present because the successive mapping between genotype, phenotype, behavior and fitness can all be very complex and non-linear. This results in the following problems:

- **The bootstrap problem and fitness plateaus:** During the first generations of the algorithm, the randomly generated controllers usually perform really badly, and can obtain systematically the same minimal fitness score (this happens when the fitness function gives little information on how to solve the task). Then no selection at all is performed on the basis of the fitness score: this is referred as *the bootstrap problem* (Mouret and Doncieux, 2009b).

- **Local optima and deception:** The evolutionary process drives the solutions towards higher fitness values. Solutions might then end up in attractive local

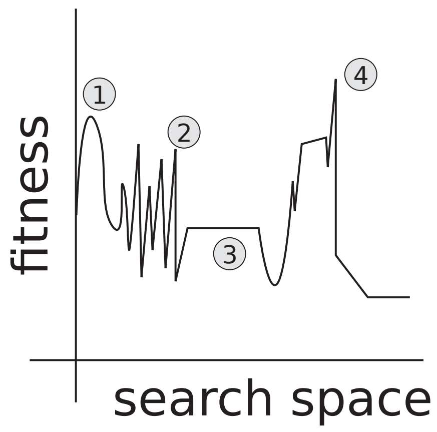


Figure 2.11: Fitness landscape for a fictive optimization problem. Finding the optimal point (point 4 in the graph) is very challenging. Common obstacles to finding such points with evolutionary algorithms are: local optima (point 1); rugged landscapes (rapidly varying landscape, point 2); fitness plateaus (point 3).

optima, and be several mutations away from better solutions. This means that in order to move away from the local optima, solutions with a lower fitness should first be selected, before reaching better ones. This happens a lot when a fitness function is said to be *deceptive*: Poor local optima are easy to find from an evolutionary point of view, while the global optimum (or better local optima) are much harder to find.

In order to solve these problems, several methods have been developed, such as the use of *Helper Objectives*.

2.2.3.1 Helper Objectives

Several researchers have advocated the introduction of additional objectives (Knowles et al., 2001; Jensen, 2004; Mouret and Doncieux, 2008a; Mouret, 2011) to enhance the performance of EAs. Instead of optimizing the main fitness objective, the selection process also selects solutions based on other *Helper* objectives. This process is similar to multi-objective optimization, except that at the end of the optimization, only the solutions maximizing the main objective are kept.

Generalization The ProGab approach (Pinville et al., 2011) enhances generalization of robotic controllers by adding an objective which rewards the generalization ability of the controllers. In order to compute this generalization ability, the controller is tested over many different contexts. Because the computation of the generalization ability for each controller of each generation would be too time-consuming, the generalization ability is only assessed on a single controller per generation. A surrogate model of the generalization ability is used to assess the generalization ability of all the other controllers. In that way, a controller with a behavior close from one which generalizes well will have a good score over the helper objective.

Exaptation Exaptation (Gould and Vrba, 1982) refers to a shift in the function of a trait during natural evolution. Mouret and Doncieux (2009a) take inspiration from this process and define a helper objective that rewards the presence of a sub-network realizing a given sub-function. This concept is applied to a neuroevolution problem in which a neural network has to realize an Xor and Xor problem. The main objective is to compute the Xor and Xor function, while the helper objective rewards sub-networks that realize an Xor function.

Behavioral Diversity Evolutionary algorithms often rely on diversity mechanisms to explore the search space, avoid premature convergence and bootstrap problems. This diversity is often based on genotype (Goldberg and Richardson, 1987; Bäck and Hoffmeister, 1991; Stanley and Miikkulainen, 2002b). Because of the genotype-behavior mapping, it has been advocated to explore the behavioral space rather than the genotype or phenotype space (Lehman, 2008; Mouret and Doncieux, 2012a; Ollion and Doncieux, 2011).

Behavioral diversity rewards solutions that have an original behavior among the population. In Evolutionary Robotics, it has been shown that using a behavioral diversity significantly improved optimization results over a standard diversity method (Doncieux and Mouret, 2010a; Mouret and Doncieux, 2012a). The behavioral diversity relies on a distance function d_b that compares the behavior of different solutions. A typical way of assessing the diversity of a solution x among the population P of N solutions is:

$$div(x) = \frac{1}{N} \sum_{y \in P} d_b(x, y) \tag{2.1}$$

Several behavioral descriptions and behavioral distances have been proposed for ER, and are described in paragraph 2.2.3.2. These distances are a key factor for comparing behaviors, and may incorporate *a priori* knowledge on the given problem. Despite all these works, there is no common agreement at the time of writing on the best way to compare behaviors. A recent contribution uses several of these distance measure and randomly switches from one to another (Doncieux and Mouret, 2013). More in-depth studies of behavioral diversity and distance measures were conducted by Mouret and Doncieux (2012a) and Doncieux and Mouret (2010b). Behavioral diversity is not only used as helper objectives, but also in combination with niching and speciation techniques (Trujillo et al., 2008; Gomez, 2009).

Novelty A more radical approach to solving deceptiveness and bootstrap was developed by Lehman (2008). This approach named Novelty Search only rewards solutions based on their behavioral novelty. This removes any deceptive gradient from a fitness function. The Novelty search relies on a behavioral distance $d_b(x, y)$ between the behaviors of x and y, and on an archive of solutions.

Novelty first computes the “sparseness” s of an individual x by finding the smallest distance between x and its k nearest neighbors in the population and the archive:

$$s(x) = \frac{1}{k} \sum_{j=0}^{k} d_b(x, \sigma_j) \tag{2.2}$$

The sparseness is then used as a ranking method to select individuals (individuals with higher sparseness are more likely to be selected). Last, if this sparseness is over a threshold, the individual is added to the archive.

Novelty has been successfully used in many recent ER and artificial evolution experiments (Lehman, 2008; Lehman and Stanley, 2010).

2.2.3.2 Behavioral Descriptions

The following paragraphs describe some of the recent behavioral descriptions used in ER for diversity and other purposes.

Ad Hoc descriptions Depending on the studied problem, it is possible to define a vector describing the behavior by choosing important parameters. In his navigation task, Lehman (2008) has used the position of the robot at the end of the evaluation as his behavior description. The distance between two behaviors is then simply the euclidean distance between final positions. In a similar way, Doncieux and Mouret (2010b) have used the position of balls at the end of the evaluation in a ball collecting task.

Binary sensorimotor information The sensors and/or motors neurons of the robot are recorded during the simulation and binarized. This results in a large binary vector which characterizes the robot behavior. The behavioral distance can then be the Hamming distance (Doncieux and Mouret, 2010b), NCD (Normalized Compression Distance, an approximation of the Kolmogorov complexity) and Entropy-based distance (Gomez, 2009).

Discretized trajectory The behavior of the robot is characterized by its discretized trajectory (Trujillo et al., 2008; Ollion and Doncieux, 2011). The computation of the distance between two trajectories use the Normalized Levenstein Distance (Yujian and Bo, 2007) (NLD), also known as Edit distance. A closely related measure used by Ollion and Doncieux (2011) is used in this thesis and will be further described.

The position of the robot is sampled over time during the simulation, and discretized over a grid of $N_d \times N_d$ square cells. This results in a vector of discretized positions (See figure 2.12). The distance between two discretized trajectories is inspired by the Normalized Levenstein Distance. The principle is to compute an edit distance between two vectors A and B (of potentially different sizes) of positions.

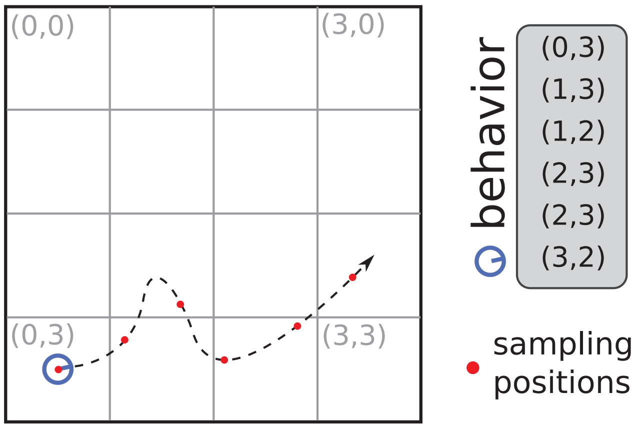


Figure 2.12: Discretized Trajectory behavioral description. Left: a simulated arena in which a robot moves (the dashed line is the robot trajectory). The arena is discretized into a 2 dimensional grid (the robot starts at position $(0, 3)$ in this case). Every T time-steps/seconds (represented with red dots), the robot position is recorded and appended to the behavioral vector. Right: the behavioral vector representing the discretized trajectory of the robot

This corresponds to computing how many operations (insertions, deletions, replacements) one would have to make to transform A to B. Each of these operations has a cost, defined as follows:

- The cost of insertions and deletions of a position is 1.
- The cost of replacement r of one position (x_1, y_1) to another (x_2, y_2) depends on the Manhattan distance between the positions:

$$r = \frac{|x_1 - x_2| + |y_1 - y_2|}{2 \times N_d}$$

  This cost is always below the cost of one insertion plus one deletion.

The lowest cost to transform a vector to another is computed using dynamic programming (similar to the Smith–Waterman algorithm (Smith and Waterman, 1981)). The Edit distance corresponds to this lowest total cost. It is a robust and simple way to compute a distance between trajectories, and has the following advantages:

- Trajectories of different length can be compared.
- The method performs an *alignment* of vectors, so trajectory similarities are taken into account, even if they are shifted in time. This can easily happen in evolutionary robotics, as one robot can have a shifted behavior vector by just waiting before moving.

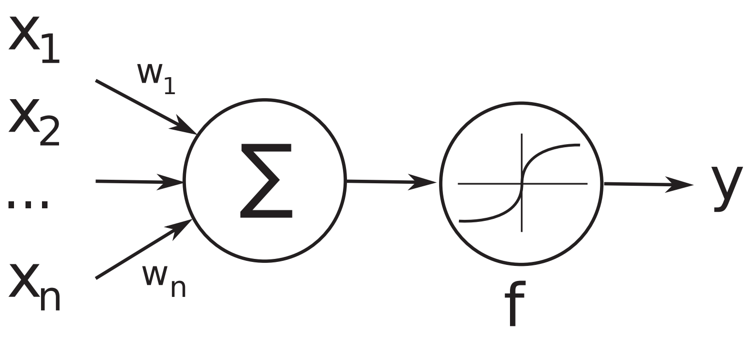


Figure 2.13: Description of a generic neuron. The inputs x_i are affected bu weights w_i then aggregated (here by summation). A second function (here a sigmoid function) is applied before generating the output value of the neuron.

2.2.4 Conclusions

Evolving a robotic controller raises usually very difficult optimization challenges, partly because of the very complex mapping between genotype and behavior and the non-linearity of fitness landscape. The study of the behavioral space and the use of behavioral distances be one of the focus of thesis, whether it be through the use of behavioral diversity or other techniques.

2.3 Artificial Neural Networks

2.3.1 Neuron models

Artificial Neuron An artificial neuron is a model of the functioning of biological neurons. Different models have been proposed, each corresponding to different abstraction level and describing more or less accurately the behavior of real neurons. The following terms are inherited from biology and will be used to describe artificial neurons:

- the *activity* of the neuron is usually represented by the firing rate of the neuron.
- the *synapses* are connections between neurons. A synapse connects a presynaptic neuron to a postsynaptic neuron.

The artificial neuron can be seen as a non linear algebraic function of real valued inputs. A neuron is defined by three main building blocks: a set of connection weights, an activation function, and a threshold (Haykin, 1998; Floreano and Mattiussi, 2008) (see figure 2.13). The following paragraphs describe some of the most common neuron models from the most simple to the most bio-inspired ones.

Mc Culloch's and Pitts McCulloch and Pitts (1943) have proposed the first highly abstract description of a neuron. Each neuron is a Threshold Logic Unit which applies a transfer function f to inputs.

$$y = f(\sum_{j=1}^{n} w_j x_j) \tag{2.3}$$

with y the output of the neuron, f the transfer function, $x_1, \ldots, x_n$ the neuron inputs and w_j the associated weights.

Originally, the Heaviside step function was used in order to produce binary outputs:

$$f(x) = \begin{cases} 1 & \text{if } x \geq h \\ 0 & \text{if } x < h \end{cases} \tag{2.4}$$

where h the threshold of the neuron. Another commonly used function is the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-\lambda(y^{(i)} + \text{bias})}} \tag{2.5}$$

with *bias* being the neuron bias and λ the rate coefficient

Neural Fields Models (NFM) In neural fields (Amari, 1977), equations are defined over a continuous field of neurons, rather than defining neurons individually. They describe the spatio-temporal evolution of coarse grained variables such as firing rate activity in populations of neurons. The models are often parametrized by lateral weight connections, as well as time constants. One of the common use of neural fields is in spatial visual attention. One experiment in chapter 4 is based on neural fields and will be described later with further details.

Radial Basis Function (RBF) Radial Basis Function (RBF) networks (Buhmann, 2003), is another model of neural networks in which the response intensity is inversely proportional to the distance between inputs and precise points (called centers) in input space. These networks have many uses, including classification, function approximation and handwritten text recognition (Lee, 1991). Their goal is to approximate a desired behavior by a set of functions, each one characterized by their center C_i and radius of receptive fields r_i.

Leaky Integrator A Leaky Integrator neuron is a neuron with a first order dynamic: $\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), t)$, with f being a non linear function, usually a sigmoid function. Such systems are useful to model groups of neurons that have linked activity and are more likely to fire at the same time (Dayan and Abbott, 2005). They were also used in systems requiring internal dynamics of neurons such as Echo State Network (Jaeger et al., 2007).

LPDS model (Locally Projected Dynamic System) This model is an extension of a Leaky Integrator neuron, in which the of the stability of networks is easier prove mathematically (Girard et al., 2008). It was used in a model of basal ganglia

(Girard et al., 2008), which will serve as a basis for an experiment in this thesis (See chapter 4. Using numerical solver through the Euler method, the equations can be reduced to:

$$p^t = \sum_{j \in C} w_j y_j^t \tag{2.6}$$

$$a^{t+dt} = \max(0, \min(1, a^t + \frac{p^t - a^t + T}{\tau}.dt)) \tag{2.7}$$

$$y^{t+dt} = \begin{cases} a^{t+dt} & \text{for excitatory neurons} \\ -a^{t+dt} & \text{for inhibitory neurons} \end{cases} \tag{2.8}$$

where:

- y^{t+dt} is the output of the neuron
- T and τ
- p^t is the sum of inputs at time t

The main difference between LPDS and Leaky Integrators comes from the boundaries of the internal state of the neuron. Thanks to the operators min and max, the time needed to converge back to the stable state is lowered (if there were no boundaries, the internal state of the neuron can become very negative, and a large amount of time is required to converge back to steady state).

Spiking neurons models These models describe the spike trains generated by a neuron. The output of the model are spike timings of action potentials. A large number of spikes within a given time window would give an indicator of higher activation. Spiking neurons encode other information than just firing rate, such as the firing timing of single spikes or the temporal coincidence of spikes coming from multiple sources (Singer and Gray, 1995).

In contrast with previous models, spiking neurons can model synchronization between neurons. One of the most well known model is the Hodgkin and Huxley (1952) model, which describes the temporal dynamics of a neuron in different compartments. It aims at reproducing accurately the different modes of activation of the biological neuron.

2.3.2 Neural Networks

A Neural Network (NN) is a description of the different neurons and the connections between them. There are many different topologies which all fit to different problems.

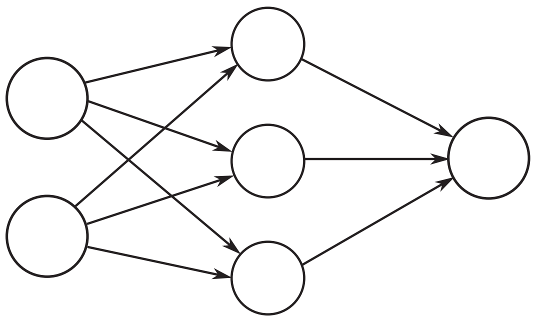


Figure 2.14: Feedforward network with one hidden layer.

Feedforward These networks have several layers of neurons (an input layer, an output layer, and eventually hidden layers; see figure 2.14). The information is fed forward from a layer to the next one. The most well known example of feedforward network is the perceptron or multi-layer perceptron (Rosenblatt, 1958; Rumelhart et al., 1986). The network itself does not induce any dynamics as the neural activations only propagate from the inputs to the outputs.

These networks are used for classification (Lippmann, 1989), signal processing (Cochocki and Unbehauen, 1993), optical character recognition (LeCun et al., 1989), prediction and forecast (Gardner and Dorling, 1998).

Recurrent Neural Networks In contrast with feedforward networks, these networks can have recurrent connections. This can strongly change the dynamics of the network, and lead to networks with self sustained activity, or even chaotic behaviors.

Recurrent networks were used alongside with multi-layer perceptron by Jordan (Jordan, 1986) and Elman (Elman, 1990). Elman networks (See figure 2.15) have a hidden layer and a context layer. The hidden layer is duplicated into the context layer each time-step. The context unit is then fed back to the hidden layer.

Continuous time recurrent neural networks (Beer, 1995; Yamauchi and Beer, 1996) are systems of differential equations that model a recurrent neural network. They can also be seen as recurrent networks of leaky integrator neurons. They can theoretically reproduce any dynamical system, and it was shown that small CTRNN can generate complex dynamics (Beer, 1995, 2006; Bongard, 2011b).

Echo State Networks Echo state networks (Jaeger, 2002) are characterized by a hidden "reservoir", sparsely connected, with allowed recurrent connections (see figure 2.16). The reservoir is initialized at random, with a low connectivity. For standard sigmoid neurons, the dynamics of the reservoir can be written in the following way:

$$\boldsymbol{x}(t+1) = f(\boldsymbol{W}\boldsymbol{x}(t) + \boldsymbol{W}^{\boldsymbol{in}}\boldsymbol{u}(t+1)) \tag{2.9}$$

where $\boldsymbol{x}$ is the internal states of the reservoir neurons, $\boldsymbol{W}$ the connection matrix of the reservoir, $\boldsymbol{W}^{\boldsymbol{in}}$ the connection weights between inputs and the reservoir, and $\boldsymbol{u}$ the input vector. By studying the connection matrix $\boldsymbol{W}$, one can infer dynamical

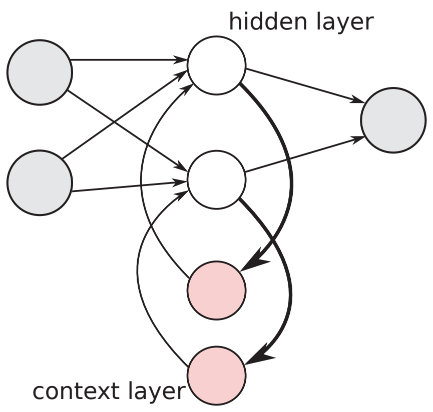


Figure 2.15: Elman Network with two hidden nodes and two context nodes. The bold connections represent weights fixed to 1 while the others can be modified.

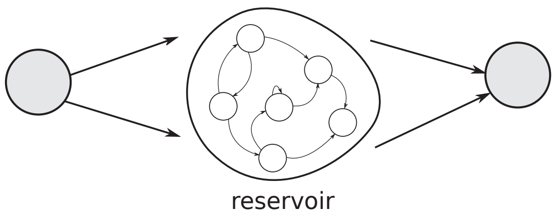


Figure 2.16: Echo State Network principle. The reservoir is sparsely connected, initialized at random, and usually not modified during training. The output is sometimes fed back to the reservoir.

properties of the networks. For instance, the “Echo State property” states that the reservoir will asymptotically wash out information from inputs. This property can be ensured if the spectral radius of $\boldsymbol{W}$ is below 1 (Jaeger, 2002).

These networks have been used in diverse machine learning tasks such as character and speech recognition (LeCun et al., 1989), as well as robotic applications (Antonelo et al., 2007) and memory tasks (Hartland et al., 2009)

Liquid state machines are another reservoir computing method that were developed simultaneously, attempting to build generic models of cortical microcircuits (Maass et al., 2002).

2.3.3 Evolution and Artificial neural networks

A widespread approach in artificial evolution of neural network is to define a fixed topology, and evolve the various parameters such as connection weights and neuron biases. This method was successfully used numerous times:

- with feedforward networks (Floreano, 1994; Bongard and Pfeifer, 2002; Bredeche et al., 2012)
- with recurrent neural networks (Beer, 1995; Ulbricht, 1996; Ziemke, 1996;

Gallagher, 1999; Ziemke, 1999; Ziemke and Thieme, 2002; Mayer et al., 2008; Maniadakis et al., 2010)

- with Elman networks (Miglino et al., 1994)

However, the use of fixed topologies is subject to a few problems:

- It can be difficult to choose the number of hidden nodes required, and usually some exploratory experiments are made to define such parameters
- The topology cannot be changed through the evolution, therefore making it more difficult to study the emergence of specific structures.

2.3.4 Evolution of Artificial Neural Network Topology

The evolution of neural network topology removes the choice of a fixed topology and parameters such as the number of hidden neurons; however it is a challenging problem.

2.3.4.1 Direct encodings

Through mutation and/or crossover operators, the evolutionary process is able to modify directly the topology of the ANN.

Common operators include:

- Mutations that add or remove connections between neurons
- Mutations that add or remove neurons

In that case, the genotype encodes a representation of the topology of the neural network. For instance, the genotypes may contain a directed graph, as well as some parameters. The directed graph is then directly used as a topology for the neural networks, and the parameters become weights and biases. Mutations are applied to both the parameters and the topology.

Crossover operators are much less common in neuroevolution techniques, because crossover implies the separation of the network into sub-parts. Deciding which sub-part of the network can be adapted to another network is a challenging problem. Among the proposed solutions, Mouret (2008) define a method in which the common sub-parts of the parents networks are selected to be switched; Stanley and Miikkulainen (2002b); Whiteson et al. (2005); Stanley and Miikkulainen (2003) propose an encoding named NEAT (Neuro-Evolution of Augmented Topologies) in which crossover between networks are possible (see part 2.3.4.1).

DNN Dynamic Neural Network is a direct encoding used in Pinville et al. (2011); Mouret and Doncieux (2006, 2008a). The network has fixed input and output neurons, and the internal topology can be initialized in the two following ways:

- No hidden neurons and only feedforward connections between inputs and outputs (full connectivity)
- Random topology: a (small) random number of hidden neurons are defined, and randomly connected to inputs and outputs. All hidden neurons should at least be connected to an input or output neuron.

There are five mutation operators defined:

1. modify a connection weight
2. modify a parameter of a node, such as neuron bias
3. add a new connection between two neurons
4. remove an existing connection
5. add a new neuron: an existing connection between neuron n_1 and neuron n_2 is removed, and replaced by a new neuron n with two connections: between n_1 and n, and between n and n_2.

No crossover operator is defined.

NEAT Neuro-Evolution of Augmented Topologies (NEAT) is one of the most used algorithm in neuroevolution (Stanley and Miikkulainen, 2002b, 2003; Moriguchi and Honiden, 2011). The evolutionary process starts with a minimal topology (most often, no hidden neuron, and fully connected network between inputs and outputs). The networks are then complexified through evolution by addition of nodes and connections.

The mutation operators have the same properties as in DNN. Additionally NEAT introduces “innovation numbers” for crossovers and niching techniques. Whenever a new connection appears (through structural mutation), a global innovation number is incremented and assigned to that connection. The innovation numbers thus represent a chronology of the appearance of every connection in the system. These numbers are copied from parent network to offspring. In that way, a correspondence between structural changes is possible, which is used to define a crossover operator.

The innovation numbers are also used to define niches to protect uncommon topologies.

2.3.4.2 Indirect Encodings

In indirect (or generative) encodings, the genotype undergoes a series of transformations to produce the final phenotype (neural network). In particular, in developmental encodings these transformations are often inspired from the development of biological organisms.

The main purpose of these encodings is to evolve quickly larger and more complex structures than using direct encodings (Hornby and Lipson, 2001). They benefit from properties found from biological organisms to engineering (Hartwell et al., 1999; Variano et al., 2004; Hornby, 2005): regularity, modularity, hierarchy, repetitivity and symmetry. These properties can be defined as follows: (taken from (Tonelli, 2012))

Definition 4 (Regularity) *An encoding is regular if the resulting artificial network can be generated from a compact representation (the resulting structure has then a low Kolmogorov complexity)*

Definition 5 (Modularity) *An encoding is modular if the functions computed by the resulting network are localized in the structure of the network.*

Definition 6 (Hierarchy) *An encoding is hierarchical if its development process is applied multiple times to generate the network (i.e., a recursive development process).*

Definition 7 (Repetitivity) *An encoding is repetitive if the resulting network has sub-parts repeated several times with little to no variation.*

Definition 8 (Symmetry) *An encoding is symmetrical if the resulting network has axial or central symmetry.*

The following paragraphs describe a few indirect encodings; a more exhaustive review of generative encodings can be found in (Tonelli, 2012).

Developmental Encodings The development process of all multicellular organisms based is **hierarchical**: the organism develops recursively from a limited number of cells. This natural process inspired cellular encodings (Gruau, 1994) and encodings based on *L-systems* (Hornby and Lipson, 2001; Hornby and Pollack, 2002; Prusinkiewicz and Lindenmayer, 1990). Hierarchical encodings also include scale-free networks which have properties (such as connectivity) independent of the scale or size of the network. Encodings such as DSE (Suchorzewski, 2011) or HyperNEAT claim this property (Woolley and Stanley, 2011). Cellular encodings produce networks that have properties such as **symmetry**, **regularity** and **repetitivity**.

Analog Genetic Encoding (AGE) This encoding (Mattiussi and Floreano, 2007; Mattiussi, 2005) is inspired from biological mechanisms used for the transcription and regulation of genes. Its goal is to generate graphs from a string of characters (inspired by DNA strands) evolved by the algorithm. This encoding was used to generate neural networks, as well as electric circuits (Mattiussi, 2005).

Modular Encodings MENNAG (Mouret and Doncieux, 2008b) is an example of encoding which promotes the reuse of functional modules. Obtaining functional **modularity** is still an open problem (Altenberg, 2004; Mouret, 2008; Hansen, 2003). Most recent works related to modularity are dependent on selective pressures rather than encoding (Kashtan and Alon, 2005; Mouret and Doncieux, 2008a; Clune et al., 2012).

HyperNEAT HyperNEAT was developed by Stanley (2007); Stanley et al. (2009) and is one of the most studied indirect encoding of the previous years (Stanley, 2007; Stanley et al., 2009; Risi and Stanley, 2010; Risi, 2011; Clune et al., 2009; Drchal et al., 2009; Risi, 2011; Suchorzewski, 2011). HyperNEAT is based on the evolution of Compositional Pattern Producing Networks (CPPN) (Stanley, 2006, 2007).

CPPNs are variations of ANNs which differ in the set of activation functions. Instead of only sigmoid or linear functions, CPPNs also include gaussian, sine or any type of function. This enables a CPPN to produce patterns with **symmetry**, **repetitions**, etc. In HyperNEAT, CPPNs are evolved using CPPN-NEAT, a variation of NEAT for evolving CPPNs.

Once the CPPN is evolved, the resulting network is derived from the CPPN in the following way:

- A set of neurons is defined in a substrate (usually 2 dimensional). Each neuron has coordinates in this space.
- For each possible connection between two neurons, the coordinates of the two neurons are fed into the CPPN. The resulting output gives the weight of the connection between the two neurons.
- Additional parameters can also be defined by using a CPPN.

Using this technique, generated networks can have a large number of neurons and connections, and the connection patterns are based on the geometric motifs generated by the CPPN.

EvoNeuro The EvoNeuro Encoding (Mouret et al., 2010) is the basis of some experiments in this thesis, and is described in the following section. It is **regular** and **repetitive**.

2.3.4.3 EvoNeuro Encoding

EvoNeuro is an encoding inspired by computational neuroscience models.

The rationale behind this new encoding is that many computational neuroscience models (Girard et al., 2008; Gurney et al., 2001a; O'Reilly and Frank, 2006) use groups of similar neurons as building blocks instead of single neurons. The models from which EvoNeuro is inspired are mainly basal ganglia models (Girard et al., 2008; Gurney et al., 2001a) and working memory ones (O'Reilly and Frank,

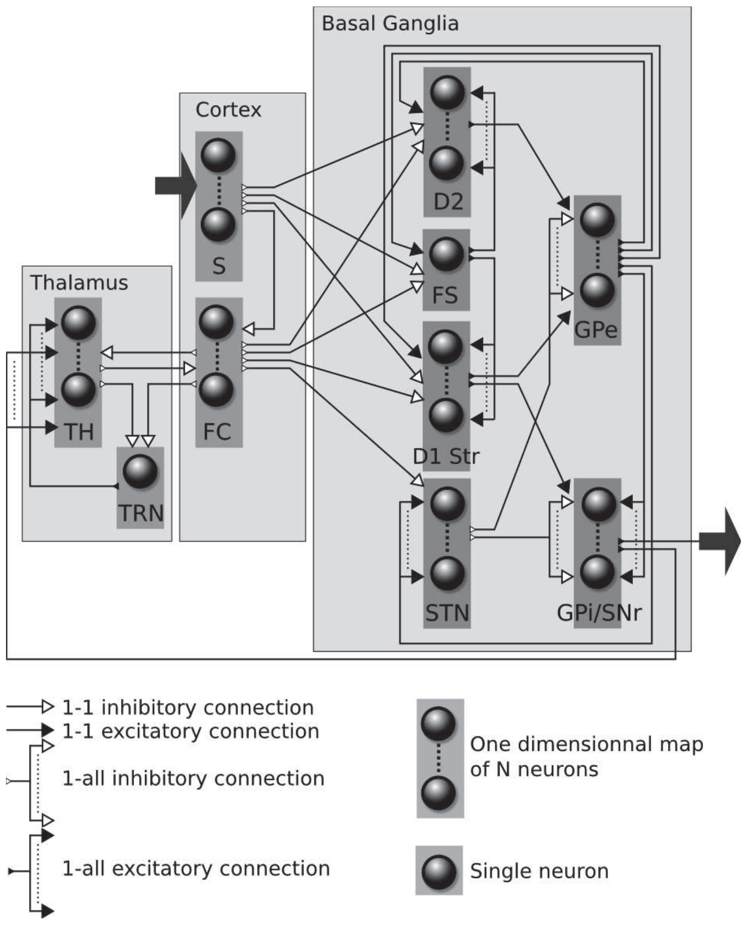

Basal Ganglia
Cortex
Thalamus
D2
S
FS
GPe
TH
FC
D1 Str
TRN
STN
GPi/SNr
1-1 inhibitory connection
1-1 excitatory connection
1-all inhibitory connection
1-all excitatory connection
One dimensionnal map of N neurons
Single neuron

Figure 2.17: Basal ganglia model taken from Mouret et al. (2010). The basic building block of this model is a one dimensional map with a constant number of neurons. All neurons of a map have similar properties (such as bias), and a similar connectivity pattern.

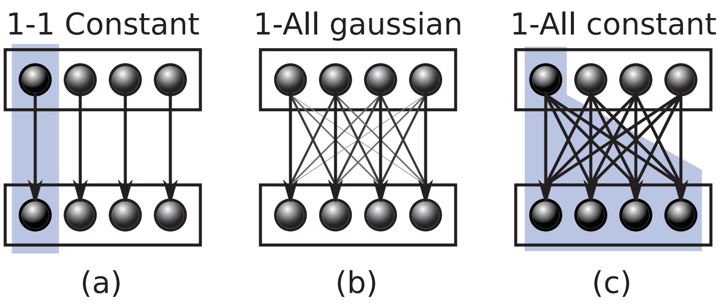


Figure 2.18: Possible connection schemes of EvoNeuro (from Mouret et al. (2010))

2006). Figure 2.17, shows a basal ganglia model from (Mouret et al., 2010). The network has a very regular structure, where channels have the exact same properties. This basal ganglia model realize a *winner-takes-all* function, in which the most active channel is selected while all others are suppressed.

All maps have neurons with similar parameters (such as time constant, biases, etc.), and the connections between maps follow regular patterns. In EvoNeuro, 3 patterns of connectivity between maps are implemented (see figure 2.18). These regularities are present in many brain structures (Alexander et al., 1990).

- 1-1 connectivity: Each neuron of the presynaptic map is connected to a neuron of the postsynaptic map with the same index.
- 1-all gaussian connectivity: Each presynaptic neuron is connected to all postsynaptic neurons. The weights are Gaussian distributed: the neuron i of the presynaptic map is connected to the neuron j of the postsynaptic map with a weight proportional to $gauss(\frac{i-j}{N}, \sigma)$ — where $gauss(x, \sigma)$ is a Gaussian function, and σ an evolved parameter. In that way, the maximal weight is between neurons of same index, while the minimum weight is for neurons with the largest index difference.
- 1-all constant connectivity: Each presynaptic neuron is connected to all postsynaptic neurons with identical weights.

Principle A graph is evolved which is then used to generate the full neural network. The genotype is then a directed graph in which each vertice and each edge has parameters modified through evolution (2 parameters for each node and 3 parameters for each connexion). The neural network can either be initialized with no hidden neurons or with a random topology, as in DNN (see 2.3.4.1).

The directed graph is evolved similarly to DNN: mutations can change parameters of each neuron or connection, as well as the topology (see 2.3.4.1).

In contrast to a direct encoding, the graph is not used as a neural network, but as a blueprint for building a more complex neural structure (see figure 2.19).

During the development phase (from genotype to phenotype, see figure 2.20), the first step is the transformation of each node to either a map of M similar neurons,

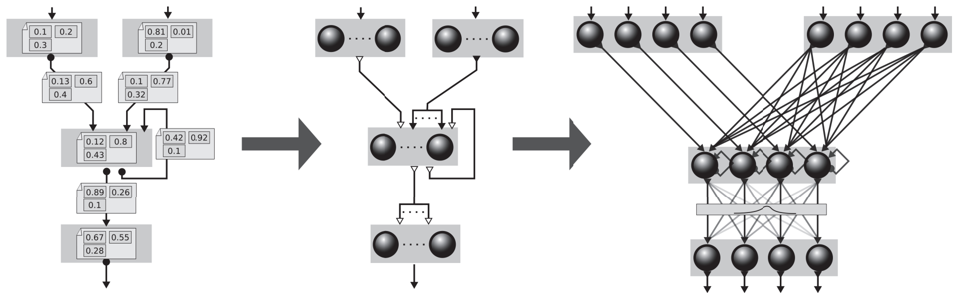


Figure 2.19: General overview of EvoNeuro functioning. From left to right, (1) the genotype is a labeled graph in which the parameters can be modified through evolution (2) Each label is developed into a building block inspired by neuroscience, which gives a general description of the network; (3) For a given map size, each building block is developed into neurons, which outputs the final neural network

or a single neuron — depending on the first internal parameter of the node. The second parameter defines the bias of the neuron(s).

The next step is the development of connections. Depending on the nature of the presynaptic node (whether it is a map or a single neuron), each connection is developed differently as shown in figure 2.20. The connection weights can either be constant or Gaussian distributed (in the case of 1-All connection). In a Gaussian distribution, the weights are distributed based on the difference (in absolute value) between the index of the presynaptic neuron and the index of the postsynaptic neuron.

For input and output nodes, the user decides whether they will be developed into maps or single neurons. This ease the evolutionary process, and ensures that the network always has similar number of input and output neurons.

Because of these maps, the network is much more constrained in its topology and connection weights than a direct encoding. This forces the network to be *regular* and *repetitive*.

Finally, evolving a graph of maps of neurons increases significantly the scalability of the algorithm. The developed network can have hundreds or thousands of neurons, while the direct evolution of a neural network of this size would be unfeasible. Furthermore, the first work using EvoNeuro (Mouret et al., 2010) has shown that the evolution of a *winner-takes-all* function (6 neurons per map) showed very good scaling properties (similar efficiency when changing the number of neurons from 6 to 15).

The EvoNeuro encoding was developed to answer the following three needs:

- propose a neural network encoding with maps of neurons as a building blocks
- use evolution to automatically synthesize computational neuroscience models
- generate artificial agent controllers that can take advantage of larger and more

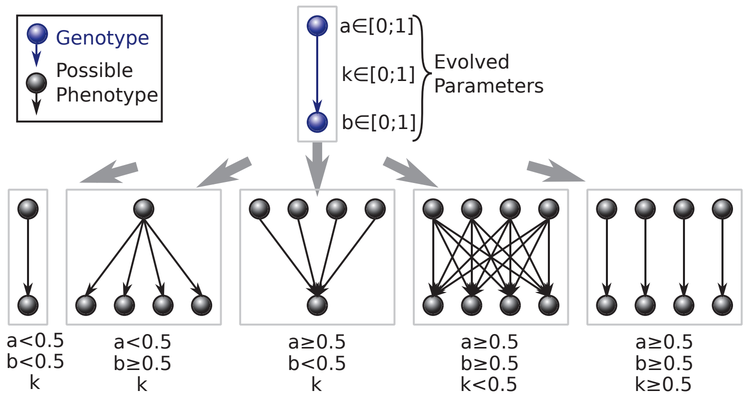


Figure 2.20: Development mechanism of EvoNeuro encoding. Each node is transformed into either a single neuron or a map of similar neurons. In a similar way Each connection is transformed into connection mappings. If the presynaptic and postsynaptic nodes are different in size, the connectivity is always 1-All. If the two nodes are maps, another parameter determines which connectivity is chosen (1-All or 1-1)

complex neural networks

EvoNeuro was initially applied in order to reproduce the behavior of basal ganglia models (implied in action selection behaviors) through artificial evolution (Mouret et al., 2010). Since then, it has been used to develop models of working memory (Pinville and Doncieux, 2010; Tonelli, 2012)

EvoNeuro2 While EvoNeuro provides a functional encoding for developing large and regular neural networks, further features were added for the needs of this thesis. The structure of Head Direction cells model in computational neuroscience have features unavailable in EvoNeuro.

In a model by McNaughton et al. (2006), the authors describe a one-dimensional attractor map model for head direction encoding based on neural integration of head angular velocity signals. In this model, the one-dimensional map has no beginning nor end: it is conceptually arranged into a circle, and has no boundaries (See figure 2.21).

While the goal is not to reproduce exactly such models, EvoNeuro2 takes advantage from some primitives from this literature. These primitives are necessary in some of the experiments presented in this thesis. EvoNeuro2 is an extension of EvoNeuro, with the addition of the two following features:

- Circular wrapping connections: Instead of standard linear neural maps, the maps are considered circular, and the gaussian weights have a torus wrapping (See figure 2.21).

- Connection shift: An additional parameter for each connection is introduced, adding a possible weight shift between the presynaptic and postsynaptic neu-

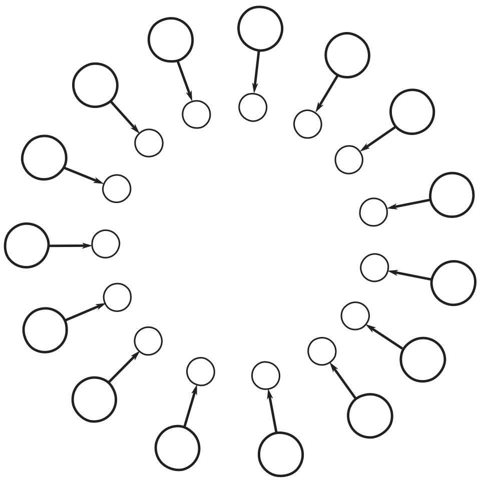


Figure 2.21: One-dimensional circular maps with 1-1 Connection

rons. The offset parameter $p \in [0, 1]$ is converted to an integer shift to the right (See figure 2.22).

2.3.5 Conclusions

This section covered the use of different neural models, neural network structures, and encodings for the evolution of artificial neural networks.

Neuron Models Because evolutionary algorithms often require large amounts of evaluations, computationally intensive models of neurons are rejected. This includes Spiking neuron models. For robotic applications, easy to compute models, such as generic sigmoid neurons and LPDS (Girard et al., 2008) are used in this thesis (chapter 4). Dynamic Neural Fields are also useful to solve visual attention task, and will be used in chapter 4. They are fast to compute provided the number of neurons used to approximate the continuous field is not too high.

Neural Structures In order to study the emergence of specific structures in neural networks, the most straightforward way is to let the evolutionary process decide of the structure. For control experiments, fixed structures such as Elman networks will be used, as these structures have a built-in layer which can be related to an internal representation (Elman, 1990).

Encodings As the experiments in this thesis rely more on the selective pressures than on the chosen encoding, the most easy to implement and fast encodings were employed. In this thesis, the following neuroevolution encodings will be used:

- DNN, in chapter 4, following the work of Ollion et al. (2012).
- EvoNeuro, in chapter 4. Following the experiment of Mouret et al. (2010).

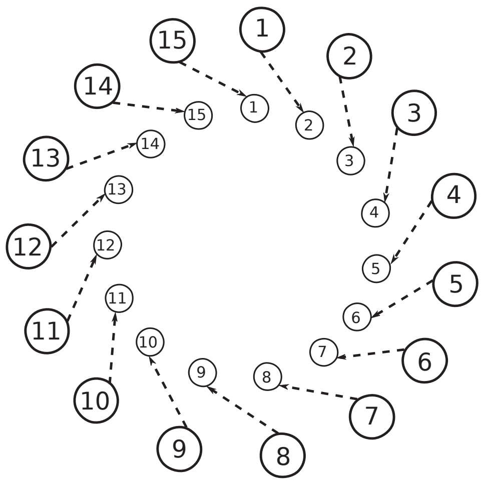


Figure 2.22: Shift of the weight depending on an additional parameter p. Here the map size N is 15, and $p = 0.07$, which corresponds to a shift of $\lfloor 0.07 \times 15 \rfloor = 1$ neuron to the right. As the map is considered circular, the last neuron of the presynaptic map (neuron 15) is projected to the first neuron (neuron 1) of the postsynaptic map.

- EvoNeuro2, in chapter 5. The proposed experiment of Chapter 5 deals with navigation and Head Direction, thus the use of EvoNeuro2 with circular maps greatly improves the evolutionary process.

NEAT was not used in this Thesis, partly because the original implementation of NEAT is dependent on a specific mono-objective algorithmic implementation, which is not NSGA-II. Note that the results should in principle not vary much by using NEAT instead of DNN because of the two following points:

- DNN is a simplified version of NEAT, and most of the principles (such as the mutation operators) are the same.
- Control experiments using both NEAT and DNN were conducted by Mouret and Doncieux (2012a). They showed that, when behavioral diversity was used, DNN was more efficient than NEAT on a ER benchmark.

Chapter 3

Internal Representations

Goals

- Find out properties of internal representations in Evolutionary Robotics and Cognitive Science.
- Design a protocol to determine if a representation carrying such a property is present in a network.

Method

- Review of internal representations in Cognitive Science models and Evolutionary Robotics literature.
- Formalization of the properties in the context of neuroevolution and Evolutionary Robotics into a protocol.

Results

- Two major internal representations are defined: representations with *Noise Resistance* and representation with *Memory*.
- The protocol successfully determines whether representations are present in neural networks.

Contents

3.1 Internal Representations and Cognitive Behaviors 46
3.1.1 Internal Representations 46
3.1.2 Examples of Reactive and Cognitive Behaviors 47
3.1.3 Internal Representations in Computational Neuroscience . . . 49
3.2 Internal Representations in Evolutionary Robotics 51
3.2.1 Representation-less ER tasks 52
3.2.2 Representation-Hungry Tasks 52
3.2.3 Tasks with unclear Representation needs 54
3.2.4 Conclusion 56
3.3 Testing Characteristics of Internal Representations 56
3.3.1 Measuring Representations in the literature 56
3.3.2 Properties of Internal Representations 57
3.3.3 Quantitative tests 58

3.3.4 Qualitative tests 61
3.4 Analysis over Existing Networks 62
3.4.1 Representations in classic networks 63
3.4.2 Visual Attention 65
3.4.3 ER task: Hard Ball-Collecting Task 67
3.4.4 Conclusion 71

3.1 Internal Representations and Cognitive Behaviors

3.1.1 Internal Representations

Traditional theories in cognitive science consider internal models as the basis of cognitive agents' actions (Craik, 1943; Tolman, 1948; Johnson-Laird, 1983). Those models are described as mental maps that incorporate symbolic representations. In the seminal work by Craik (1943), states that the mind constructs "small-scale models" of reality that it uses to reason, and to anticipate events. Tolman (1948) describes cognitive maps, that take part in animal navigation and path planning. These maps are the basis of models and cognitive functions and are called **internal representations** (Tolman, 1948; Marr, 1982). In the case of Tolman's maps, the representations are then symbolic, and their content was expected to be easily interpreted. A review of such mental models was conducted by Johnson-Laird (1980).

In the late 1980s, Harnad (1990) introduces the symbol grounding problem, which questions the presence of symbolic representations. In the 1990s, cognitive science has seen a new emphasis on the embodied and situated intelligence, including works by Brooks (1991b); Varela et al. (1991), and rejecting the symbolic only views. They argue that "the world is its own best model" (Brooks, 1991b), and emphasize intelligence without representation, with reactive behaviors. This has led to an anti-representationalist view and created debates in the cognitive science and artificial intelligence communities. However, Clark and Toribio (1994) argues that there might not be a representational/non-representational dichotomy, but rather different degrees and types of possible representations, from no representations at all to sophisticated models. In embodied views of cognition, cognition can rely on prediction and internal representations, however the representations are based on sensorimotor codes rather than symbols (Wolpert et al., 1995; Keijzer, 2002; Pfeifer and Bongard, 2006).

The term representation can be misleading and has been criticized. Keijzer (2002) argues that "Internal Control Parameters", which can be interpreted as specific biological internal states, are *not* representations and advocates for the use of both representational and non-representational internal states in embodied views of cognition. However, this debate is beyond the scope of this thesis. The definition of representations in this work will rather be focused on their *function*.

3.1.2 Examples of Reactive and Cognitive Behaviors

What role do these internal representations have ? They have been associated with Cognitive behaviors (see, e.g. (Craik, 1943; Tolman, 1948; Johnson-Laird, 1980; Marr, 1982)). Shettleworth (2009) has studied cognition in the animal kingdom, which include very diverse nervous system structures and behaviors. For instance, behaviors purely based on reflexes are called *reactive behaviors*; behaviors involving complex computations and integration of information over time are *cognitive behaviors.*

The following paragraphs illustrate some reactive and cognitive behaviors in animal species. In particular, navigation provides a good context for the study of reactive and cognitive behaviors, because the trajectory of studied animals can easily be recorded in nature, or by cognitive scientists in setups such as mazes.

Reactive behavior and taxis Reactive behavior are behaviors only depending on current sensor information, and not integrating other information. For instance, chemo-taxis is the process of moving towards a gradient of concentration of a given chemical substance. An example of such kind of behavior would be the chemotaxis of bacteria (for instance in *Escherichia Coli* (Berg et al., 1972)). In taxis, the navigation is then local and does not rely on past sensor information.

Another example of simple navigation behavior can be found in some insects, such as the African dung beetle *Scarabaeus zambesianus* (Dacke et al., 2003). Behavioral experiments show that in some conditions, the dung beetle has to move along a straight path. In order to do so, it heads towards the direction of polarized light (sunlight, or moonlight (Dacke et al., 2003)). Changing the polarization of the light (using a polarized screen) makes the beetle instantly change its direction. In that case, the beetle's heading solely depend on its light polarization sensors, and thus it has a reactive behavior. However, other navigation strategies of the beetle might include non-reactive behaviors.

Non-reactive behaviors Insects such as bees and ants exhibit *homing* behaviors: from any place they wandered, they have to find a way back to their nest. It has been shown that one of their strategies relies on *dead reckoning* (Cheng et al., 1987; Wehner and Srinivasan, 2003). This means that while navigating away from their nest, they remember information about direction and distance towards their nest, and mostly use those information to get back to their nest. By catching an ant just before it starts its homeward journey and artificially releasing it several hundred meters away, the ant does not head towards the nest (Wehner and Srinivasan, 2003). Instead it heads in a direction similar to that which it would have headed from the point of capture (See Figure 3.1). Upon reaching its goal — where the nest should have been — the ant starts circling. This behavior is non-reactive: the ant *integrates* its path and remembers information necessary to come back to its nest. Little to no local cues are used to decide of the direction.

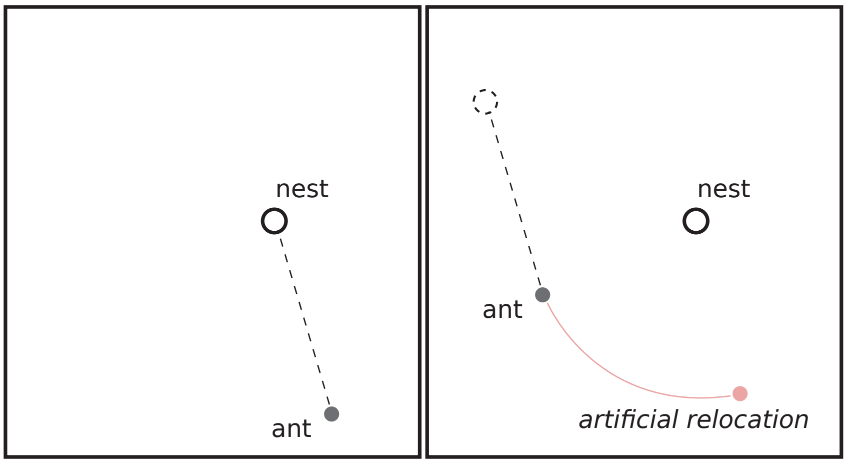


Figure 3.1: Illustration of dead reckoning and path integration for desert ants *Cataglyphis albicans*. Left: an ant is captured away from its nest, and has integrated its path from the nest. Right: the ant is relocated, and starts heading towards where it would have headed from the point of capture.

Advanced cognitive navigation Animals have many different sophisticated navigation strategies, some of which lead to reactive behaviors, while some are more complex (a review of cognitive navigation in different animal kingdoms can be found in Shettleworth (2009)). For instance, many researchers have focused on the navigation strategies of rodents (Redish, 1994), and uncovered brain regions that can be considered as internal maps used for navigation. The navigation strategies involved in rodents can include some local navigation, or sole path integration for homing (*dead reckoning*), but can be much more complex (Redish, 1994; Trullier et al., 1997). For instance rats have been shown to rely on landmark learning/recognition, and path planning.

Many Computational Neuroscience models of these functions have been proposed and rely on internal maps (Tolman, 1948; O'Keefe and Nadel, 1978; Touretzky et al., 1993). Some evolutionary robotics experiments have also included (much simpler) representations in the control architecture. Figure 3.2 details a summary of the current position of this development of ER from reactive, feed-forward networks to systems with internal representations. While there was some attempts to define minimally cognitive behaviors (Beer, 1996), there is no clear way to determine whether representations are present within a network. The two next sections briefly review the use of internal representations in 1) Computational Neuroscience; 2) Evolutionary Robotics.

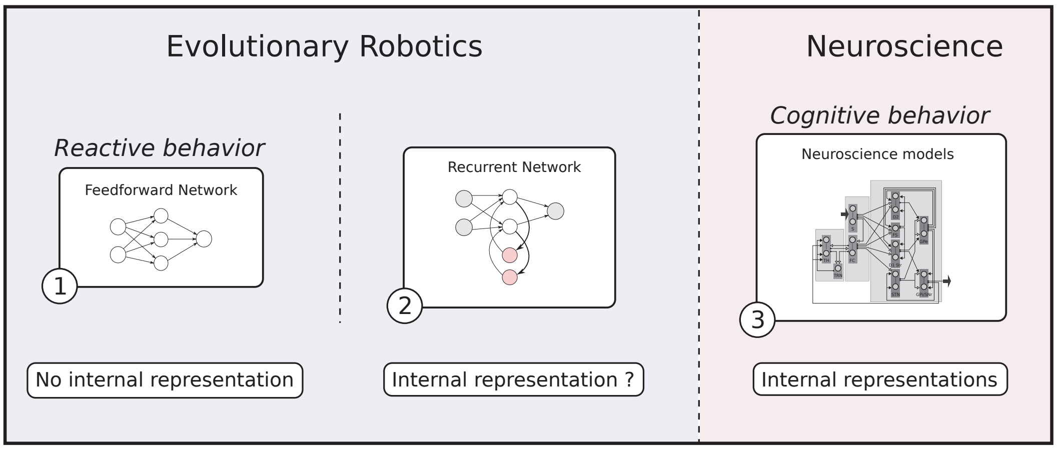


Figure 3.2: Overview of internal representations, in Evolutionary Robotics (left) and Computational Neuroscience (right). 1) Evolutionary Robotics methods successfully solve tasks involving reactive behaviors, without the need for any internal representation. 3) Neuroscience studies cognitive systems and models them using internal representations. 2) Recurrent topologies evolved in many ER experiments; do they have internal representations ?

3.1.3 Internal Representations in Computational Neuroscience

3.1.3.1 Examples of tasks requiring internal representations

The following Computational Neuroscience review focuses on the use of internal representations in the different models. While the encoding and the use of these representations can be very different from a model to another, common properties will be highlighted. The following list of cognitive functions is not exhaustive, nor mutually exclusive, as it is probable that some of these functions might be linked (see for example Deco et al. (2005)).

Working Memory Working memory is a short-term memory, which is referred as a “temporary storage and manipulation of the information necessary for complex cognitive tasks” (Baddeley, 1992). Working memory then is believed to be the basis of multiple cognitive processes. Several neural models of memory and working memory imply a *buffering* of the information in an internal neuron or group of neurons. The most simple information is a bit of information, which can be stored in a single internal neuron. This corresponds to a very small and simple internal representation.

Attention Focus Attention focus is the process of focusing visual attention to a single cue, when multiple cues are available in noisy and distracted environments (Rougier and Vitay, 2006). Several models of visual attention rely on neural fields Amari (1977). The model by Rougier and Vitay (2006) consists of a visual 2D input map, and a 2D internal map. A visual cue (a Gaussian activity "blob"-shaped), noise, distractors, and is subject to occlusions. The internal map must be able to focus to one single visual cue and have a well defined internal activity and shape. The connectivity of neurons in the internal map involve lateral inhibition and local excitation in a "Mexican hat" connectivity pattern (Müller et al., 2005). This connectivity enables a spontaneous stabilization and memorization of information, provided the right parametrization (Rougier and Vitay, 2006; Quinton, 2010b).

Action Selection Action selection is the process of choosing a single action when given multiple channels of inputs with different strength. The basal ganglia is a brain structure believed to be involved in action selection and modeled by Redgrave et al. (1999); Gurney et al. (2001b); Girard et al. (2008). These models involve a dynamical system, which converges to a map with a channel singled out — the selected channel. Internal representations (maps of neurons), which contain information about the selection, are involved in this computation.

3.1.3.2 Internal models for navigation

Cognitive Navigation is the process of recognizing places, following landmarks, integrating path, and path planning. Since the cognitive maps described by Tolman (1948), many different computational neuroscience models for navigation have been described. Those models are intensely based on internal representations, which include:

- Place Cells (O'Keefe and Nadel, 1978)
- Head Direction Cells (Taube et al., 1990)
- Grid Cells (Hafting et al., 2005)

For instance, a model of the functioning of path integration in mammals, includes Place and Head Direction cells (McNaughton et al., 1996). More recently McNaughton et al. (2006) describe models of this ability implying Grid Cells.

Place cells are neurons firing at an increasing rate when the animal is in a specific place (the "place field" of the cell) (O'Keefe and Nadel, 1978; Muller et al., 1996). They are believed to be the basis of the representation of space in some animals (Touretzky et al., 1993). The place cells can encode more information than just a position: for instance, a place cell can be sensitive to a particular direction (see Redish (1994) for a review). A place cell should be active when the animal revisits the place field associated with it. Those place cells are also involved in navigation such as path planning (See Trullier et al. (1997) for a review).

Head Direction cells (HD cells) are a population of neurons, each having different firing rates when the animal's head point towards a specific direction (Taube

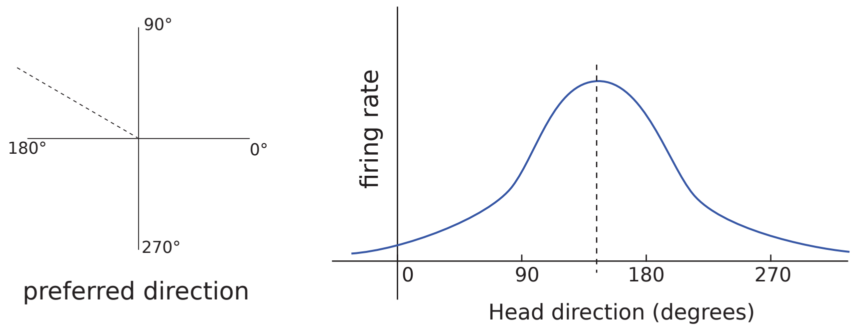


Figure 3.3: Firing rate of a model of Head Direction cell. (Left) Preferred firing direction of the cell. (Right) Idealized firing scheme of the cell in function of the Head Direction.

et al., 1990). The population is believed to encode the direction the head in mammals such as rats, each cell having its preferred direction (See Figure 3.3). The Head Direction cells system can be interpreted as a 1-dimensional circular map of neurons which represents the Head Direction of the animal (Taube et al., 1990). The encoding and decoding of the head direction (a continuous information) is realized through population coding (Deneve et al., 1999, 2001). Each neuron have a Gaussian firing rate centered around its preferred direction; the firing rate of the whole population is used to decode the actual head direction. Population coding was shown to encode information reliably in noisy conditions (Deneve et al., 1999, 2001).

Head Direction Cells (or HD cells) are sensitive to vestibular input (inertial compass), as well as visual inputs (Wiener and Taube, 2005). In this model, even if the visual inputs are completely lost (when the animal is in the dark for instance), the Head Direction system maintains its activity and is able to encode the direction of the head.

Grid cells are neurons typically found in dorsomedial enthorinal cortex, which fire in regular and geometrical way (Hafting et al., 2005). They appear to encode abstract spatial structure and are believed to encode a cognitive representation of euclidean space (Hafting et al., 2005).

3.2 Internal Representations in Evolutionary Robotics

In the light of the cognitive functions described in the previous sections, the following general definition of representation will be used in this work:

Definition 9 (internal representation) *An* ***internal representation*** *is a piece of information located in internal neurons which 1) is dependent on inputs; 2) cannot be deduced only from the values of the inputs at a particular time.*

Through this definition, several points can be made:

- Reactive agents do not make use of internal representations — their potential internal nodes can be deduced from a direct combinations of inputs.

- An internal circuitry not connected to inputs is not considered as a representation.

While the models from computational neuroscience have complex internal structures, many ER experiments rely on much simpler control architectures. This section reviews different tasks — in the light of the present definition of internal representation — in the past ER experiments. This review is restricted to the evolution of artificial neural networks in ER.

Definition 10 (representation-hungry) *A representation-hungry (Clark and Toribio, 1994) ER problem corresponds to a task in which an internal representation in the controller is necessary for solving the problem.*

In order to study the nature and emergence of representations, *representation-hungry* tasks must be chosen; however the definition of such tasks is not always straightforward.

3.2.1 Representation-less ER tasks

Obstacle avoidance and Navigation Many ER tasks involve the optimization of a controller for a robot to navigate in a given environment, while avoiding obstacles. In (Lund and Miglino, 1996), a Khepera robot is used, controlled by a simple neural network with no hidden layers. This behavior is fully reactive, and can be very efficient for obstacle avoidance or wall avoidance (Zufferey et al., 2002). Other works imply photo-taxis behavior and obstacle avoidance (Miglino et al., 1998; Seok et al., 2000; Watson, 2002; Parker and Georgescu, 2006). In these tasks the robot has to move towards a light point, avoiding obstacles or walls along the way.

Interestingly, some navigation tasks can sometimes be solved by the evolutionary process using complex internal dynamics, without the help of sensors (Lehman et al., 2012). The internal structures involved in these dynamics are then not considered as representations in the light of the present definition. The author then encourage the “reactivity” of systems to select solutions that do not only rely on internal dynamics without taking into account the sensory information.

3.2.2 Representation-Hungry Tasks

3.2.2.1 Minimally Cognitive Behavior

Beer (1996) defines the idea of Minimally Cognitive Behavior, in which the controllers are evolved in tasks complex enough to be *representation-hungry*. Therefore, the task should raise interesting cognitive questions while remaining simple enough for analysis.

In order to evolve controllers with Minimally Cognitive Behaviors, Beer (1996, 2003) proposed a framework in which a visually guided agent has to perform the following tasks:

- Orientation: the agent has to orient towards a visual stimulus, and adjust their position to catch falling objects. Gallagher (1999); Slocum et al. (2000) have proposed an extension of Minimally Cognitive Behaviors in which the agent is evolved to catch falling objects, having only briefly seen their positions.

- Discrimination: the agent has to visually discriminate between different objects

- Pointing: the agent has to point its 1 degree-of-freedom manipulator towards the center of an object

In some of these experiments however, Beer concludes that it is difficult to understand whether an internal representation is encoded in it, or if the success of the agents is just based on coupled dynamics with the environment, merely instantiated by the CTRNNs (Beer, 2003).

3.2.2.2 Predictive Models

In the experiments by Nolfi and Tani (1999); Tani and Nolfi (1999), mobile robots navigate in an environment with several rooms. The robot has the possibility to anticipate and predict the next sensory input. It uses an internal model with a Recurrent Neural Network, with context units and a gating system. A later work by Gigliotta et al. (2011) also uses the same kind of network with specific context units which predict the input. In that case, the network has to predict the input, so that when the input disappears the robot can still perform the task. An ad hoc architecture made it possible to realize this prediction, even when the inputs are periodically removed (input occlusions), and in the presence of noise.

3.2.2.3 Evolution of representations in a discrete environment

Edlund et al. (2011); Marstaller et al. (2012) are explicitly interested in the evolution of representations in agents, in a discrete environment. The experiments use a recurrent artificial network as well as a network of hidden Markov gate. In the artificial neural network experiment (Marstaller et al., 2012), the network internal structure consists of six hidden units and four recurrent units. This architecture was chosen for the short-term memory potential. While the networks are shown to have representations, they do not improve much during the evolutionary process, and the behavioral analysis of the internal nodes is therefore made on the hidden Markov gate architecture. Furthermore, the same methodology wouldn't apply to a continuous evolutionary experiment.

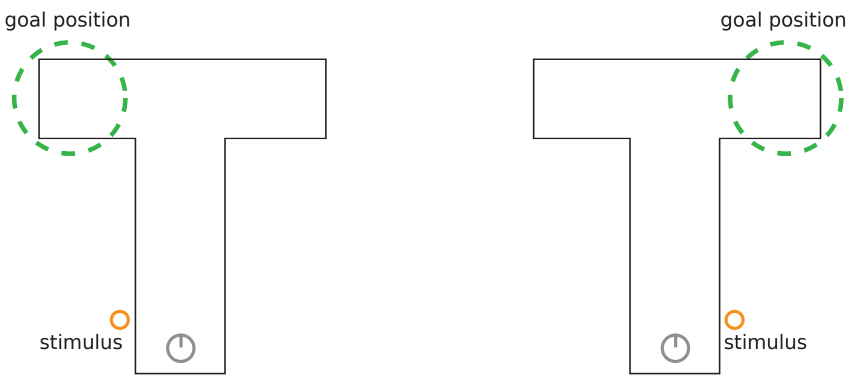


Figure 3.4: Illustration of a T-maze for the road-sign problem. The robot is located at the bottom of the T-maze, and receives a stimulus (orange light to the left or to the right). This stimulus quickly disappears. The robot has to turn left or right depending on this stimulus.

3.2.3 Tasks with unclear Representation needs

In several other ER tasks, the need for internal representation is much more difficult to assess. These tasks have control architectures that could allow for internal representations, but in which it is unclear whether these representations are actually needed.

3.2.3.1 Delayed Response Task

In a delayed-response task, there is a delay between the presentation of a stimulus and the expected response of the robot. The robots have to acquire a form of memory to solve the tasks. The controllers of the robots usually have a fixed topology with specific neurons corresponding to memory states, or use CTRNNs.

Road-Sign Problem The Road-Sign Problem is one of the first task used in ER to test the synthesis of a memory (Ulbricht, 1996; Jakobi, 1997; Rylatt and Czarnecki, 2000; Linaker and Jacobsson, 2001; Bergfeldt and Linaker, 2002; Blynel and Floreano, 2003; Kim, 2004; Ziemke and Thieme, 2002; Ziemke et al., 2004). The Road-Sign Problem is a delayed response task, in which the robot has to choose a direction depending on a stimulus received some time before. The simplest Road-Sign Problem implementation is the T-maze problem (See Figure 4.1), in which a robot has to turn left or right depending on a light stimulus. The road-sign setup was used to test generalization ability (Ulbricht, 1996); transferability from simulation to reality (Jakobi, 1997; Koos et al., 2012); synthesis of memory (Ziemke and Thieme, 2002; Kim, 2004) and meta-cognition (Maniadakis and Tani, 2009).

This task is usually considered to require a short-term memory, as the agent must remember which stimulus (left or right) it received in order to turn left or right. However, evolutionary experiments (Ziemke et al., 2004; Pinville, 2013) showed that the T-maze navigation task does not always require internal memory to be solved.

The agents instead can use behaviors such as wall-following: when they receive the stimulus, they directly go towards the left or right wall of the corridor, and then follow the wall. In that case, an external memory (See for instance (Hutchins, 1995)) is used: the agent's own position with respect to the wall; instead of an internal memory or representation.

Other Memory Tasks Other evolutionary experiments include the work of Capi and Doya (2005), in which a robot has to alternatively navigate to a food place, a water place and its home. This task relies on a specific architecture in which some neurons are explicitly binary memory units. For a more complete review of working memory in a context of ER, the reader is referred to the work of Pinville (2013).

As shown by Nolfi (2002); Ziemke et al. (2004), relatively complex tasks (such as memory tasks) can often be solved by reactive agents which exploit the sensorimotor loop from the interaction with the environment. In that case, those relatively complex tasks do not require any internal representation, and the evolutionary process may then lead towards solutions that are reactive and solve the task (Pinville, 2013).

Homing The robot has to find its way back home, and the localization of its home can't be directly deduced from a light source as in photo-taxis (it might not be indicated at all). Vickerstaff and Di Paolo (2005a,b) use a genetic algorithm to evolve neural models of path integration, with particular emphasis on reproducing the homing behavior of *Cataglyphis fortis* ants. They do not impose any particular system for the internal representation of the agent's home vector. The agents are then able to perform basic path integration. However, the dynamics of the behavior is strongly influenced by the dynamics of the leaky integrator neurons; it is therefore difficult to study the presence of representations.

Locomotion and Legged motion Many evolutionary robotics experiment imply the use of recurrent neural networks and therefore internal dynamics. The optimization of walking gaits take advantage of such networks, whether the gait is evolved on a hexapod (Filliat et al., 1999), quadripod (Hornby et al., 2000) or biped robot (Reil and Husbands, 2002). Commonly used neural structures are Central Pattern Generators (Billard and Ijspeert, 2000). These structures were not analyzed in the light of internal representations, but some of them might satisfy the definition of internal representation given here. However, in several cases, the internal neurons are solely used to produce oscillatory dynamics necessary for the walking behavior. They are then totally independent from the inputs and cannot be considered as internal representations.

Foraging Tasks In these tasks, the robots must find objects in an environment, gather them and eventually drop them in another place. Tasks such as "peg-collecting" (Nakamura et al., 2000), or "object pushing" (Ishiguro et al., 1999;

Hornby et al., 2000) have been treated, as well as ball collecting Mouret and Doncieux (2012a); Pinville et al. (2011). In the latter, the robot has to navigate, find balls, catch balls, navigate to a basket, and release the ball. This task is considered as one of the most complex in Evolutionary Robotics (Nelson et al., 2009), because of the sequence of behaviors to perform. Ollion and Doncieux (2011) extended this task by adding a switch which opens a new room with new balls to collect. At first glance, this task seems to be difficult to solve using a purely reactive system and therefore *representation-hungry*. However controllers without any internal neuron, with weights fine-tuned by evolution can obtain very good performances (See Subsection 3.4.3).

3.2.4 Conclusion

This short review of the study of internal representation in ER shows the three following points:

- A large number of tasks — even complex ones — are not *representation-hungry*, and therefore do not provide a good context for the study of the emergence of internal representations.
- The emergence of working memory has been studied by several researchers — working memory is a type of internal representation.
- Most of the studies rely on ad hoc and fixed structures: the emergence of representations in a network structure designed by evolution is therefore not studied.

The following section aims at characterizing more precise representations through the definition of tests. Most of the protocols mentioned in this section do not fit the study of these representations — which may be more complex that a one bit memory involved in most delayed-response tasks. A new protocol for studying the emergence of these representations will be defined in Chapter 5. The choice of this protocol is discussed in section 5.2.

3.3 Testing Characteristics of Internal Representations

3.3.1 Measuring Representations in the literature

Due to the broad definitions of internal representations and their various forms in neural networks, little quantification of internal representations in neural network have been made. Some researchers count the number of internal neurons to measure representation and intelligence (McNally et al., 2012). However, this technique does not take into account the connectivity of these neurons, their activity, their dynamics, or the information they represent.

Quantification based on Information Theory A more sophisticated quantification of internal representations was defined by Marstaller et al. (2012). The authors give a formal definition of representations based on information theory. They define the representation R as "the shared entropy between environment states, but *given* the sensor states, that is conditioned by the sensors." This means that R is a measure of the correlation between environment states and internal states, after the removal of direct correlation between sensor states and internal states. This can be formalized in the following way:

$$R = I(E : M) - I(E : M : S) \tag{3.1}$$

where $I(X : Y)$ is the shared entropy of X and Y, E is the environment states, M the internal states, and S the sensor states. The term $I(E : M)$ represents the information of the environment stored in the internal states. The term $I(E : M : S)$ represents the part of information of the environment stored in the internal states that is directly seen by the sensors (further details about the computation of R is found in Marstaller et al. (2012)). In that way, a quantitative value of information (R, measured in bits) *represented* in the network is defined. This information measure was used in several works such as Edlund et al. (2011); Marstaller et al. (2012). While this definition is very generic because of its mathematical grounding, the application to evolutionary robotics is not straightforward, and brings several problems in the cases studied in this thesis:

- It is assumed that all states, internal, or external (possible states of the environment) are discrete. For a continuous task, this would be unfeasible unless a very coarse discretization of states is performed.
- This definition is not compatible with structures such as maps which can represent information in a different way from how single neurons do.
- This definition requires to *a priori* define which discrete states of the world E can be represented by the internal structure.

3.3.2 Properties of Internal Representations

Instead of quantifying directly the level of representation, the following approach focuses on the *functional* properties of the Internal Representations. Based on the reviews of Internal Representations in Cognitive Neuroscience (Section 3.1.3), the following properties can be stressed out:

- **Noise resistance**: The behavior of a noise resistant model/controller is not too affected by noise. This property is useful in biological models, because the input of the networks are often noisy.
- **Occlusion resistance**: The behavior of an occlusion resistant model/controller is not too affected by occlusions in its inputs. Occlusions correspond to the temporary loss of information in an input.

- **Input dependence**: The behavior of the model/controller is affected by the inputs: it is not merely a closed internal dynamic independent of inputs.
- Distractor resistance: The behavior of an distractor resistant model is not too affected by distractors in its inputs. A Distractor corresponds to a temporary competing information in an input. Distractors resistance is only found in models with competing inputs (Action selection and Attention Focus). This property is then too specific to be found in ER studies, and will not be used to study the reviewed models.

Table 3.1 reveals which of the three selected properties can be associated with the reviewed models and controllers.

For Computational Neuroscience models, it is notable that many of the representations are noise resistant and depend on inputs. Several of these models also include a resistance to occlusions. This could be explained by the fact that biological models are built in noisy environments, in which the inputs are not always reliable (hence the occlusions).

For Evolutionary Robotics Tasks, much less emphasis is made on the resistance to noise. The resistance to occlusions can be linked with Memory Tasks: In order to maintain an input information (such as a the stimulus in the Road-Sign Problem) even when this information is lost, one way is to memorize this information.

Summary of reviews Three important properties are focused on: the capacity to be noise resistant, to resist to occlusions, and to have a dependence to the inputs.

3.3.3 Quantitative tests

The Internal Representations defined in this thesis rely on the functional properties of internal nodes of a neural network (a node corresponds to a single neuron or to a map of neurons). After defining several quantitative properties of internal nodes, the aim is to find a qualitative protocol to test whether or not representations are present in an artificial neural network. The main idea is to isolate a node (neuron or group of neuron), and study its activity given different inputs, and determine which category of representation it belongs to, if any (**N** representation or **M** representations, explained later in this section). Figure 3.5 displays the overall procedure.

For each node, the three properties previously highlighted can be defined in relation with the variation of activity of the node when the network is subjected to different inputs. They are defined as follows:

Definition 11 (Noise Robustness) *The noise robustness of a node is the variation of its activity when the inputs are subject to noise.*

Model	N.R.	O.R.	I.D.
Computational Neuroscience Model			
Working Memory (Baddeley, 1992)		x	x
Attention Focus (Rougier and Vitay, 2006)	x	x	x
Action Selection (Redgrave et al., 1999)	x		x
Place Cells (O'Keefe and Nadel, 1978)	x	x	x
Head Direction Cells (Taube et al., 1990)	x	x	x
Grid Cells (Hafting et al., 2005)	x		x
Evolutionary Robotics Task			
Obstacle avoidance (Zufferey et al., 2002)			x
Object Pushing (Hornby et al., 2000)			x
Ball Collecting (Mouret and Doncieux, 2012a)			x
Biped Walking Gaits (Reil and Husbands, 2002)	x		
Homing (Vickerstaff and Di Paolo, 2005a)			x
Minimally Cognitive Behavior (Beer, 2003)			x
Delayed-response Task (Ziemke and Thieme, 2002)	x		x
Memory Task (Capi and Doya, 2005)	x		x
predictive models (Gigliotta et al., 2011)	x	x	x

Table 3.1: Properties of the representations included in CNS models previously reviewed, and Properties required in Evolutionary Tasks previously reviewed. N.R. stands for Noise Resistance, O.R. Occlusion Resistance, and I.D. for Input Dependence.

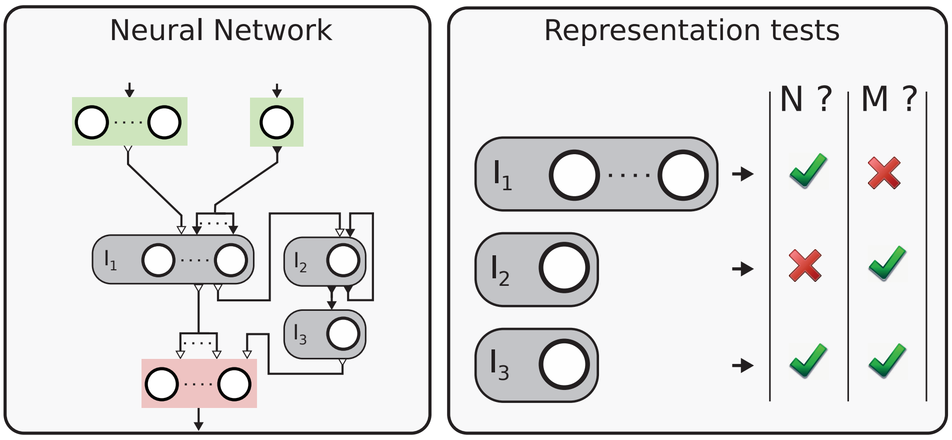


Figure 3.5: Overview of the representation test protocol. (Left) a neural network with different internal nodes (here neurons or maps of neurons, in dark grey) is simulated over different inputs. (Right) The internal nodes are tested for **N** representations and **M** representations. See text for details.

Definition 12 (Occlusion Resistance) *The occlusion resistance of a node is the variation of its activity when a subset of the inputs are muted.*

Definition 13 (Input Dependence) *The input dependence of a node is the variation of its activity when subject to different sets of inputs.*

The Input Dependence property differentiates internal structures that are affected by inputs, and internal structures that are constant or only rely on internal dynamics. Input dependence is different from reactivity to inputs, which is an *immediate* response to inputs.

Each of these definitions goes along with a different quantitative test. Let $\mathbf{X} = \{x\}$ be the set of all internal nodes x (neurons or maps of neurons). The following paragraphs define quantitative descriptors which are applied to each node x. The principle is to quantify the impact of different inputs on the activity of x, over different, *predefined* inputs.

Noise resistance Let S_n be a set of N_n scenarios within the same environment but with different noise levels on inputs. The first (index 0) scenario has no noise. $N(x)$ is the noise impact on x over the different scenarios:

$$N(x) = \frac{\sum_{i \neq 0 \in S_n} d_a(x_i, x_j)}{D_{N_n}}$$

where $d_a(x_i, x_j)$ represents the difference of neural activity of node x_i and node x_j, and the D_{N_n} is a normalization factor.

Occlusion resistance Let S_o be a set of N_o scenarios within the same environment but different occlusions on inputs. In the following work, occlusions are usually introduced by muting the input after several time-steps. The first (index 0) element has no occlusion. $O(x)$ is the occlusion impact on x over the different scenarios:

$$O(x) = \frac{\sum_{i \neq 0 \in S_o} d_a(x_i, x_j)}{D_{N_o}}$$

where $d_a(x_i, x_j)$ represents the difference of neural activity of node x_i and node x_j, and the D_{N_o} is a normalization factor.

Dependence to inputs and situations Let S_d be a set of N_d different scenarios within different environments (and thus with different inputs). $D(x)$ is the dependence of x over the different scenarios:

$$D(x) = \frac{\sum_{i \neq j \in S_d} d_a(x_i, x_j)}{D_{N_d}}$$

where $d_a(x_i, x_j)$ represents the difference of neural activity of node x_i and node x_j, and the D_{N_d} is a normalization factor.

In all definitions, the difference of activity is computed depending on the neural structure, and the node. For a single neuron, this usually corresponds to the following summation over time:

$$d_a(x_i, x_j) = \int_t |a^t(x_i) - a^t(x_j)|$$

where $a^t(x)$ is the activity of neuron x at time t. For maps of neurons, some neurons may not always carry information. This is the case in population coding in which the important information decoded from the map corresponds to the position of the peak of activity in the map (Salinas and Abbott, 1994; Deneve et al., 1999). In the experiments involving that case, the activity difference will be described explicitly.

The resulting value of these three quantitative descriptors depend on the definitions of the sets of scenarios S_n, S_o, and S_d. Unfortunately there is no generic way to define these sets without constraining the reach of this method. However, given a reference input s_0 and s_1 (without noise nor occlusions), one can artificially build the following scenarios:

- For noisy scenarios: S_n is a set including s_0 with Gaussian noise artificially added to the inputs.
- For occlusions scenarios: S_o is a set including s_0 with an input muted after a few time-steps.
- For input dependence, S_d includes two scenarios s_0 and s_1 assuming that s_0 and s_1 should result in different behaviors.

For experiment using these tests, the sets of scenarios S_n, S_o and S_d will be explicitly defined.

Summary of computation For each internal node x, three quantitative descriptors are defined: input dependence $D(x)$, occlusion resistance $O(x)$ and noise resistance $N(x)$. They are computed by comparing the internal node activity when the network is subject to different predefined inputs.

3.3.4 Qualitative tests

Finally, qualitative representation tests are defined in the following way:

Definition 14 (N Representation) *A **N** Representation (for Noise robustness) is an internal node x with the following conditions on its quantitative descriptors:*

$$\begin{cases} D(x) > 0 \\ N(x) < T_N \end{cases}$$

Definition 15 (M Representation) *A **M** Representation (for Memory) is an internal node x with the following conditions on its quantitative descriptors:*

$$\begin{cases} D(x) > 0 \\ O(x) < T_O \end{cases}$$

Definition 16 (N+M Representation) *A* ***N+M*** *Representation (for Noise robustness and Memory) is an internal node x with the following conditions on its quantitative descriptors:*

$$\begin{cases} D(x) > 0 \\ N(x) < T_N \\ O(x) < T_O \end{cases}$$

where T_N and T_O are threshold values which can depend on the noise level, the number of scenarios and the nature of the nodes (neurons or maps of neurons). In the following experiments, these thresholds are fixed to 0.02 unless stated otherwise. The reasons behind the choice of N and M representations are:

- **N** Representations: noise resistant representations should not differ too much when noise is added to the input.
- **M** Representations: memory representations should maintain information from the input even when this input disappears. Therefore it should not differ when occlusions are added to the input.
- Each representation must be dependent on the input. This means that for two very different input scenarios, different values should be represented in the internal node. This prevents internal nodes which are constant or only rely on internal dynamics without taking into account the input. It is important to note that this property is very different from being reactive.

Neurons that merely receive copies of the input sensors will be input dependent, but will not satisfy the **N** or **M** other properties, and will therefore not be considered as internal representations.

Throughout this thesis, a Neural Network will be said to have a **N** representation (resp. **M** and **N+M**) if at least one of its internal nodes is an **N** representation (resp. **M** and **N+M**).

3.4 Analysis over Existing Networks

In the previous section two qualitative representation tests (**N** and **M**) were defined. In order to validate this methodology before using it on ER problems, existing networks are tested fpr representations.

- Handcrafted networks used in Evolutionary Robotics, such as feedforward Networks, Fully Recurrent networks, and Elman Networks.
- A visual attention experiment taken from Computational Neuroscience literature. The described function is thought to be cognitive, and is based on an internal 2-dimensional map, tested against the newly defined representation criteria.

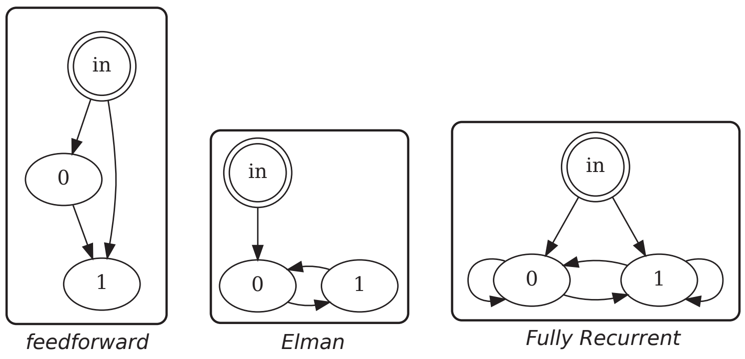


Figure 3.6: Three simplified networks with one input and two internal nodes. From left to right: 1) a feedforward network; 2) an Elman network with only one hidden and context node; 3) a fully connected recurrent network with two internal nodes.

Finally, the method will be tested on an ER experiment, the Hard Ball Collecting task, which is a complex foraging and navigation task. The neural architecture is not fixed, and eventual internal neurons are tested against the newly defined representation criteria. It is not known whether yet representations are present in the resulting networks.

3.4.1 Representations in classic networks

Setup The **N** and **M** representation tests are first validated on classic neural networks used in Evolutionary Robotics. Figure 3.6 displays the three considered minimal networks, with one input and two hidden neurons:

- A feedforward network. Such networks do not have any internal dynamics, and therefore no memory is expected in their internal nodes.
- An Elman network, which has a recurrent loop between its hidden and context nodes. It can display very simple dynamics and hysteresis.
- A fully connected recurrent network, which can have more complex dynamics than the Elman network. The weights of recurrent connections are all positive.

The networks are simulated for 50 time-steps over different scenarios, and the activity of their internal neurons are recorded to compute the 3 descriptors (input dependence, noise impact and occlusion impact). A scenario here consists of the input activity of the "in" neuron over the 50 time-steps. Scenarios sets are defined as follows (see Figure 3.7):

- For input dependence, S_d includes the first (oscillating) and second scenarios (constant) (see Figure 3.7). It is assumed here that such scenarios should produce different internal activity.
- For noisy scenarios: S_n includes the first (oscillating) scenario and the three last scenarios which correspond to the same scenario with uniform noise ($\varepsilon \in$

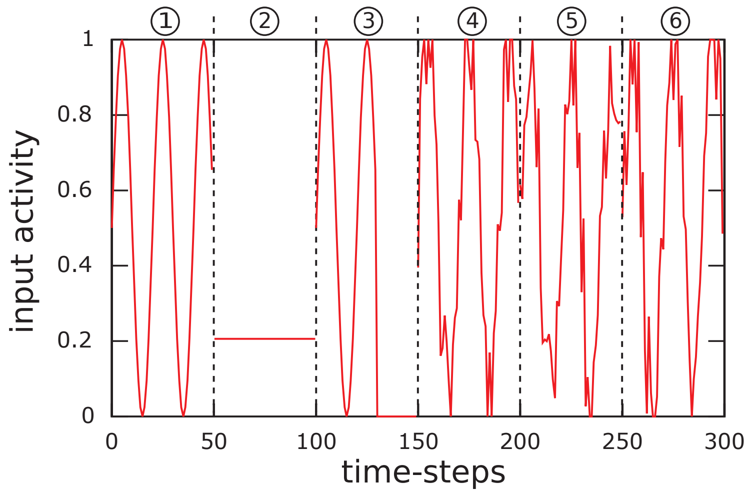


Figure 3.7: 6 Different 50 time-steps inputs (scenarios) used to determine whether Internal Representations are present in the 3 networks. From left to right: 1) basic oscillating input; 2) constant low input; 3) oscillating input, muted after 30 steps; 4,5,6) oscillatory input with added noise (taken from a uniform distribution). The network is reinitialized every 50 steps, which means that any eventual internal information is then reset.

$[-0.1, 0.1]$ is added to the input, different random numbers are chosen for each noisy scenario).

- For occlusions scenarios: S_o includes the first (oscillating) scenario and the third scenario (oscillating, but muted after 30 time-steps).

The comparison of an internal neuron in the different scenarios is a simple summation of the difference of neural activity over the last 10 time-steps:

$$d_a(x_i, x_j) = \sum_{t=40}^{t=50} |a^t(x_i) - a^t(x_j)|$$

Results The representations tests results are displayed in table 3.2. The table also displays the quantitative test values for node 1: $D(1)$, $N(1)$, and $O(1)$.

- As expected intuitively, the feedforward network does not pass any representation test. While the condition to be input dependent is satisfied, the activity of the internal neurons are sensible to noise and completely drop because of the occlusion. The test is valid here, as it reflects the representation intuition on this network.

- The Elman network behaves as a hysteresis memory: When the input is over a certain threshold, the two internal nodes have a large activity. They are then sufficiently activated to maintain their activity even if the input disappears, and will never drop back to lower activity. This behavior is not affected

Params	D(1)	N(1)	O(1)	**N**	**M**
Feedforward	**0.65**	0.058	0.86	no	no
Elman	**0.94**	**0.001**	**0.0**	**yes**	**yes**
Fully Recurrent	0.0	**0.0**	**0.0**	no	no

Table 3.2: Values of quantitative tests for the internal node 1, and representations (present in at least one of the two internal node 0 or 1) for the different networks.

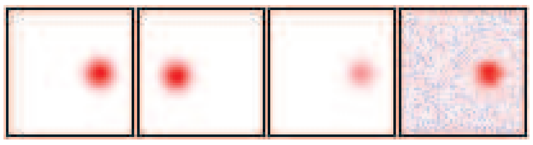


Figure 3.8: Typical scenarios (inputs) snapshots. From left to right: 1) Basic input 2) displaced input 3) faded input 4) noisy input. Reproduced from Quinton (2010b) with permission.

by noise. The newly defined test then displays a **N+M** Representation as expected from the behavior.

- The fully connected recurrent network was designed with large recurrent weights. Therefore it corresponds to a dynamical system which, for any input, quickly converges to a fully saturated state (all internal activity is always maximal). These internal nodes do not carry any information about the input. As the internal neurons are not input dependent, the nodes are not considered as internal representations by the test.

By choosing different parameters for the Elman or the Fully connected recurrent network, other properties could have emerged or disappeared. The three given examples display different behaviors and are correctly categorized by the representation tests.

3.4.2 Visual Attention

Rougier and Vitay (2006) use Continuous Neural Field Theory (CNFT) (Amari, 1977) to model the emergence of attention within a neural population. The experiment was later extended by Quinton (2010b), in an attempt to find the right parameters using evolutionary techniques. The model is composed of two 2-dimensional maps: 1) an input map, which represents a visual input (for instance, several visual cues with noise, See Figure 3.8 for examples); 2) an internal map — the attention focus map — in which only a focus "bubble" should appear. The internal map then "represents" the point of attention.

The best resulting parameters from the experiment by Quinton (2010b) are used to parametrize the Continuous Near Field (CNF). Those parameters determine the lateral connections between the neurons in the internal map. Table 3.3 shows the

Name	Properties
Quinton	Parameters taken from the evolutionary experiments performed by Quinton (2010b).
Param I	Handmade parameters: No lateral inhibition, and very strong self activation.
Param II	Handmade parameters: Strong lateral inhibition, weak self activation.

Table 3.3: Parameters of the lateral connections in the CNF.

Params	**N**	**M**
Quinton	yes	yes
Param I	no	yes
Param II	yes	no

Table 3.4: Representations for the different parameters of the CNF.

different sets of parameters tested for **N** Representations, **M** Representations, and both.

For each parameter set, the network is tested with different scenarios; a scenario corresponding to a set of inputs over time. The sets of scenarios for input dependence S_d, noise dependence S_n and occlusion dependence S_o are then defined in the following way (See Figure 3.8 for illustration of these scenarios): .

- S_d is a set of standard inputs at different locations. These inputs are maintained for 50 time-steps in the neural field. s_0 is the leftmost input on Figure 3.8.
- S_n is composed of s_0 and 9 other inputs corresponding to s_0 with a Gaussian noise added.
- s_0^O is a copy of s_0 for the 25 first time-steps, and a copy of s_0 with lower intensity for the 25 next steps. For the sake of simplicity, S_o is composed here of s_0 and s_0^O with a muted input (the intensity of the input is set to 0%).

The single internal map is tested for representations. Let ${\mathbf{x_i}}^{t=T}$ be the activity of the internal map for scenarios i at time T. The difference of activity is computed at the last time-step ($t = 50$) by normalized euclidean distance:

$$d_a(\mathbf{x_i}, \mathbf{x_j}) = ||{\mathbf{x_i}}^{t=50} - {\mathbf{x_j}}^{t=50}|| \tag{3.2}$$

From the computation of quantitative descriptors, the qualitative representation tests are computed. Thresholds for the computation were $T_N = 0.02$ and $T_O = 0.02$. The results are shown in Table 3.4

Analysis of the results The three sets of parameters resulted in the following internal behaviors:

- With the Quinton parameters, the internal map focuses on the input, discards the noise, and remains activated when the input is lost. The newly defined representation test shows that this internal map is indeed a **N+M** Representation (See Table 3.4).
- With no lateral inhibition (Param I), the activity of the internal map instantly focuses on any input and remains focused. However, any positive activation induced by noise is also focused on, which results in a fully saturated internal map in noisy conditions. In these conditions, the representation found by the newly defined test is a **M** Representation.
- The strong lateral inhibition parameters and weak activation (Param II) made it impossible for the internal map to have any activity when the input is lost. However, any noise is nullified by the lateral inhibition. The representation result show, for this case, a **N** Representation.

For all three cases, the representation test found representations which intuitively correspond to the expected behavior of the internal map.

3.4.3 ER task: Hard Ball-Collecting Task

Experimental Setup The Hard ball-collecting Task is an extension of the ball-collecting Task described by Mouret and Doncieux (2012a). It was used by Ollion and Doncieux (2011) as an experiment to measure the behavioral exploration of the evolutionary process during evolution. It is a complex task requiring many different elementary behaviors to be performed in a sequence. The fitness is simply the normalized number of balls collected by a robot. The robot is evaluated with three different starting positions (see figure 3.9), and its fitness is the average fitness in each of these three simulations. The robot has to navigate toward balls, grab them, navigate towards the basket, and finally release the ball, in order to score and have a fitness point. Alongside the main fitness objective, an helper objective was added which rewards novel behaviors. This novelty multiobjectivization was described by Mouret (2011). The control architecture is a neural network with a topology generated by the evolutionary algorithm using DNN (see 2.3.4.1). Networks are initially feed-forward with no hidden neurons, and neurons can be added through mutations during the evolutionary process. The input neurons of the controller are defined as follows:

- *i0, i1, i2*: 3 distance finding sensors, normalized in $[0, 1]$;
- *i3, i4*: 2 ball sensors (1 if a ball is in the view field of the sensor, 0 otherwise);
- *i5, i6*: 2 basket sensors (1 if the basket is in the view field of the sensor, 0 otherwise);
- *i7, i8*: 2 switch detectors (1 if the switch is in the view field of the sensor, 0 otherwise);

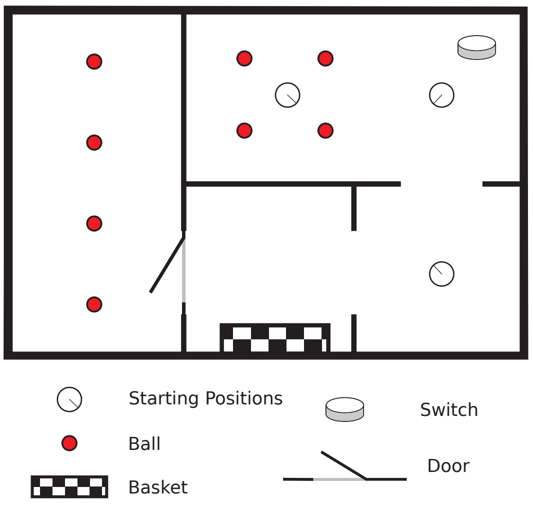


Figure 3.9: Experimental setup for the Hard ball-collecting Task. The robot is simulated three times with different starting positions, and has to collect balls and release them into the basket. The switch has to be pressed once per simulation in order to open the door to access to the left-most room.

- *i9, i10*: 2 bumpers (1 if it touches a wall, 0 otherwise);
- *i11*: 1 ball carrying sensor (1 if the robot currently carries a ball, 0 otherwise).

The outputs of the neural network control the robot actions as follows:

- *o0, o1*: 2 wheel motor commands: controls the speed of left and right robot wheels;
- *o2*: 1 "action" motor: if greater than 0.5, pick up a ball if possible, else throw the carried ball if any; if no ball is carried and greater than 0.5, press the switch if possible.

Resulting Networks The evolution of controllers for the Hard ball-collecting task is very challenging, and the 30 runs did not converge to maximal fitness at all (average fitness 0.5 — half of the balls were collected in average). Another batch of runs was evolved, using novelty only (no fitness score based on the number of ball collected). After 4000 generations of evolution, the resulting networks (corresponding to different number of balls collected) had the following properties:

- For Objective + Novelty, about one third of the networks (9 out of 30) had no hidden neurons at all (see figure 3.10).
- Other networks had between 1 and 5 internal neurons, with an average of 2.4 (see figure 3.11), which were tested for representations (see paragraph 3.4.3).
- When using Novelty only, the best controller with regards to the number of balls collected is considered after 4000 generations. The resulting controllers

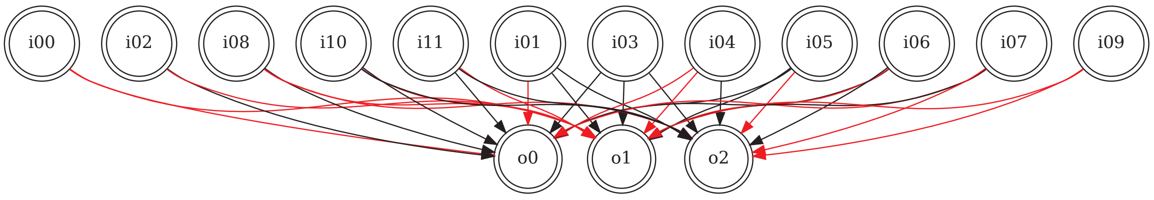


Figure 3.10: Resulting neural net for the Hard ball-collecting Task, with no hidden neurons. Double circled nodes represent input and output neurons. Weights and neuron biases are not represented. The fitness of this controller is 0.5

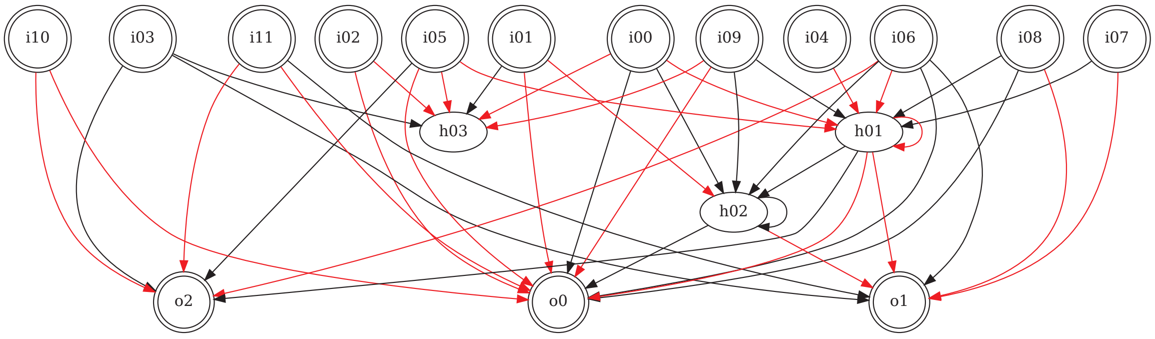


Figure 3.11: Resulting neural net for the Hard ball-collecting Task, with three hidden neurons (single circled nodes). The neuron $h02$ is a **M** Representation. The fitness of this controller is 0.7.

in all 30 runs had at least one hidden neuron, and up to 11, with an average of 4.7.

Representations in Ball Collecting Task Like in the previous experiment, in order to test for **N** Representations and **M** Representations in a controller, sets of scenarios for input dependence S_d, noise dependence S_n and occlusion dependence S_o have to be defined (see 3.3.3). A scenario corresponds to a disembodied set of inputs over time which is applied to the controller of the robot. A simulation of the robot in the arena with the given controller is performed. The inputs of the robot are recorded; each 50 time-steps gives a scenario. s_0 represents the recorded inputs for the 50 first time-steps, s_1 the 50 next, and so on.

We then define these sets in the following way:

- The set of all scenarios s_0, s_1, etc. is S_d. A total of 6 scenarios were used.
- s_0^N is the scenario s_0 with 20% Gaussian noise added to the inputs. S_n is a set composed of: 1) s_0; 2) 9 of these randomly modified scenarios s_0^N.

Fitness function	# of runs	# of networks with Internal Neurons	**N**	**M**	**N+M**
Standard	30	21	0	1	0
Novelty Only	30	30	0	0	0

Table 3.5: Number of Neural Networks with at least one internal neuron, or representation.

- s_0^O is the scenario s_0 with inputs muted after 25 time-steps (all inputs are then equal to 0). S_o is a set composed of: 1) s_0 2) s_0^O.

The quantitative descriptors are computed for each internal neuron, and the difference of activity is computed between time-step $t = 40$ and time-step $t = 50$. Thresholds for the qualitative computation were $T_N = 0.02$ and $T_O = 0.02$. The results are shown in Table 3.5.

These sets of scenarios are used to determine whether internal neurons are **N** Representations, **M** Representations or **N+M** Representations. Results can be seen in 3.5

Analysis of the results Through the evolution of the Hard ball-collecting task, internal representations do not seem to emerge. In 9 runs out of 30, the resulting neural networks do not have a single hidden neuron. Figure 3.10 shows an example of a rather successful feed-forward neural network for this task (the robot is able to collect half of the balls). The representations tests show only 1 resulting controller including an internal **M** Representation. This controller is displayed on figure 3.11. In-depth handmade analysis show that the neuron $h02$ has a positive feedback and a bias that generates a memory dynamic: when the input $i6$ (basket sensor) is active long enough, the neuron enters an hysteresis state and its activity is maximal even when the input $i6$ disappears.

The other internal neurons, are not considered as representations by the newly defined protocol. A further study shows that most of the neurons show the following behavior:

- Many neurons are simply copies or summation of inputs. The sensory information is fed to them. Therefore those neurons cannot have any memory, and are usually affected by noise.

- Many neurons quickly converge towards a constant value (usually minimal or maximal value possible for the neuron). These neurons still provide a few time-steps of dynamics that is used by the evolution process to generate more complex behaviors, but these neurons are not input dependent.

- Some interconnected internal neurons oscillate and are merely affected by any input.

The networks evolved in the Hard ball-collecting task seem to rely on reactive behaviors, and in some cases have some internal dynamics uncorrelated with inputs.

Intuitively, these networks do not have any internal representation. This intuition is reflected by the newly defined representation test protocol, which indicates only one representation out of 30 runs. Interestingly the Novelty only setup generated controllers that had more internal neurons in average, but those neurons only generate dynamics which enables more complex dynamics. Most of the internal neurons also quickly converge to a constant value.

3.4.4 Conclusion

Three internal representation tests were defined, which test whether an internal representation with a given propriety is present in the neural network:

- **N** Representations (robust to noise)
- **M** Representations (with memory)
- **N+M** Representations (with both properties)

These tests were validated on three experiments: 1) Simple classic neural networks, 2) a Computational Neuroscience experiment in which the internal map can be considered as an internal representation 3) a Hard ball-collecting task, in which the controllers rely on reactive behavior and simple dynamics, without representations. In all three cases, the test protocol correctly indicated whether representation were present.

Chapter 4

Behavioral Consistency Method

This chapter is composed of two parts:

Goals

1. Promoting the emergence of **M** representation with a *memory helper objective.* [a]

2. Definition of the Behavioral Consistency Method — a generalization of the previous helper objective. [b]

Method

1. Application of the **M** representation test, as a *memory helper objective* in a T-maze ER experiment

2.
   - *Behavioral Consistency* — a new method for building fitness objectives based on the consistency of behavior over different scenarios.
   - The new method is applied to two different setups involving the evolution of cognitive abilities — action selection and attention focus.

Results

1. Successful emergence of memory units in networks, favored by the new *helper objective.* Resulting networks generalize better.

2. The evolution of action selection and attention focus is successful using *Behavioral Consistency*, while using little knowledge about the desired behavior of the system.

[a]Work realized with T. Pinville (Ollion et al., 2012).
[b]Work realized with S. Doncieux (Ollion and Doncieux, 2012).

Contents

4.1 Introduction 74
4.2 Working Memory Experiment 74
4.2.1 T-Maze navigation task 75
4.2.2 Methods 76

4.2.3 Results 78
4.2.4 Conclusion 82
4.3 Behavioral Consistency Method 85
4.3.1 Related Work 87
4.3.2 Behavioral Consistency Method 87
4.4 Attention Focus 90
4.4.1 Experimental Setup 90
4.4.2 Results 93
4.5 Action Selection 94
4.5.1 Experimental setup 94
4.5.2 Results 96
4.6 Conclusion 97

4.1 Introduction

In the previous chapter, internal representation tests in neural networks were defined: **M** (for *memory*) and **N** representations (for *noise resistance*). In order to obtain such representations, these tests can directly be used in the selection process: individuals that pass the tests can be given a higher chance of survival by the selection process. These selection process can be added to a MOEA as a new *helper objective*. The new helper objective explicitly promotes internal representations in a ER task. The first part of this chapter covers the use of such an objective in a T-maze problem. This work was realized with Tony Pinville (Ollion et al., 2012). Furthermore

The method is useful for the synthesis of specific internal representation, however it can have limitations (the uses and limitations of this approach will be discussed in part 4.2.4.1). Furthermore, this approach has a lower interest for the study of the conditions for the emergence of representations, as these representations are explicitly promoted.

In a second part, the method used to build the helper objective is generalized to the *behavioral consistency method* (BC method). This method, based on behaviors, adds less bias and can be applied to many more problems. The method is applied to two problems from computer neuroscience. This work is adapted from (Ollion and Doncieux, 2012).

4.2 Working Memory Experiment

In this section a selective pressure inspired by the **M** representation test is applied to an ER experiment: A T-maze navigation task involving working memory (Ollion et al., 2012; Pinville et al., 2011).

4.2.1 T-Maze navigation task

The task is an extension of the "road-sign problem" (Ziemke and Thieme, 2002; Rylatt and Czarnecki, 2000): an agent starts off at the bottom of a T-shaped maze, encounters an instruction stimulus (e.g. a light) while moving along a corridor and, when it reaches the junction, it has to turn left or right, depending on which stimulus has been encountered (Figure 4.1).

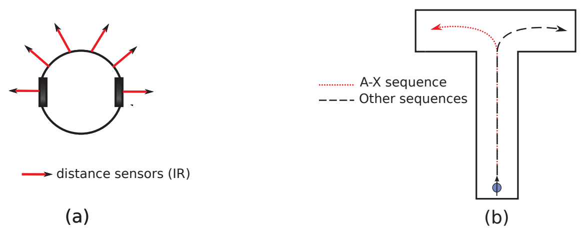


Figure 4.1: (a) Simulated mobile robot used for the T-maze task. The robot has four additional sensors, one for each letter. (b) Map employed for this task.

In the initial setup, controllers that simply follow the right or left wall after the signal can solve the task while not having any memory (Ziemke and Thieme, 2002). To make this task more cognitive, in our experiment the instruction stimulus is a combination of four stimuli (A, B, X, Y) following the same rule as in the AX-CPT working memory test (Braver et al., 1995; Pinville and Doncieux, 2010). This task consists of a context stimulus (A or B), followed by a second stimulus (X or Y) after some delay. The agent must turn to the left when the stimulus A is followed by the stimulus X, and to the right otherwise (for AY, BX, BY).

Here, the agent is a simulated two-wheeled robot receiving sensory inputs from 6 infrared distance sensors and four letter sensors, one sensor for each letter A, B, X, Y, which receives 1 if the letter is presented, 0 otherwise. The robot controls its speed through two output units corresponding to its left and right motors. The agent is evaluated on each letter sequence (A followed by X, AY, BX, BY). The fitness increases by 1 if it turns to the correct side for the sequences AY, BX, BY and by 3 for the sequence AX, for a maximal value of 6. This fitness will be referred as "Goal oriented fitness".

Both motors are disabled during the presentation of the letters. The whole task lasts 350 time-steps and takes place as follows with t the number of elapsed time-steps:

- $0 < t < 50$: presentation of the first letter (A/B);
- $50 \leq t < 100$: delay, all the sensors are set to 0;
- $100 \leq t < 150$: presentation of the second letter (X/Y);
- $150 \leq t \leq 350$: the robot can move and must reach the correct side of the T-maze.

In order to avoid over-fitting to a specific initial configuration of the robot, 12 different contexts have been defined for each possible letter sequence. A context is described by an initial starting position (4 different positions) and an initial starting angle (3 different angles).

4.2.1.1 Neural network encoding

The agent is controlled by a neural network whose structure and parameters are evolved with DNN encoding (See section 2.3.4.1 for further details). A LPDS-based (locally Projected Dynamic System) neuron model (Girard et al., 2008) is used to simulate the neurons with an output in $[-1, 1]$ (Neuron models are detailed in section 2.3.1). Both DNN and LPDS were already been used conjointly in (Pinville et al., 2011).

4.2.2 Methods

The multi-objective approach has an interesting feature: adding a selection pressure can be done simply by adding as a separate objective with no need to tune any new parameter for the relative importance of each objective to be optimized. This means that all objectives are considered equally important and multi-objective evolutionary algorithms aim at finding the best trade-off solutions relative to them (Deb, 2001). The two selection pressures studied here are then defined as separate objectives to be optimized with a multi-objective evolutionary algorithm. Such objectives do not describe the goal to be reached, but aim at enhancing the evolutionary search, they are then *helper objectives*. This approach is called multiobjectivization (Knowles et al., 2001; Mouret, 2011). Further information about multiobjectivization and helper objectives is located in section 2.2.3.1.

In the following, two helper objectives have been considered:

- a behavioral diversity, as defined by Mouret and Doncieux (2012b);
- a memory helper objective, inspired by the **M** representation test, as introduced in the following section.

4.2.2.1 Behavioral diversity

The behavioral diversity (See section 2.2.3.1 for more information about diversity) associated with individual x in a population of N individuals is computed in the following way:

$$div(x) = \frac{1}{N-1} \sum_{y \neq x} d_b(x, y)$$

The behavioral distance $d_b(x, y)$ is the euclidean distance between the positions of the two robots x and y at last time-step $t = 350$.

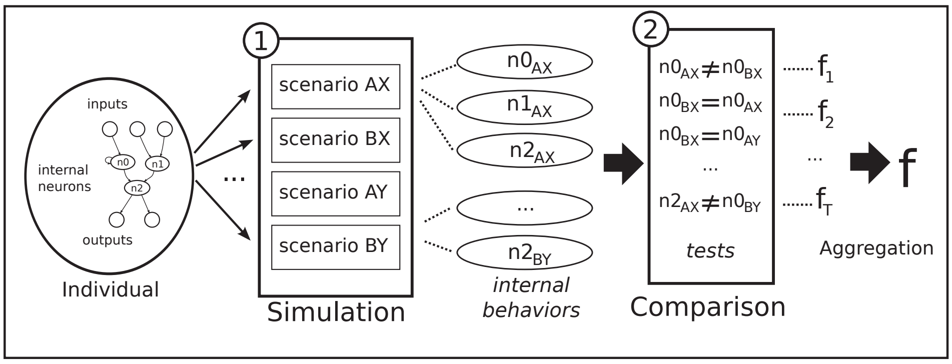


Figure 4.2: Details of the evaluation of the memory helper objective. 1) An individual (here a neural network with internal neurons $n_1, n_2, ...$) is simulated in the 4 scenarios AX, BX, AY and BY. During this simulation, the behavior of each internal neuron is stored. 2) The internal behaviors are compared and checked for coherence (for instance, AX should have a different behavior than BX), resulting in a partial fitness value f_i. Then, the partial fitness values are aggregated into the final evaluation f.

4.2.2.2 Memory Helper Objective

In an experiment using the same T-maze protocol, Pinville (2013) show that hardly any internal memory emerge in the evolved controllers. By using helper objectives, the emergence of memory can significantly increase. The Memory Helper Objective is an objective designed to explicitly promote controllers in which at least a neuron shows signs of a memory. It is inspired by the **M** representation test: An individual has N internal neurons — N may vary from individuals to individuals and during evolution. The helper objective relies on testing each of these internal neurons to see if they meet the memory requirements. These tests rely on the comparison of the behavior of internal neurons when the network is subjected to different scenarios (inputs). The global method for computing the Memory Helper Objective score is displayed on Figure 4.2. The exact computation of the memory score relies on the following steps:

In each context, the individual is simulated over the 4 different possible inputs (referred later as "scenarios") AX, BX, AY, and BY.

For each scenario s, let $b^i_s(t)$ be the output of the i-th internal neuron in scenario s at time-step t, after the presentation of letters ($t > 150$). The goal of the scenario based objective is to reward individuals that obey the following rules:

$$\forall s \in S,\, b^i_{AX}(t) \neq b^i_s(t)$$

$$\forall s, s' \in S,\, b^i_s(t) = b^i_{s'}(t)$$

With $S = \{BX, AY, BY\}$. In other words, the behavior of an internal representation should be the same if the inputs are AY, BX, BY, and different if the input is

AX. The behavior is computed after the presentation of letters, which means that the input letters are no longer active. The existence of a difference between the scenarios should reflect the emergence of a memory.

For each internal neuron i two partial fitness terms f_1^i and f_2^i are computed, they measure how well the internal neuron respects the two previous rules:

$$f_1^i = \frac{1}{|S|}\sum_{s\in S}\frac{1}{200}\sum_{t=150}^{350}\frac{|b_{AX}^i(t) - b_s^i(t)|}{2}$$

$$f_2^i = 1 - \Big[\frac{1}{|S|^2 - |S|}\sum_{s,s'\in S, s\neq s'}\frac{1}{200}\sum_{t=150}^{350}\frac{|b_s^i(t) - b_{s'}^i(t)|}{2}\Big]$$

Then, the fitness of each internal neuron is computed as follows:

$$f^i = f_1^i + f_2^i$$

As the goal of this experiment is to select individuals that have *at least* one internal neuron that represents the information, the final fitness is computed as the maximum of all internal fitnesses f^i:

$$f = \max_{0\leq i<N} f^i$$

The fitnesses f compare the four letter sequences evaluated in the same context. The overall scenario-based fitness corresponds to the average of the 12 fitnesses thus defined (one for each context). Therefore there are $12 \times 4 = 48$ total simulations for each individual of each generation. As the computation of the goal-oriented fitness already requires the simulation of these scenarios, no additional evaluation is required.

4.2.2.3 Setups summary

In this section, the different objectives are referred as follows:

- **G**: Goal-oriented objective;
- **D**: Diversity objective;
- **S**: Scenario-based *memory helper* objective;

To test the influence of each objective, experiments are launched with various combinations of objectives as shown in Table 4.1. The multi-objective evolutionary algorithm is NSGA-II (Deb, 2001) and each of these setups is run 30 times.

4.2.3 Results

Figure 4.3 depicts box-plots for the goal-oriented fitness results on each different setups. The red line represents the median value, the box extends from the lower to upper quartile values of the data. The whiskers extend to the most extreme data points not considered outliers, these ones are plotted individually.

Table 4.2 displays the corresponding p-values using Mann-Whitney statistical test. Figure 4.4 shows the median fitness values for the 4 setups.

Table 4.1: Summary of different setups used

	Setup	Description
1	**G**	Goal-oriented
2	**G + D**	Goal-oriented + Diversity
3	**G + S**	Goal-oriented + Scenario-based
4	**G + D + S**	Goal-oriented + Diversity + Scenario-based

4.2.3.1 Diversity Effect

Figure 4.3 shows that a fitness rewarding the completion of the task (**G**) only has poor results. This is confirmed by Figure 4.4 in which one can see that a fitness plateau is quickly reached. The fitness plateau is at $f = 0.5$, which corresponds to controllers that always go to the same side of the maze. Adding a diversity objective (D) significantly increases performance and delays fitness plateaus. This result is compatible with our hypothesis that the evolution of a memory is a deceptive problem and shows that selective pressures have indeed a significant impact on the success rate.

4.2.3.2 Scenario-Based Objective Effect

The use of the Scenario-based Objective also increases the performance significantly, to the same extent as the diversity objective. There is no statistical difference between $G + D$ and $G + S$ setups.

Using both objectives further increases performance, and as no fitness plateau was reached during the 2000 generations (Figure 4.4). One can then expect the fitness to be even better with more generations.

Table 4.2: P-values between each setup on goal-oriented fitness value

	G + D + S	**G + S**	**G + D**	**G**
G + D + S	x	**0.04013**	0.0639	**<1e-05**
G + S	**0.04013**	x	0.18504	**0.00409**
G + D	0.0639	0.18504	x	**<1e-05**
G	**<1e-05**	**0.00409**	**<1e-05**	x

The two next sections present a more in depth study of results: the resulting networks are tested for a reliable memory and generalization ability.

4.2.3.3 Memory computation

After the optimization phase, the internal nodes are tested for internal representations with memory. This computation is very similar to the **M** representation test defined in the previous chapter. The difference is that the neurons are tested for a *reliable memory*: the memory should not be too specific to the delays presented

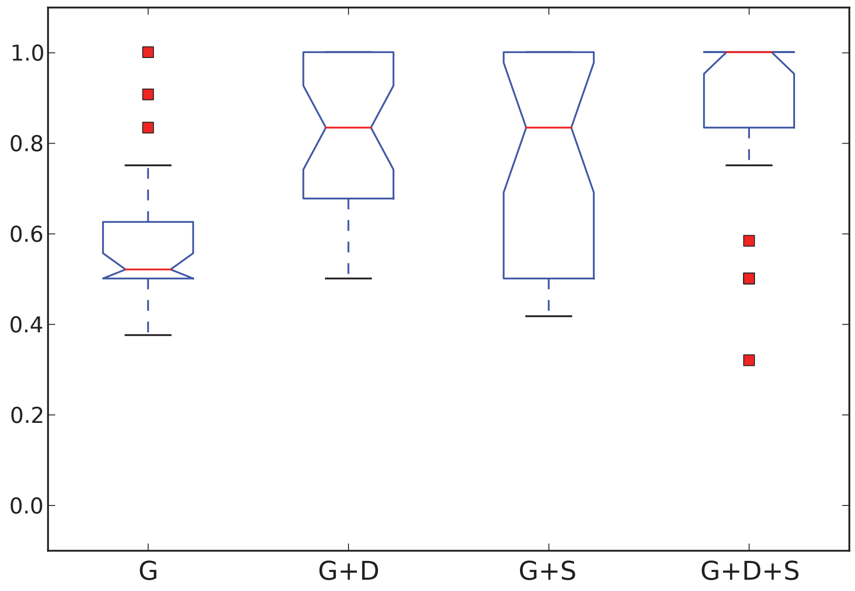


Figure 4.3: Box-plots for the goal-oriented fitness results on each different setups

during evolution. A network is considered to exhibit this *reliable memory* if at least one internal neuron respects the two following points:

- After presentation of the letters, the neuron has a different output for AX scenarios and for AY, BX, BY scenarios.
- The memory is not affected by the duration of the presentation of the letters. While during evolution the duration of the presentation was 50 time-steps for each letter, the activity of the network is tested —after evolutionary process— with a duration of 400 time-steps. This is aimed to detect networks that rely on complex dynamics to have different activities after exactly 50 time-steps, but would not work with a different duration.

In Figure 4.5, the black histogram displays the percentage of runs (out of the 30 runs per setup) in which the best individual achieves a reliable memory. While the diversity objective slightly increases memory, the Scenario-based objective significantly affects memory. Interestingly using both helper objectives results in less memory than using the Scenario-objective alone: this is likely due to a compromise between the diversity objective **D** and the memory helper objective **S**.

4.2.3.4 Generalization ability

Another important aspect studied here is the generalization ability, which is the ability to perform well in contexts unseen during evolution. *During evolution*, the robot is tested in 12 different contexts for each letter sequence, and maximal fitness is achieved only if the individual manages to solve the problem in all the contexts.

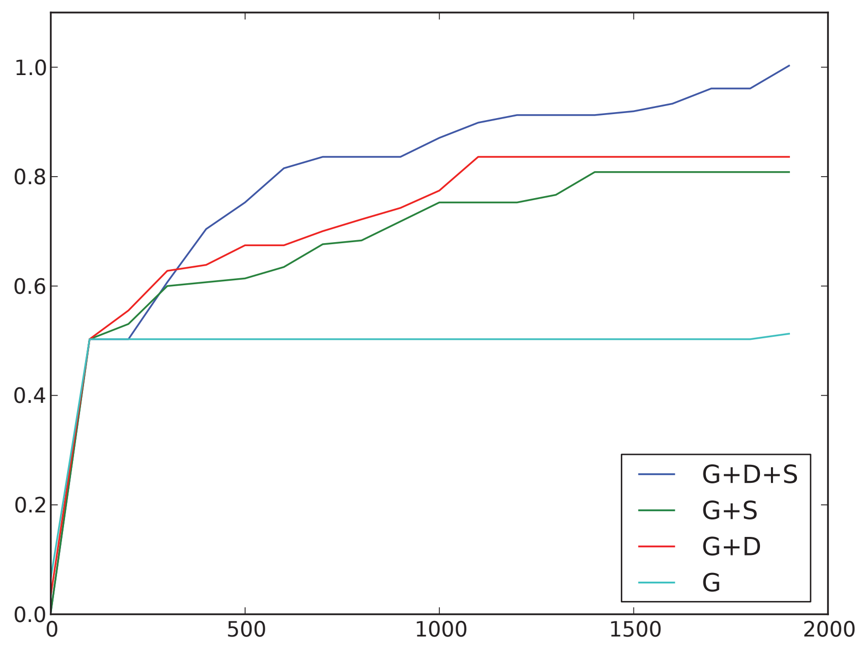


Figure 4.4: Evolution of fitness objective (median value of all 30 runs).

After evolution, the best controllers are tested in 180 previously unseen contexts. The 180 new contexts include different map sizes, starting positions, and starting orientations of the robot. A controller is considered to generalize well if it can still perform the task in at least 60 of these new contexts. Figure 4.5 shows the proportion of runs with individuals which generalize. Figure 4.6 details the number of contexts in which these individuals generalize. There is a very significant increase of generalization when using the helper objective, and even more when using both objectives. Table 4.3 displays the corresponding p-values.

Table 4.3: P-values between each setup on the generalization ability

	G + D + S	**G + S**	**G + D**	**G**
G + D + S	x	**0.03241**	**0.00364**	**1e-05**
G + S	**0.03241**	x	0.08408	**9e-05**
G + D	**0.00364**	0.08408	x	**0.0094**
G	**1e-05**	**9e-05**	**0.0094**	x

4.2.3.5 Analysis of the resulting networks

Two resulting networks, shown in figures 4.7 and 4.9, are analyzed in this section. They both achieve maximal fitness, but only the second one exhibits reliable memory and generalization ability. Blue neurons have a different neural activity for AX sequence during the memory test. Figure 4.8 and 4.10 show the corresponding internal behavior of the neurons during the test for networks in figure 4.7 and 4.9.

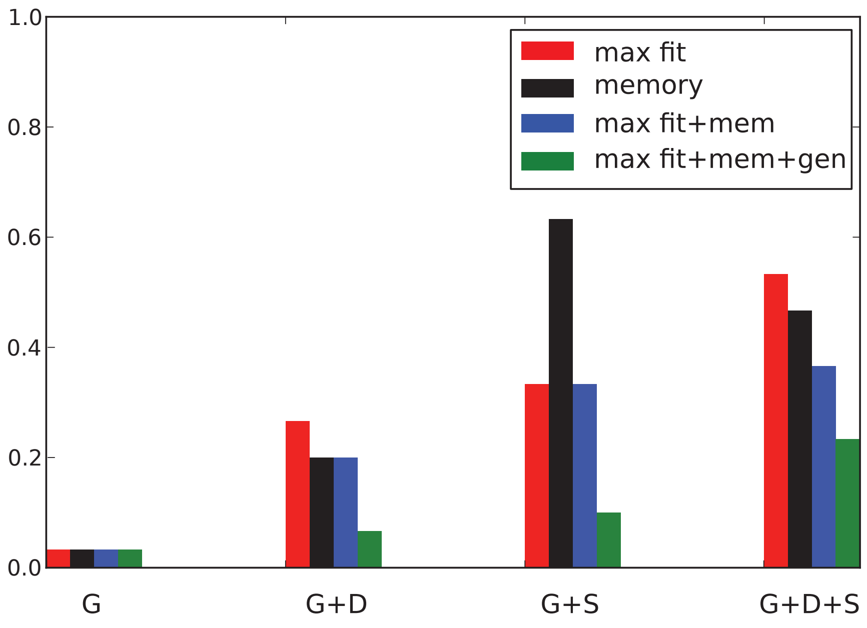


Figure 4.5: Proportion of runs matching different criteria: (1) achieving maximal fitness (2) having memory (3) having both (4) having both and generalizing to 60 of the 180 extra contexts.

The first presentation of letters lasts from 0 to 400, the delay from 400 to 800, the second letter from 800 to 1200. In order to distinguish AX and BX sequences, the network must remember A or B stimulus during the delay period.

Figure 4.8 shows that the network depicted on figure 4.7 is not able to retain A or B stimulus when the delay interval is extended. At time-step $t = 800$, the internal behavior of the neurons are similar for the 4 sequences. At the end of the presentation of letters, the neural network cannot therefore distinguish AX and BX sequences. In figure 4.10, there are two different neurons, neurons 0 and 3, able to memorize stimulus B even if the delay interval is extended. In this case, at the end of the presentation of letters, the internal behavior of the neurons for AX sequence is different from that of the other sequences.

4.2.4 Conclusion

These experiments confirm that the emergence of memory is a challenging problem. With the present encoding, structures with memory require several mutations to

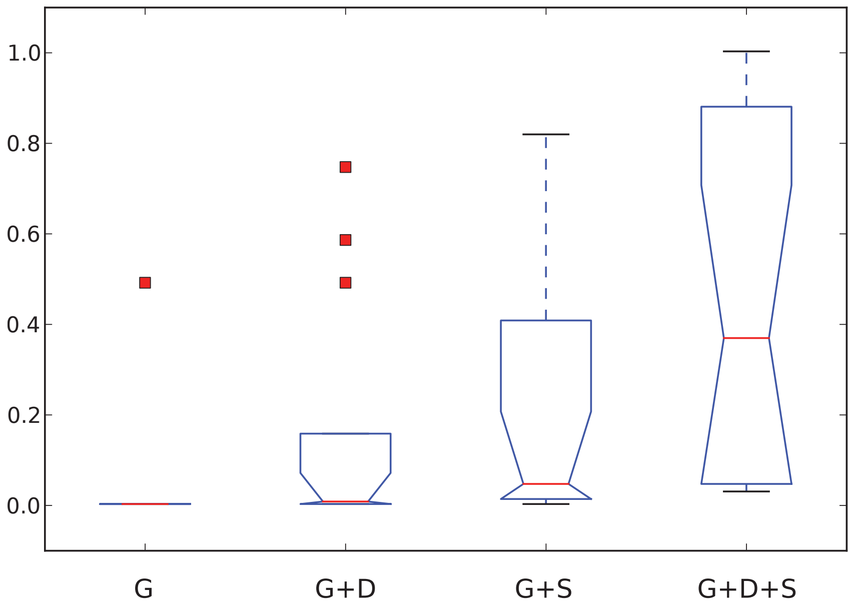


Figure 4.6: Generalization ability of the 15 best runs for the 4 different setups. The value corresponds to the proportion of contexts in which the agent solves the task.

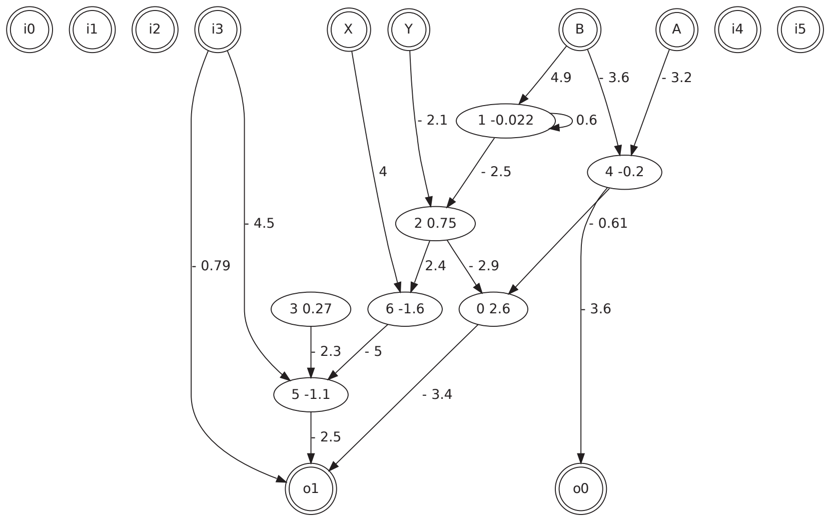


Figure 4.7: Resulting neural network with maximal fitness, but no memory nor generalization

appear; they are much more likely to appear under specially-designed selective pressures. The helper objectives considered, both diversity and the newly defined memory helper objective, significantly increase the convergence rate on this task.

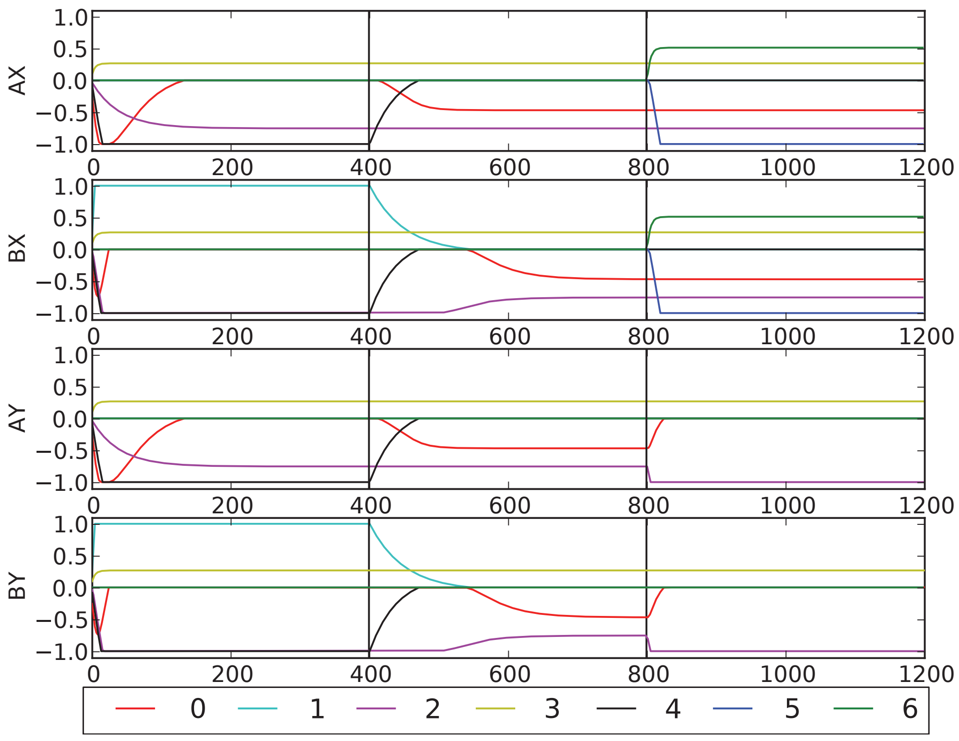


Figure 4.8: Internal behavior of the neurons corresponding to neural network displayed in Figure 4.7, for the 4 different sequences during the memory test.

The memory helper objective —and, to a lesser extent, the diversity objective— promote memory in the resulting networks. Moreover, the helper objectives are shown to have a large impact on generalization ability even though they aren't specifically designed to do so. This leads to the hypothesis that there is a link between the presence of memory in agents and the generalization ability on this task.

The memory helper objective is scenario-based and does not assume a specific neural structure for internal memory. This has two main interests:

- The method could potentially be used in any neuroevolution experiment involving elementary memory.

- The objective can select individuals with many different internal representations. It applies therefore a weaker bias on the network than directly constraining the structure.

Another methodological aspect highlighted in this chapter is the use of a multi-objective evolutionary algorithm. Additional objectives are simply added, selecting individuals that might have a low fitness regarding to the main objective, but have an original behavior or efficient internal representation. We believe that those individuals can be good stepping stones to efficient cognitive solutions.

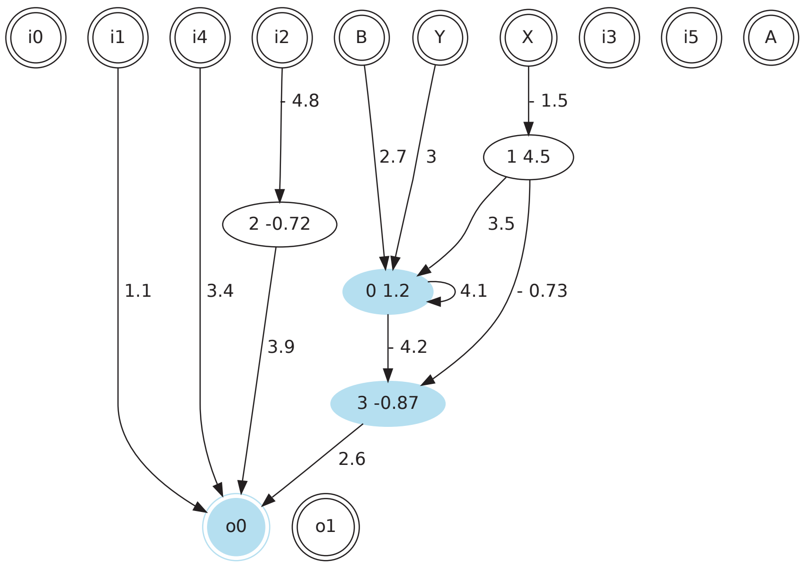


Figure 4.9: Resulting neural network with maximal fitness, memory and generalization

4.2.4.1 Scope and usefulness

This helper objective based on the memory capacity of internal nodes was successful, as it made it possible to achieve greater rates of robust memory in controllers. However the two following points can be discussed:

- First, even if a controller is selected because it has an internal neuron with memory, it doesn't mean that the memory is useful at all: for instance the neuron having memory could be connected to an irrelevant output, or even not connected to any output at all. Therefore probing into the neural network for finding memory might not be the best solution to promote effective memory.
- The objective cannot be used to study the origins of the emergence of memory in artificial brains, as it explicitly promotes internal structures with memory.

These points lead to the design of a new objective, which drives the evolutionary search without constraining explicitly the internal structure.

4.3 Behavioral Consistency Method

Drawing inspiration from the previous helper objective, a methodology to build selective pressures is defined. The goal is to create a rather general framework, which relies on the comparison of controllers in different scenarios. Instead of probing

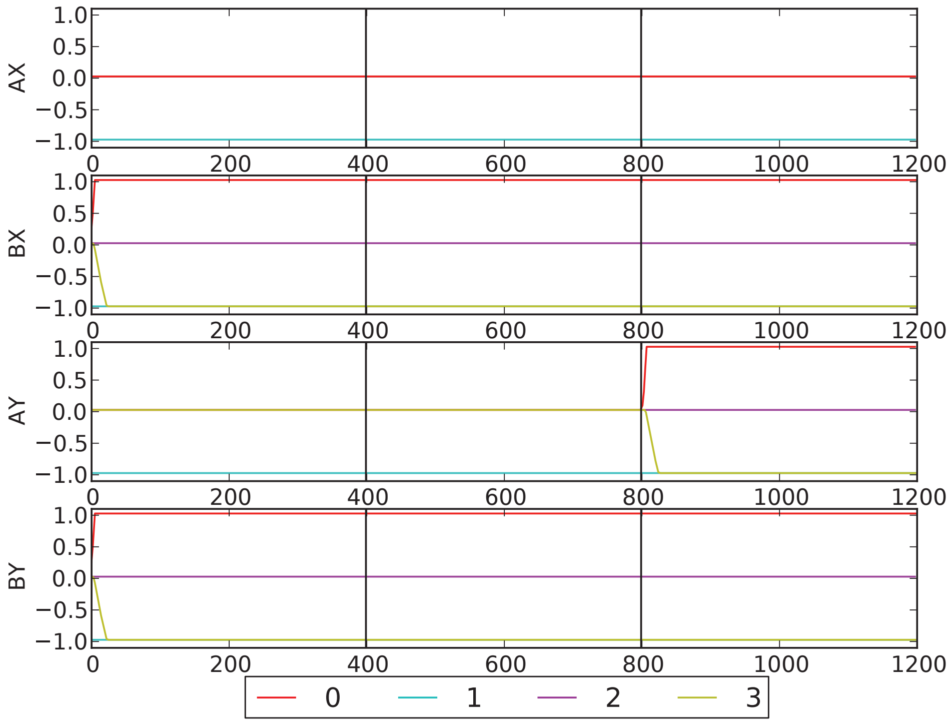
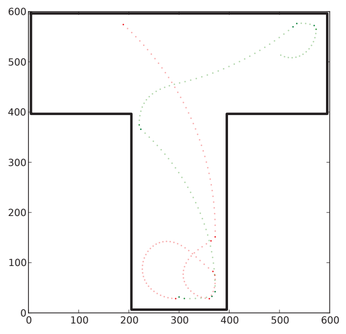
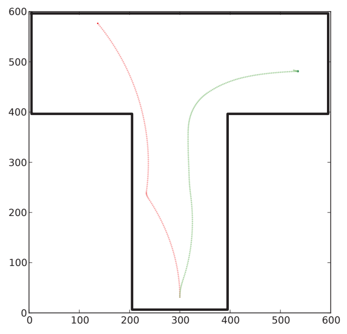


Figure 4.10: Behavior of internal neurons corresponding to neural network displayed in Figure 4.9, for the 4 different sequences during the memory test.



Successful, without Memory and Generalization



Successful, with Memory and Generalization

Figure 4.11: Trajectory of the robot in the T-maze of the two previous individuals in a single context.

inside of the networks of the controllers, the selective pressure is based on the behavior of the system. Globally, the selective pressures assess whether the behavior of the controllers is consistent in the different scenarios. It will therefore be referred as the Behavioral Consistency Method, or BC method.

4.3.1 Related Work

As of today, no straightforward methodology may help in designing a fitness function in a context of neuro-evolution. Section 2.2.3 relates the different approaches used in ER. In particular, Nelson et al. (2009) have made a review of the different fitness functions used in ER classified according to the degree of a priori knowledge incorporated in the fitness. Nelson states that the “behavioral” fitness functions, which imply explicitly defining a target behavior or a sub-part of this behavior requires a lot of *a priori* knowledge.

The Behavioral Consistency Method builds a fitness objective which still includes knowledge about the considered behavior, but does not specify this behavior at all. Instead, it specifies constraints that the behavior should fulfill. This leads to a more open-ended search, because potentially several very different behaviors may result from evolutionary runs.

4.3.2 Behavioral Consistency Method

4.3.2.1 Overview

The Behavioral Consistency Method aims at defining a selective pressure (the **Behavioral Consistency Objective**, or BCO) which ranks individuals based on how consistent their behavior is, without explicitly describing a target behavior.

A simple example would be a noise robustness BCO:

Individuals are simulated on scenarios with various levels of noise. Individuals that have close behaviors in those scenarios should be rewarded, while individuals whose behaviors are strongly affected (i.e. have very different behaviors) by noise should be punished. In that way, the BCO promotes individuals with a behavior resistant to noise.

The main idea of the method is that the BCO is calculated by simulating the individuals over different scenarios and comparing their features. The BCO rewards either their similarity or difference. The design of this objective actually consists in 1) defining the different scenarios and 2) choosing whether the corresponding behavior should be similar or different one with another. By rewarding the similarity between behaviors or, in contrast, their difference, the scenario based objective encourages the emergence of a coherent behavior.

This is very different from defining the features of a target behavior: by using comparisons over scenarios, the features of the behavior do not need to be specified. Instead, the comparisons between scenarios generate constraints —such as noise resistance or others — on the behavior.

4.3.2.2 Behavioral Space

The BC method relies on comparisons on the behavioral space. This is different from the memory helper objective defined in the previous experiment: instead of probing into the neural network to determine which part of the network should

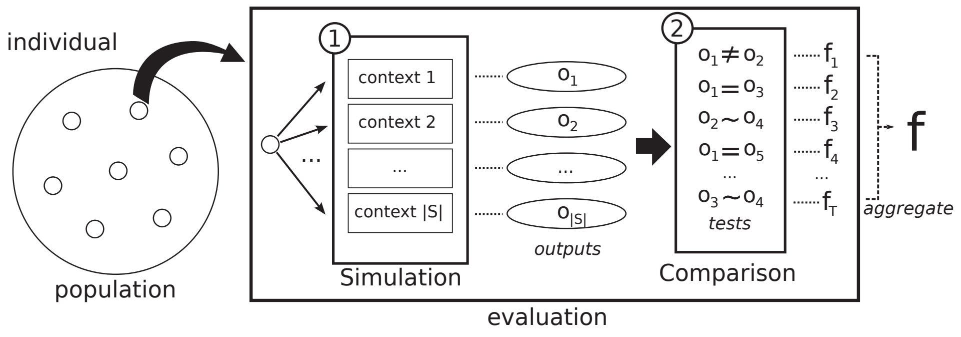


Figure 4.12: Details of the evaluation of agents by the BCO. 1) An agent is simulated in several predefined scenarios. During this simulation, a characteristic behavior is stored. 2) The behaviors are compared and checked for coherence, resulting in a partial fitness term f_i. Then, the partial fitness terms are aggregated into the final evaluation f.

have consistent properties, the BCO relies on comparisons on the behavioral space. This is a major aspect, as in ER, the mapping between genotype and the behavior is complex and depends on the environment the agent is in (See section 2.2.2 for more details about the choice of behavioral space).

4.3.2.3 Definition

The consistency objective relies on simulating one individual on a set S of different scenarios, as shown on figure 4.12. The consistency of the individual is then evaluated by comparing the behaviors (outputs). We denote $o_i(t)$ the simulated output (behavior) of scenario i after t time-steps. Depending on the problem considered, the experimenter defines different scenarios and different consistency constraints between scenarios. Three possible constraints will be used in the following experiments:

- output of scenario i is exactly the same as output of scenario j: $o_i(t) = o_j(t)$
- output of scenario i is different from output of scenario j: $o_i(t) \neq o_j(t)$
- output of scenario i has similar properties as scenario j: $o_i(t) \sim o_j(t)$. The definition of similarity is specific to each experiment, but typically means equivalent to within a translation.

The constraints are computed in the following way:

- $o_i(t) \neq o_j(t)$: $f_t = d(o_i(t), o_j(t))$
- $o_i(t) = o_j(t)$: $f_t = 1 - d(o_i(t), o_j(t))$
- $o_i(t) \sim o_j(t)$: $f_t = 1 - d_s(o_i(t), o_j(t))$

The normalized distance d denotes the difference between outputs behaviors. d_s is a similarity distance that usually describes how close the shapes of outputs are. d and d_s are specific to the experimental setup.

The final assessment of the quality of an individual is then computed by aggregating the fitness terms. For the sake of simplicity, the simplest aggregation is used:

$$f(x) = \frac{1}{T} \sum_{t}^{T} f_t \tag{4.1}$$

In short, building the consistency fitness objective involves three steps:

- Defining a collection of different scenarios
- Defining constraints between outputs of scenarios
- Defining how to compare outputs with distances d and d_s

For instance a consistency objective could be designed to build a navigation behavior that is robust to noise. In that case, one individual is simulated in two different scenarios:

- scenario 0: no noise added
- scenario 1: gaussian noise added

The constraint is then $o_0(t) = o_1(t)$. The fitness of an individual is then: $f_t = 1 - d(o_0(t), o_1(t))$ with d corresponding to a normalized distance between trajectories of the individual in the two scenarios. An individual that has very different trajectories in the two scenarios would then have a low fitness, while close trajectories would lead to a high fitness.

Summary The Behavioral Consistency Objective is a selection objective that drives the population towards individuals that follow best a set of predefined constraints. Defining those constraints adds *a priori* knowledge to the evolutionary process and biases the search. However it requires much less knowledge than the exact specification of a target behavior. Thus, the method is useful in cases where the target behavior is not known or should not be specified *a priori.*

The next two sections present the application of the BC method to two evolutionary experiments. Both of these original experiments involve the optimization (via artificial evolution) of a non-trivial system by specifying its exact target behavior. The BC method is applied to create an evolutionary objective which does not require the exact specification of a target behavior, in contrast with the original fitness functions.

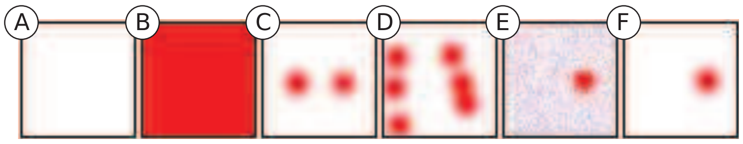


Figure 4.13: Typical inputs (scenarios) from left to right: A: empty B: full C: competition D: distractors E: noise F: simple

4.4 Attention Focus

4.4.1 Experimental Setup

The first experiment considered here is inspired by the work of Quinton (2010a). The goal is to use Continuous Neural Field Theory (Amari, 1977) to model attention selection. The model is taken from (Rougier and Vitay, 2006) in which the neural field is known to output a robust neural activity able to track perceptual information in noisy scenarios, even in the presence of distractors. In Quinton's work, the parameters of the neural field are optimized using an evolutionary algorithm, published in (Quinton, 2010a).

The neural field is represented by a two dimensional map, and the potential at position vector $\mathbf{x}$ and time t is $u(\mathbf{x}, t)$, with $\mathbf{x}$ in $[-0.5, 0.5]^2$. The field is stimulated by perceptual input $s(\mathbf{x}, t)$, and lateral connections. The dynamics of the neural field follows the equation (from (Quinton, 2010a)):

$$\tau \frac{\partial u(\mathbf{x}, t)}{\partial t} = -u(\mathbf{x}, t) + \int_{\mathbf{x}'} u(\mathbf{x}, t) w(\mathbf{x}, \mathbf{x}') d\mathbf{x} + s(\mathbf{x}, t) + h$$

h is the resting potential, set to 0 at first. The lateral connection weights follow the equation:

$$w(\mathbf{x}, \mathbf{x}') = Ae^{-\frac{|\mathbf{x}-\mathbf{x}'|^2}{a^2}} - Be^{-\frac{|\mathbf{x}-\mathbf{x}'|^2}{b^2}} \tag{4.2}$$

The lateral connection parameters A, B, a and b are optimized through evolution, as well as τ the inertial parameter of the dynamics. As in Quinton's work, the parameters are constrained in order to obtain a "Mexican-hat"-like lateral connectivity: $A > B$ and $b > a$. In order to simulate the dynamics, the map is discretized onto a grid of 50×50 units. The dynamics of the neural fields are simulated on different scenarios, in which the inputs of the neural field $s(\mathbf{x}, t)$ differ, as depicted in figure 4.13.

The first and second scenarios A and B correspond to constant inputs. C, D, E are 3 scenarios which correspond to Quinton's scenarios. In scenario C, a static input bubble is in competition with a second one of variable intensity. In scenarios D and E, a rotating input bubble must be tracked, in presence of distractors (D) or noise (E). Finally, scenario F has a single static input bubble and no distractors. The details of those scenarios can be found in (Quinton, 2010a).

4.4.1.1 Control experiment

The fitness used in the original experiment is used as a control fitness. It is based on the three scenarios C, D and E, and is described in (Quinton, 2010a). The principle is as follows: The neural field has to focus on the input bubble, and follow it if it moves. If the bubble intensity changes, some distractors are added, or noise is present, the focus should remain the same. The fitness depends on the position and shape of the output bubble at each time-step. This means that a very precise output behavior is expected.

4.4.1.2 Behavioral Consistency Objective

Using the BCO, the population is not evolved towards a given, *a priori* behavior. Instead, the objective rewards solutions with a consistent behavior when subject to noise, distractors, and more generally with a non degenerate activity (i.e. the activity of the map should converge, not saturate, have a consistent shape, etc.).

In order to compute the objective, several scenarios are defined. The first two correspond to scenarios with a totally activated/deactivated neural field (See A and B in Figure 4.13). All other scenarios correspond to one of the inputs C, D, E, or F. In these cases the initial position of the input bubble, the number of distractors or the noise level can vary. Table 4.4 details all the different scenarios considered. Four sets of scenarios are defined in the following way: S_{const} includes constant scenarios (the first two scenarios A and B in Figure 4.13), S_{nconst} includes non constant scenarios (All other scenarios). Additionally S_{nconv} includes not convergent scenarios (those derived from D and E have their input moving, therefore they should not converge towards a fixed output). On the other hand, S_{conv} represents convergent scenarios (those derived from C and F). From those sets, the following constraints are defined:

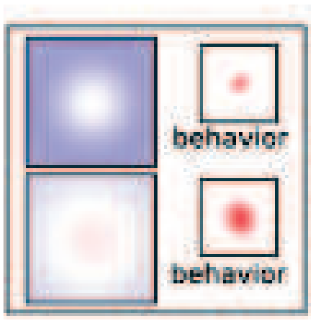
- The output behavior of scenarios from S_{nconst} should be different from constant behavior of S_{const} scenarios: $\forall i \in S_{nconst}, j \in S_{const},\ o_i(t) \neq o_j(t)$
- The output of S_{conv} should stabilize: $\forall i \in S_{conv}, \exists \varepsilon_t\, \forall t, t' > \varepsilon_t\, o_i(t) = o_i(t')$
- The output of S_{nconv} should be different over time:
  $\forall i \in S_{nconv}, \forall t > \varepsilon_t, \exists t' > t, o_i(t) \neq o_i(t')$

Furthermore, two constraints are added on the shape of the output activity (whatever the shape of the output activity, it should remain the same over time, and for different scenarios):

- The output of S_{nconst} should exhibit consistency in time:
  $\forall i \in S_{nconst}, \exists \varepsilon_t\, \forall t, t' > \varepsilon_t\, o_i(t) \sim o_i(t')$
- The outputs of different scenarios of S_{nconst} should be similar:
  $\forall i, j \in S_{nconst},\ o_i(t) \sim o_j(t)$

Table 4.4: Details of possible scenarios

type	description	scenario sets
C	4 different starting position	S_{nconst}, S_{conv}
D	5 number of distractors, 2 rotation directions	S_{nconst}, S_{nconv}
E	5 noise levels, 2 rotation directions	S_{nconst}, S_{nconv}
F	4 starting position	S_{nconst}, S_{conv}



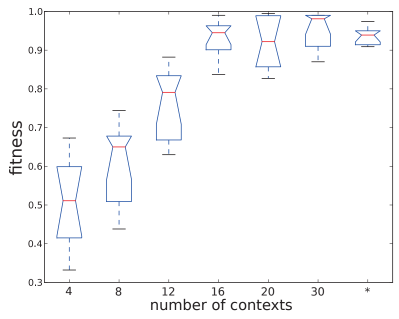


Figure 4.14: (Left) Two resulting behaviors obtained with the scenario-based fitness. Left is the lateral weight profile and right the corresponding output activity. (Right) Influence of the number of scenarios on the scenario-based fitness with 30 scenarios. $*$ represents parameters obtained with the fitness of the original experiment.

Finally, the distance between behaviors is computed in the following way:

$$d(o_i(t), o_j(t)) = \int_{\mathbf{x}} ||o_i(\mathbf{x}, t) - o_j(\mathbf{x}, t)|| \tag{4.3}$$

The similarity distance computes the same distance, after aligning the outputs:

$$d_s(o_i(t), o_j(t)) = \min_d \int_{\mathbf{x},\mathbf{x}'} ||o_i(\mathbf{x}, t) - o_j(\mathbf{x} - \mathbf{d}, t)|| \tag{4.4}$$

d corresponds to the shift between the center of activities between the two outputs. Details of this computation can be found in (Rougier and Vitay, 2006).

The experiment is run several times with different numbers of scenarios, in order to measure the bias added by the choice of scenarios. The scenarios are chosen randomly for **each run**, in order to reduce any bias in the result statistics.

Each individual is evaluated during 20 time-steps on each scenario. In order to avoid initial chaotic dynamics, the first 5 time-steps of the resulting behavior are discarded. Each experiment —with various numbers of scenarios— is run 10 times, with a population size of 20 and the number of generations is 30 (same values as in Quinton's experiment).

The population size is 200 and the algorithm stops after 1000 generations.

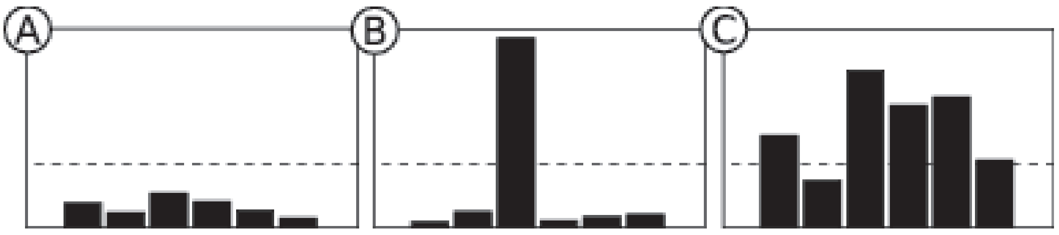


Figure 4.15: Typical scenarios from left to right (Input of the action selection model): A: empty or low noise B: easy selection C: competition

NSGA-II algorithm is used (Deb, 2001), and the source code is available online at `http://pages.isir.upmc.fr/evorob_db`

4.4.2 Results

The consistency objective with maximum number of scenarios provides results with the expected qualitative behavior, in all 10 runs. This means that the output signal (a bubble of activity) is quickly able to follow the input with a signal consistent in time and shape, and is not damaged by noise or distractors. The evolved systems exhibit then the expected function. Furthermore, different behaviors have been discovered, as shown in Figure 4.14, left. For instance, two very differently sized "blob" of activity were found, both respecting the constraints. This highlights an interesting property of the proposed method: it looks for behaviors respecting the given constraints, no matter how they manage to do it.

The number of scenarios used greatly influences the effectiveness of the consistency objective. In order to study this influence, the individuals obtained with k scenarios are tested on the fitness based on 30 scenarios. Additionally, an individual obtained with Quinton's fitness was tested on the 30-scenario-based fitness. The graph (Figure 4.14 right) shows that:

- not surprisingly, individuals obtained with Quinton's fitness perform well on the scenario objective. It should be underlined here that the reverse is generally not true as Quinton's fitness looks for a particular shape that is not the unique solution respecting the constraints as mentioned above;

- the number of scenarios needs to be high enough for the objective to be effective (around 16), but there is little difference in performance if the number of scenarios is over 16 ($p > 0.1$). The difference between 4, 8, 12 and 16 scenarios is significant ($p < 0.01$ for each).

4.5 Action Selection

This section describes the second experiment, based on a neuroevolution experiment on basal ganglia (Mouret et al., 2010). The goal of this experiment is to synthesize a neural network able to perform action selection, a cognitive function supposed to be performed by the basal ganglia. The search space is much larger than in the previous experiment, as the structure and parameters of the neural network are evolved. Action selection in the brain is the problem of choosing an action, given external and internal sensory information. The focus is on the process of selection of a single action among conflicting ones, also known as a winner-takes-all (WTA) circuit. This section first describes the original experiment and fitness, and then the definition of scenarios to build the new fitness function.

4.5.1 Experimental setup

4.5.1.1 Encoding

In order to build an action selection model that generalizes to different number of action channels, Mouret et al. (2010) used a map-based encoding in which parameters and structure of a neural network are evolved. The neural network is initially a feed-forward network, and the structure is modified through evolution by the addition of new neurons or maps of N neurons, and connections between them. Furthermore, connections between maps of neurons have additional evolved parameters that determine the nature of a connection: a 1-1 connection scheme connects each neuron of the input map to a single one in the output map, while a 1-all connection scheme connects one neuron to all neurons in the output map.

This experiment uses this map based encoding, with a modification: the connections between maps have an additional evolved parameter: an offset o. For instance, a 1-1 connection between two maps of size N with offset o connects neurons as follows: neuron i of the input map is connected to neuron $i + o \, modulo \, N$ of the output map.

4.5.1.2 Control experiment

The authors of the original experiment expect to build a model that reproduces the functioning of basal ganglia in the brain. The fitness function assumes full knowledge of the output and is described in (Liénard et al., 2010). At rest the output of the basal ganglia is active, and represents inhibition to the target. The selected action will then be the channel where inhibition is removed. The neural networks have one input map and one output map of N neurons, N corresponding to the number of channels. Over a collection of $K = 500$ random inputs (inputs range from 0 to 1), the most activated input neuron has to be selected, which means that the corresponding output should be close to zero while others should be close to one. It rewards individuals that desinhibit the selected channel and inhibit all others.

As this setup is more challenging than the previous one due to the complex search space, a behavioral diversity helper objective is added, in a multi-objective scheme. The diversity objective is based on the distance between the behaviors of the different models. The behavioral distance is a Euclidean distance of the output map activity at the last time-step of the simulation (see section 2.2.3.1 for further information about behavioral diversity).

4.5.1.3 Behavioral Consistency Objective

The goal of the BCO is to promote networks that perform action selection in any possible way, not only in a biologically plausible way.

Three types of scenarios are displayed in Figure 4.15. Scenario type A corresponds to a very weak noise input. Scenario B corresponds to a very simple action selection: one channel is set to 1.0 while the others are set to 0 plus a uniform weak noise. C represents randomly generated scenarios with random inputs.

Like in the previous experiment, 2 sets of scenarios are defined: S_{select} (standard scenarios in which a selection should occur) for scenario A, $S_{nselect}$ (degenerate scenarios with no selection at all) for all others. S is the set of all scenarios. The following constraints are added on the different scenarios:

- All the outputs should stabilize: $\forall i \in S, \exists \varepsilon_t \, \forall t, t' > \varepsilon_t \, o_i(t) = o_i(t')$
- The output behavior of scenarios from S_{select} should be different from the degenerate behavior of $S_{nselect}$ scenarios: $\forall i \in S_{select}, j \in S_{nselect}, \, o_i(t) \neq o_j(t)$
- The outputs of two different scenarios of S_{select} should have similar behaviors: $\forall i, j \in S_{select}, \, o_i(t) \sim o_j(t)$
- The outputs of two different scenarios in S_{select} which select the same channel should be exactly the same: $\forall i, j \in S_{select}, max_i = max_j \Rightarrow \, o_i(t) = o_j(t)$
- The outputs of two different scenarios in S_{select} which select different channels should be different: $\forall i, j \in S_{select}, max_i \neq max_j \Rightarrow \, o_i(t) \neq o_j(t)$

Finally the distances to compute difference and similarity are defined as follows:

$$\begin{aligned} d(o_i(t), o_j(t)) &= \textstyle\sum_k ||o_i(k,t) - o_j(k,t)|| \\ d_s(o_i(t), o_j(t)) &= \textstyle\sum_k ||o_i(k,t) - o_j(k-\delta,t)|| \end{aligned}$$

where δ is the shift between centers of activity of the two outputs, computed in a similar fashion as in the first experiment.

The population size is 200 and the algorithm stops after 1000 generations. NSGA-II algorithm is used Deb (2001), and the source code is available online at `http://pages.isir.upmc.fr/evorob_db`

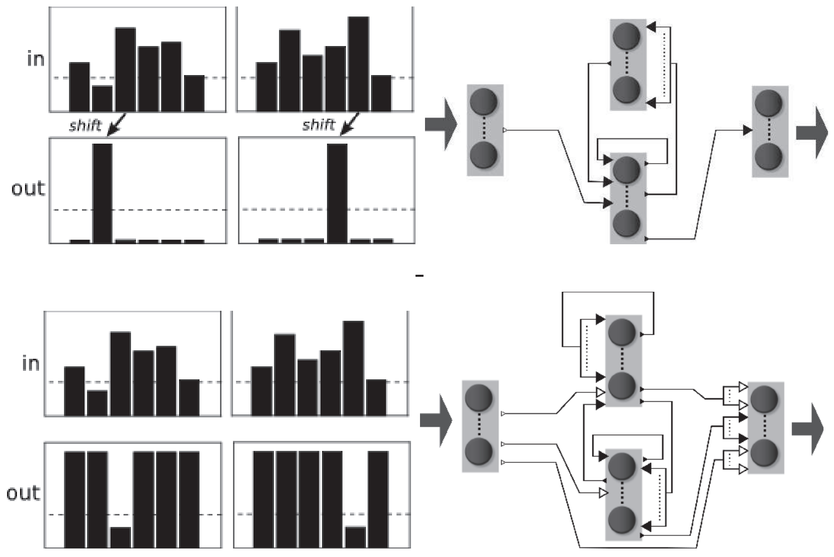


Figure 4.16: **(Left)** Two behaviors generated with the consistency objective. The graphs show a random input (in), and the output of the evolved model (out). Consistency objective fitness: 0.94 (top) and 0.96 (bottom). **(Right)** corresponding neural networks. Parameters and offsets are not displayed.

4.5.2 Results

Concerning action selection, the consistency objective was able to evolve successfully two main categories of solutions. Both of them realize action selection by outputting a coherent response for any input, and making a single output stand out from others. The first one (Figure 4.16, top), realizes the most intuitive action selection, and its neural network has two internal maps of neurons and excitatory recurrent connections. The second category (Figure 4.16, bottom) of behaviors are similar to (Mouret et al., 2010) results, except for a shift in the output channel index. The shift means that the index of the selected channel might be different from the index of the input channel with maximal activity. This is not a problem, as an action selection is still realized correctly. The corresponding neural network has two internal maps, but more connections, both excitatory and inhibitory.

The number of scenarios have exactly the same influence as in the previous experiment: with a low number of scenarios, the performance is not reliable, while with at least 15 scenarios, the performance stabilizes. The difference between 15 and 30 scenarios is not statistically significant (p-value > 0.1 Mann-Withney U test). The solutions obtained with the original fitness perform very well when evaluated with consistency objective.

4.6 Conclusion

In this chapter, a newly defined *helper objective* (based previously defined **M** representation test) was applied to improve the rate of emergence of memory in an evolutionary robotics context. Several obtained controllers have internal representations with a memory of the input stimulus. Those representations can be located on one or several neurons, and might not be simple binary memory units, as shown in the resulting networks of the experiment.

While only a few memory neurons would emerge when using only the main fitness objective, the use of *helper objectives* in a multi-objective way greatly improves the performance. Not only do the internal representations appear more often, but also the main fitness improves, as well as the generalization ability over previously unseen environments. This shows a correlation between the presence of internal representations and generalization ability. Finally, the two objectives — a diversity objective, and the new helper objective — both increase significantly the performance, and can be used simultaneously to further increase the success rate. This means that the objectives do not overlap much and both help the evolutionary process, and advocates for the use of multi-objective evolution.

The second part of the chapter is focused on the definition of a new framework for designing fitness functions. The Behavioral Consistency Method enables the researchers to build fitness objectives that reward individuals based on their fulfillment of previously designed constraints. Using this method, one can add evolutionary pressures towards satisfying constraints such as being resistant to noise, to distractors, etc. This new fitness objective does not require the exact knowledge of the target behavior. Instead, the *a priori* knowledge is incorporated in the definition of scenarios, and the comparison between the behaviors in these scenarios.

The consistency method is used to explicitly drive the evolutionary search toward the emergence of cognitive properties such as Action Selection and Attention Focus. The method showed results comparable to the original evolutionary experiments.

Furthermore, the consistency method does not drive the evolutionary search to an explicit behavior. This means that the exact knowledge of a behavior presenting the desired property is not required. In addition, this does not restrain the evolutionary search to one particular solution, leading to the emergence of a diversity of different solutions. It is important to note that even if the knowledge of a behavior is not required, the experimenter includes knowledge while building the different scenarios and comparison between scenarios. While the design of these scenarios is straightforward in simple cases, one can expect more challenges for difficult properties to emerge.

The Behavioral Consistency Method can be used to efficiently build a helper objective, used alongside other objectives. For instance, a Goal-objective may be defined, which requires the simulation of one or several scenarios. Additional simulations of scenarios may then be needed for the computation of the behavioral consistency objective. This methodology can be used to add specific selective pressures to an evolutionary experiment.

Chapter 5

Environmental Pressures and emergence of Internal Representations

Goals

- Study the impact of selective pressures on the emergence of internal representations.

Method

- Definition of Circular Maze Evolutionary Robotics Protocol (CMERP), a new ER experiment.

Results

- The Environmental Pressures and novelty significantly influence the presence and type of internal representations.
- The presence of internal representations is correlated with the ability to generalize to new contexts.
- Specific structures present similarities with some Computational Neuroscience networks.

Contents

5.1 Selective Pressures for the emergence of Internal Representations . . . 100

5.1.1 Task and World Complexity . . . 100

5.1.2 Helper Objectives and Specific Pressures . . . 101

5.2 Circular Maze Evolutionary Robotics Protocol . . . 102

5.2.1 Goal-oriented and Novelty Objective . . . 105

5.2.2 Behavioral Consistency Objective and Environmental Pressures 108

5.2.3 Representation Tests . . . 111

5.2.4 Setups and Selective Pressures . . . 113

5.3 Results: Emergence of Internal Representations under Environmental Pressures . . . 114

5.3.1 Impact of Selection Pressures . . . 114

5.3.2 Resulting Architectures 121
5.3.3 Lineage and selection for Internal Representation 126
5.4 Conclusion 129

5.1 Selective Pressures for the emergence of Internal Representations

The reasons for the emergence of internal representations in natural and artificial systems are not well known. Several ER studies have studied the origins of internal representations, and whether specific selective pressures or encodings were necessary for representations to emerge.

5.1.1 Task and World Complexity

The most straightforward approach in ER to obtain controllers with representations is the design of a *representation-hungry* task. The fitness function reflects the completion of the task, and no further selective pressure is added. Internal representations should then naturally emerge when the population converges towards high fitness values.

Marstaller et al. (2012, 2013) evolve complex representations for adaptive behavior. The task involved implies discrete states which the robot must remember to be able to solve the task. The structure of the controller is not constrained and mutations can generate new internal nodes. The authors "show that when the world (and task) is complex enough, agents react to this challenge by evolving representations of that world."

Cooperation and the evolution of intelligence In the article by McNally et al. (2012), the authors build a model of social interactions using a game theory protocol. The agents decide their next action depending on a neural network, in which the structure (number of internal nodes) can evolve. They define the "intelligence" level of their agents as the number of internal nodes present in their network. They show that the selection for intelligence is correlated with cooperation between agents: when the population is using mostly cooperative strategies, the evolutionary process is more likely to select individuals with high intelligence. This could show that social interactions promote the presence of internal representations in a population of artificial agents.

However, an analysis from the literature (Nolfi, 2002; Van Dartel et al., 2005; Pinville, 2013) as well as the preliminary experiments on the Ball Collecting (See Chapter 3) task show that the definition of a "representation hungry" task in Evolutionary Robotics is not straightforward. Moreover, even if a task might require internal representations to be optimally solved, there is no evidence showing that the evolutionary process drives the solutions towards such optimal solutions. Instead, in tasks such as a continuous T-maze (Pinville, 2013) the evolutionary process falls into local optima, which have no internal representations. Figure 5.1 illustrates this problem, by displaying a fictive fitness landscape:

- The initial population is driven to the easily reachable local optimum by the goal objective fitness (solid arrow). Solutions with internal representations are probably more difficult to reach than solutions without representations. This is illustrated by the presence of a gap between the local optimum and the global optimum.
- The use of diversity objective and other selective pressures (dashed arrows) could drive the population out the local optimum to reach new regions of the fitness landscape.

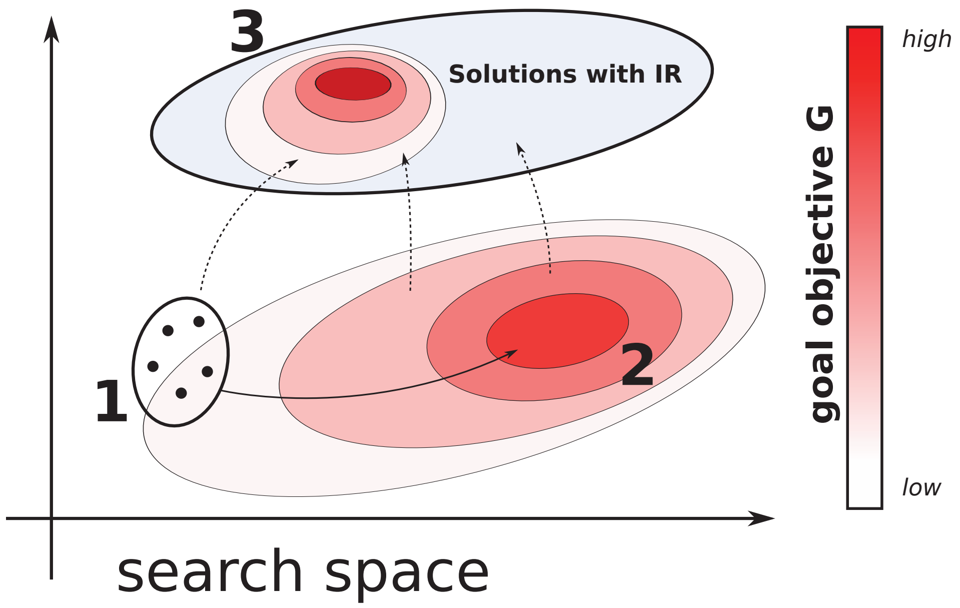


Figure 5.1: Representation of a fictive fitness landscape. The different level of red color saturation represent the goal objective: how well the solution performs. (1) the initial random population (2) A local optimum with regards to the goal objective, with no internal representation (3) The global optimum, with internal representations (IR).

5.1.2 Helper Objectives and Specific Pressures

The first class of helper objectives rely on a better exploration of the search space. Behavioral diversity (Mouret and Doncieux, 2012a) and Novelty (Lehman, 2008) have been applied in a multi-objective way, as helper objectives. These selective pressures drive the population towards unexplored parts of the search space, and can thus prevent the evolutionary process from premature convergence. More information about diversity and Novelty can be found in Section 2.2.3.1.

Pinville (2013) noticed how fine-tuned and fragile the local optima solutions were: by modifying a few parameters in the experimental context, the evolved controllers weren't able to perform the tasks anymore. The author then hypothesized that promoting the generalization ability (Pinville et al., 2011) of the controllers might have an effect on their internal structure — in that case the emergence of internal memory. The conclusion is that promoting diversity as well as generalization ability both increased the proportion of runs in which an internal, robust memory emerged.

Environmental Pressures Several researchers have hypothesized that variability and noise in sensorimotor control have had a strong impact on how natural systems behave (Körding et al., 2006; Gigliotta et al., 2010). Motor control and sensor information processing can then be improved by having internal models which cope with noise and other variabilities. These variabilities will be referred as **Environmental Pressures** as they are naturally present in real environments.

These environmental pressures have been applied to several evolutionary robotics experiments:

- Gigliotta et al. (2010, 2011) evolve a robot controller subjected to sensory deprivation. They show how predictive internal models found by evolution can solve the tasks. The architecture of the model already incorporates *a priori* knowledge on how the model works, and the structure of the neural network is carefully designed and fixed.

- Wang et al. (2013) add environmental pressures on a prey-predator evolutionary robotics experiment. They show how the introduced variabilities (noise and sensor deprivation) influence the behavior of their controllers. However their controllers are not based on neural networks and cannot be used to study the emergence of internal representations.

Conclusion Environmental pressures, other helper objectives, as well as the complexity of tasks have been shown to have an impact on the evolutionary processes. This chapter aims at studying the influence of these selective pressures on the emergence of internal representations, with the architecture of the neural network modified through evolution.

5.2 Circular Maze Evolutionary Robotics Protocol

A new protocol In section 3.2, several *representation-hungry* experiments were briefly described: Minimally Cognitive Behaviors, Beer (1996, 2003), Predictive Models Nolfi and Tani (1999); Tani and Nolfi (1999), Evolution of representations

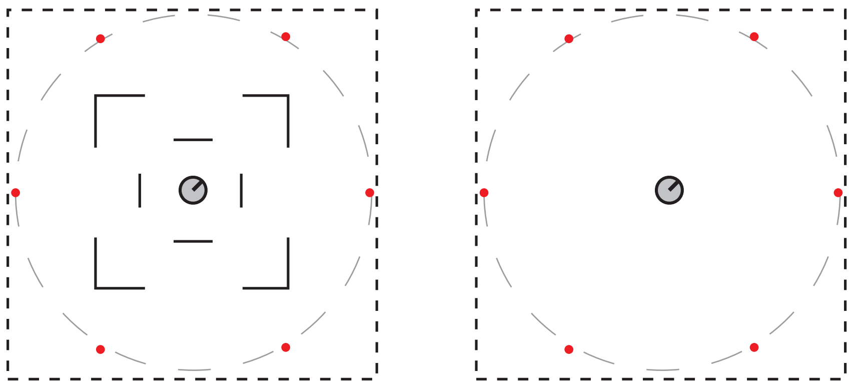


Figure 5.2: Simulated environments for the CMERP. The robot starts in the center of the arena, and sees one of the 6 distant visual landmark (not represented), at 6 different angles. The robot has to navigate towards one of the 6 goal points. Two environments are considered (Left) environment with walls “arena” (right) simplified environment with no walls “void”.

in a discrete environment Edlund et al. (2011); Marstaller et al. (2012). Such experiments could have been used to study the emergence of representations. However, the a new protocol was chosen, and the two following emphasis motivated this choice:

- The evolution of mobile robot controllers in continuous spaces.
- The use of protocols loosely related to Computational Neuroscience protocols.

The Circular Maze Evolutionary Robotics Protocol (CMERP) is an ER task introduced in this section. The task was designed to potentially require complex representations, but still be a task as simple as possible.

The principle of the task is as follows: a simulated robot is located at the center of an open environment. It receives a distant visual cue, and has to navigate towards that cue. Two environments are considered (see figure 5.2), a simple one with no walls (named “void” environment), and another one in which the robot has to navigate between walls (named “arena” environment).

The robot is controlled by a neural network as shown in figure 5.3. The inputs of the neural network are of three different kinds:

- Visual input. A map of neurons receives input from a visual landmark (see figure 5.4). The activity of the map is Gaussian, centered in the direction of the visual cue. The activity of the map is relative to the robot: for instance, when the robot rotates left, the center of the Gaussian activity shifts to the left of the map. The map is circular, therefore the activity is wrapped at its boundaries.
- Laser input. 3 range-finding lasers. The neurons are not activated if no walls are in front of the laser direction; if there are, the activation depends on the

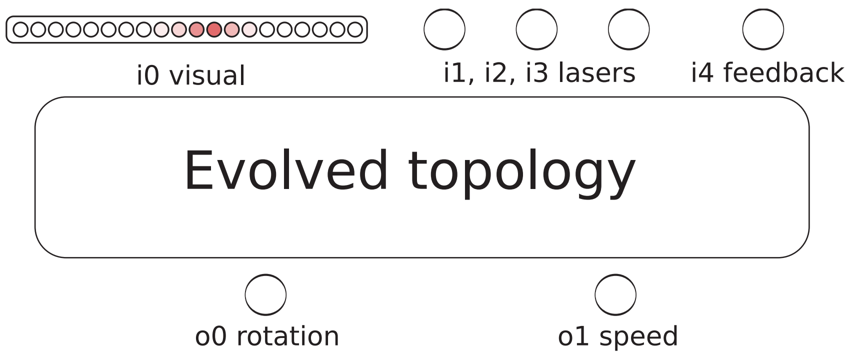


Figure 5.3: Network used for the CMERP. The inputs are: 1) A map of neurons 2) 3 single neurons for distance sensors 3) A proprioceptive feedback input. The outputs correspond to the rotation and speed of the robot.

wall distance (the closer, the higher the activation). In the Void environment, all laser inputs are set to 0.

- Feedback input. The robot can sense its own rotations: it receives the output rotation as input.

The output neurons are tied to the robot movements in the following way:

- Rotation output. The activation of the neuron controls the rotation speed of the robot. The activation $a \in [0, 1]$ is mapped into a rotation factor $r \in [-0.5, 0.5]$ (radian).
- Speed output. The activation of the neuron controls the speed of the robot. The activation $a \in [0, 1]$ is mapped into a movement in the robot direction of $m \in [-4, 4]$ (pixels).

During the simulation, the robot is first rotated depending on the Rotation output, then moved depending on the Speed output.

This protocol is inspired by the two following experiments:

- The Barnes Maze, a protocol used in biology to determine what kind of cue (local or distant) the animals use for navigation. This protocol was used for determining whether some animals had spatial memory (LaDage et al., 2012).
- Head Direction models tested on robots: the protocol is related to Head Direction cells (See part 3.1.3.2), because of the nature of the inputs (a visual cue depending on the direction the robot is facing). In the work of Kyriacou (2011), the author studies a biologically inspired model of head direction cells, and uses an evolutionary algorithm to evolve some of the parameters associated with this model. The head direction cells is represented as a map of neurons, from which the head direction can be decoded. Head direction cells are influenced in different ways by idiothetic (self-centered) and allothetic (not

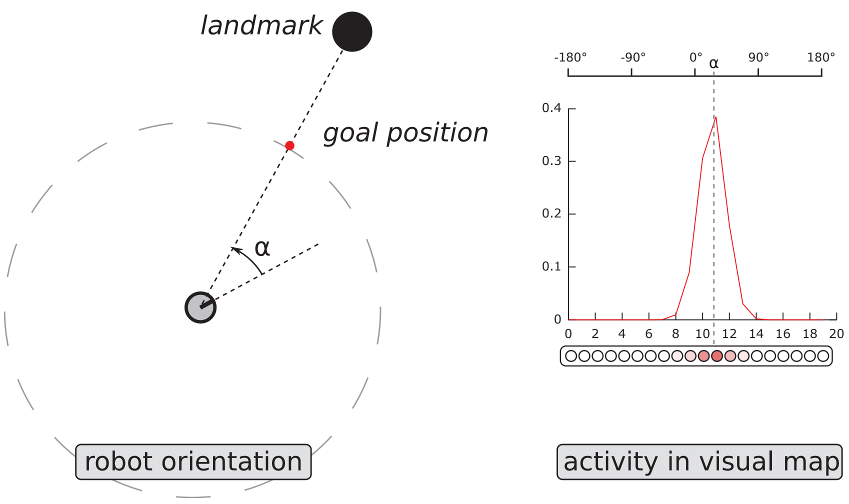


Figure 5.4: Visual inputs. (Left) α is the angle difference between the robot orientation and the landmark angle.

self-centered) cues. When some of these inputs are missing (visual occlusions) or when subjected to noise, the direction decoded from the head direction cells should still exhibit a consistent behavior. Therefore the map containing the head direction cells can be considered as a **N+M** Representation.

5.2.1 Goal-oriented and Novelty Objective

5.2.1.1 Goal-oriented Objective

The goal-oriented objective of the evolutionary process (denoted **G**) corresponds to how well the robot reaches the 6 points, for the 6 different landmark positions. Let $s_0, \ldots, s_5$ be six different scenarios. Each scenario corresponds to a simulation of 200 time-steps with the robot starting from the initial position, and a different visual landmark:

- s_0: landmark at $angle = 0$, corresponding to the rightmost goal in figure 5.2.
- s_1: landmark at $angle = \pi/3$
- $\ldots$
- s_5: landmark at $angle = 5\pi/3$

The fitness $G \in [0, 1]$ corresponds to the number of goals reached in the 6 scenarios. When the robot navigates out of the radius, the simulation is ended. The position of the robot is then discretized (see figure 5.5), and the robot position (x, y) corresponds to one of the $9x9$ discrete cells. If the robot is in the goal cell corresponding to its visual landmark, the fitness G is increased by $1/6$; otherwise, the fitness is not increased.

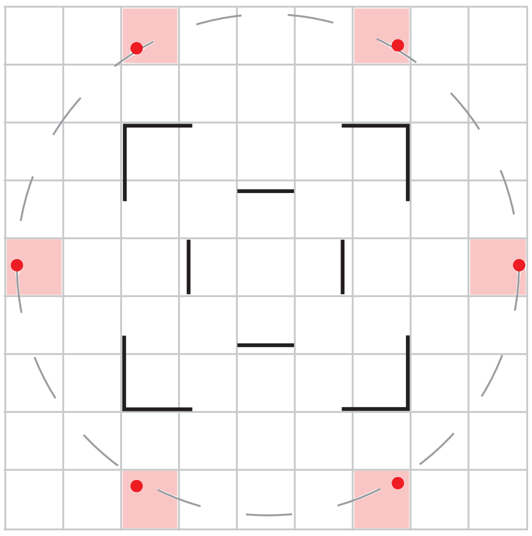


Figure 5.5: Grid discretization of the simulated environment. The robot position (x, y) corresponds to one of the $9x9$ discrete cells. The goal cells are represented in red.

The definition of the Goal-oriented objective is discrete. This leads to the two following difficulties:

- The evolutionary process faces the *bootstrap problem*: it is very likely that all initial controllers do not reach any final goal point (in the Arena environment). Therefore the goal fitness of the whole population may very well be 0, and no selection would occur.
- The goal-fitness can only take 7 different values in $\{0, \frac{1}{6}, ..., 1\}$. This results in very little evolutionary gradient. Instead, many fitness *plateaus* are present in the evolutionary experiment.

However, this choice is motivated by several reasons:

- Adding artificial gradients in the fitness to "help" the evolutionary process may bias the resulting solutions toward specific behaviors.
- Recent ER results show that adding bias in the fitness function can lead to worse results (Nelson et al., 2009; Pinville, 2013), because of the local optima it might generate.

Last, the lack of selection induced by the *bootstrap problem* and fitness *plateaus* are compensated by helper objectives as described in the following paragraphs.

5.2.1.2 Novelty Objective

The novelty objective (denoted **N**) is a helper objective added to reward individuals based on their novelty. Such objectives can help the evolution overcome premature

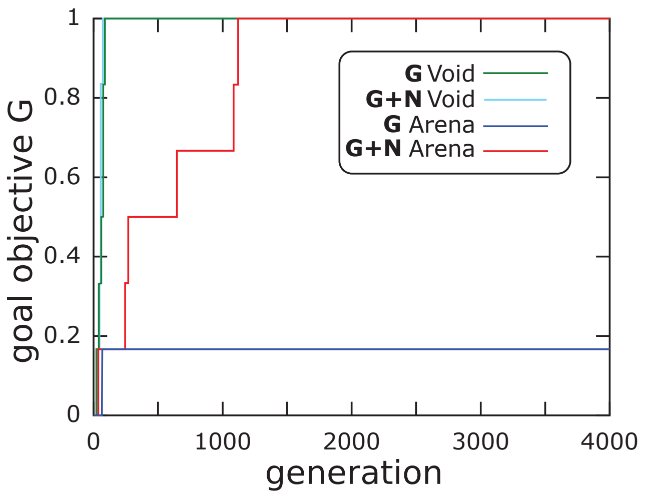


Figure 5.6: Median fitness out of 20 runs for the Void and Arena environments, with Goal objective **G** and eventually Novelty objective **N**.

convergence problems, as well as driving the evolutionary process towards unexplored search spaces (Doncieux and Mouret, 2010a). In this experiment, a Novelty objective was added (See paragraph 2.2.3.1), which rewards individuals that have novel behaviors. The Novelty objective can be used solely (as in the original experiments on Novelty), or alongside with other objectives (Mouret, 2011).

The behavior description for computing the distance between behaviors is the "Discretized trajectory" (See section 2.2.3.2), and the distance is the Edit Distance. The trajectory is sampled over time and discretized into a sequence of cells (see figure 5.5 for the discretization). In other words, individuals that have trajectories different to trajectories previously found by the evolutionary process will have a high novelty score.

5.2.1.3 Preliminary Experiments

Using the Goal objective **G** and the Novelty objective **N**, preliminary evolutionary experiments are conducted. The median convergence for the Void and Arena environments is displayed on Figure 5.6. This figure shows that the convergence rate strongly depends on the environment (Arena or Void). The evolutionary process quickly converges in the case of the Void environment, with very simple behaviors (see figure 5.7). In this figure a very simple controller solves the task: the robot simply rotates when it is not facing the landmark, and moves forward otherwise. Both the **G** and **G+N** setups converge in less than 200 generations.

By adding walls in the Arena environment, the task is more challenging. The evolutionary process do not converge at all when using only the goal objective **G**, and most of the runs result in robots reaching only a single target out of 6 ($\mathbf{G} = 0.1666$). By adding the novelty objective **N**, the process converges in average in ~ 1200 generations. Figure 5.8 displays a typical trajectory found with **G+N** with

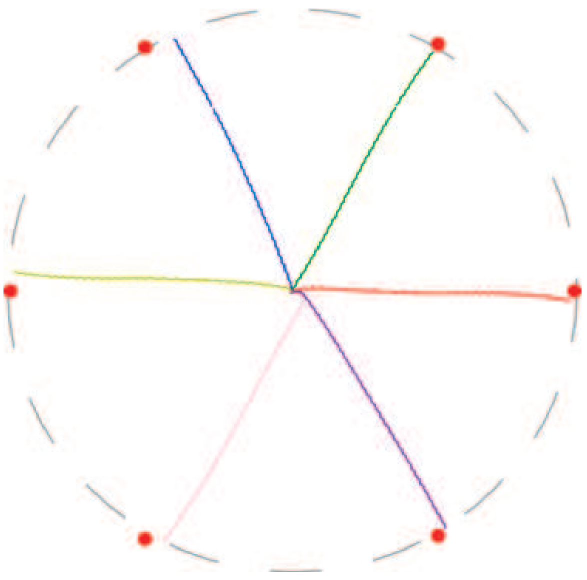


Figure 5.7: Trajectory of a controller with maximal goal objective ($\mathbf{G} = 1$) obtained in setup **G+N** (Goal objective + Novelty objective), on the Void environment. Each color represents a different scenario with a different goal and landmark position.

maximal fitness. The behavior is much more difficult to understand and involves many rotations. The controller generating this behavior is displayed on figure 5.9: it has no internal node, and is purely reactive.

Preliminary conclusion By using of Goal objective and novelty **G+N**, controllers with maximal fitness with regards to goal-objective **G** can be successfully evolved. The presence of successful solutions with no internal nodes confirms that the task is not “representation-hungry”.

5.2.2 Behavioral Consistency Objective and Environmental Pressures

This section defines “Environmental Pressures” that can be added to the evolutionary process. To do so, the Behavioral Consistency Method (BC method) (See previous Chapter) is applied. In that way, a new objective (denoted **E**) is added to the evolutionary process, which rewards individuals that have a consistent behavior under the following pressures:

- sensor noise (denoted **E (n)** for Environmental pressure with Noise)
- visual occlusions (denoted **E (o)** for Environmental pressure with Occlusions)
- both (denoted **E (no)** for Environmental pressure with Noise and Occlusions)

Solving the task (having a perfect **G** score) with a good resistance to noise and occlusions (good **E (no)** score) should be “representation-hungry”.

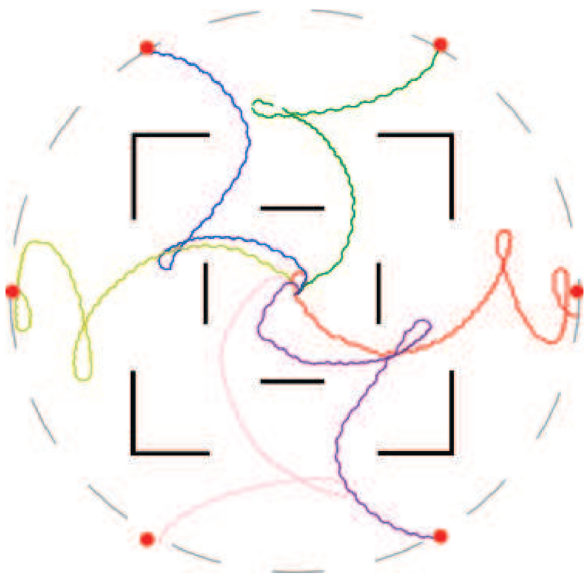


Figure 5.8: Trajectory of a controller with maximal goal objective ($\mathbf{G} = 1$) obtained in setup **G+N** (Goal objective + Novelty objective), on the Arena environment. Each color represents a different scenario with a different goal and landmark position.

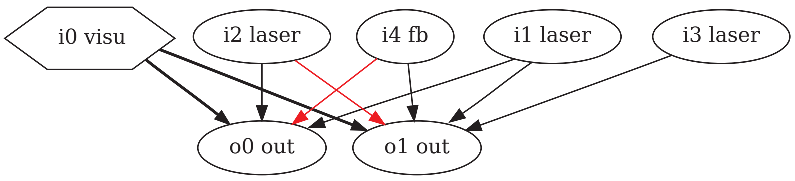


Figure 5.9: Neural network obtained with fitness **G+N** (Goal objective + Novelty objective), on the Arena environment. Red arrows represent inhibitory connections (i.e. with negative weights); bold arrows represent Gaussian weights connections. Biases and offsets are not represented. The trajectory resulting from this controller is displayed on figure 5.8.

The BC method implies three steps (See section 4.3.2 for full description):

1. Defining a collection of different scenarios
2. Defining constraints between outputs of scenarios
3. Defining how to compare outputs with a behavioral distance

Definition of scenarios For all environmental pressures, the scenarios s_0 to s_5 (scenarios with different goal positions) are used and altered with either noise or occlusions:

- Let s_6, s_7, s_8 be 3 copies of s_0 with a uniform noise $\nu \in [-0.1, 0.1]$ added to the visual input.
- Let s_9 (respectively s_{10} and s_{11}) be a copy of s_1 with a muted visual input after 150 time-steps (respectively 100 and 50 time-steps).
- Let s_{12} (respectively s_{13} and s_{14}) be a copy of s_4, with a uniform noise $\nu \in [-0.1, 0.1]$ added to the visual input, and muted visual input after 150 time-steps (respectively 100 and 50 time-steps).

Definition of constraints The robot controller should behave coherently even in presence of noise or occlusions. The constraints can then be written in the following way:

- Noise (**E (n)**): $s_6, s_7, s_8 = s_0$
- Occlusions (**E (o)**): $s_9, s_{10}, s_{11} = s_1$
- Noise and Occlusions (**E (n+o)**)
  1. $s_6, s_7, s_8 = s_0$
  2. $s_9, s_{10}, s_{11} = s_1$
  3. $s_{12}, s_{13}, s_{14} = s_4$

In other words, if the controller is adapted to the environmental pressures, altered scenarios should have a behavior similar to the corresponding unaltered scenario s_0, s_1 or s_4.

Comparison of behaviors Each behavior is described by the controller's "Discretized trajectory" (See section 2.2.3.2) over the $9x9$ cells grid (see figure 5.5 for the discretization). The comparison of two trajectories is realized using an Edit distance.

Summary Environmental Pressures (Resistance to Noise **E (n)**, Occlusions **E (o)** and both **E (n+o)**) are added as an additional objective through the use of the Behavioral Consistency method.

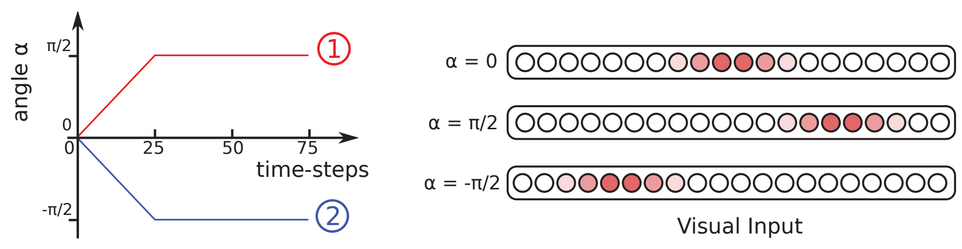


Figure 5.10: (left) Angle evolution for two scenarios (ds_1 top red and ds_2 bottom blue) (right) Illustration of the visual inputs for different angles, $0, \pi/2, -\pi/2$

5.2.3 Representation Tests

The emergence of representations is studied using the **N** and **M** representation tests defined in Chapter 3. Each internal node (map or single neurons) is tested for representations. As in the BC method, the representation tests require to define input scenarios, and a way to compute the difference of activity of a node in different scenarios.

In the CMERP, the focus is on the representation of the visual input, thus **N** and **M** representations are tested with regards to resistance to noise and occlusions in the visual input only (disregarding occlusions and noise in the lasers and feedback inputs). Furthermore, the tests are realized in a disembodied setup: instead of simulating the whole robot controller in its environment, the network is given some artificial inputs and only the internal nodes activity are taken into account; the outputs are not taken into account. This choice is motivated by three factors:

- In this disembodied setup, all controllers receive the same neural inputs, so their internal activity can be more easily compared and analyzed.
- The representations found will not depend on interaction with a specific environment.

A disembodied scenario (noted ds) here consists of the visual input activity over 75 time-steps. Scenarios are defined in the following way:

- ds_1 and ds_2 have different visual inputs as shown in figure 5.10). All other inputs are set to 0. The set S_d for computing input dependence consists of these two scenarios.
- For noisy scenarios: the set of scenarios S_n includes ds_1 as well as 10 copies of ds_1 with uniform noise added ($\varepsilon \in [-0.1, 0.1]$ is added to each neuron in the input map).
- For occlusions scenarios: S_o includes ds_1 as well as an altered copy of ds_1 with the input muted from time-step 50 to time-step 75.

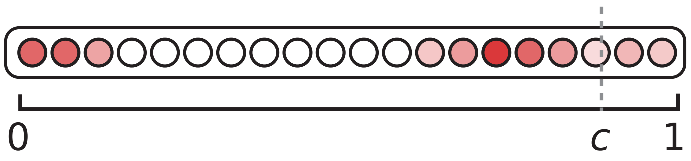


Figure 5.11: Computation of the center of mass of a neural map. The result is a simple center of mass of the neurons, each weighted by their activity. The computation also involves the fact that the map is circular to find the true center of mass.

Difference of activity Depending on the nature of the internal nodes, different distances are used to compare the activity of the internal node. If the node x is a **single neuron**, the activity difference in scenario i and j is the summation of the difference of neural activity over the last 25 time-steps:

$$d_a(x_i, x_j) = \sum_{t=50}^{t=75} |a^t(x_i) - a^t(x_j)|$$

If the node is a **map of neurons**, the information can be decoded in several ways (Salinas and Abbott, 1994; Deneve et al., 1999), however for the sake of simplicity, the "center of mass" decoding was chosen. Each neuron is weighted by its activity, and the center of mass is computed (This calculation includes the fact that the map is circular; see figure 5.11). This means that the map of neurons encodes a continuous variable $c \in [0, N-1]$ (N is the number of neurons in a map). The difference of activity of a map x in scenario i and j is then the summation of the difference of their center of mass over the last 25 time-steps:

$$d_a(x_i, x_j) = \frac{1}{N} \sum_{t=50}^{t=75} |c^t(x_i) - c^t(x_j)|$$

The difference is computed modulo N as the maps are circular.

Summary After the evolutionary process, the representation tests are performed on each internal nodes of the resulting controller. Tests determine whether each node is a **N**, **M**, **N+M** Representation, or no representation at all.

Generalization Ability The ability of a controller to generalize to contexts unseen during evolution is a very important factor. As the individuals are only evaluated on very few scenarios, they are likely to perform poorly on unseen scenarios. The ability to generalize is then an interesting measure of the quality of the resulting solutions, and has been explicitly sought (Pinville et al., 2011). To assess the generalization score of a solution, it is tested on a large number of scenarios, *after the evolutionary process*. The scenarios have the following properties:

Setup Name	Goal Objective	Novelty Objective	Environmental Pressure
G	•		
N		•	
G+N	•	•	
G+N+E (n)	•	•	Noise
G+N+E (o)	•	•	Occlusions
G+E (n+o)	•		Noise and Occlusions
G+N+E (n+o)	•	•	Noise and Occlusions

Table 5.1: Summary of setups: Combinations of different selective pressures applied to the CMERP.

- There are 6 different landmark positions as in figure 5.2.
- 10 instances of uniform noise are added ($\varepsilon \in [-0.1, 0.1]$ is added to each neuron in the input map); plus one instance with no noise at all.
- 3 different level of occlusions are added to the visual input (muted visual input after 50, 100 and 150 time-steps); plus one instance with no occlusions at all.
- The starting position of the robot is slightly modified (the robot is rotated).

The difference with scenarios seen during evaluation is that the controllers are tested in all possible landmark position, noise instances, occlusions levels, plus their starting position is slightly modified. These $6 \times 11 \times 4 = 264$ scenarios are simulated after the evolutionary process. If the robot reaches the goal, the generalization score g is incremented by $1/264$. A controller then has a generalization score $g \in [0, 1]$ which indicates whether it is capable of realizing the task in scenarios slightly different from those encountered during the evolutionary process.

5.2.4 Setups and Selective Pressures

All these selective pressures are tested on evolutionary runs on both the Void and Arena environments.

All setups were run 20 times, with a population of 200 individuals over 4000 generations. The experiments were coded using the SferesV2 evolution framework (Mouret and Doncieux, 2010), using the NSGA-II algorithm, and the EvoNeuro2 encoding (See Section 2.3.4.3). Note that the EvoNeuro2 encoding allows for two types of internal nodes: maps of $N = 20$ neurons or single neurons. When a new node is added, a parameter $p \in [0, 1]$ is associated with the node, at random. If this parameter is below 0.5, the node is a single neuron, otherwise it is a map of neurons: there is no natural bias towards a type of internal node. Parameters p are subjected to mutations — along with the other parameters — through the evolutionary process. Mutations rates and other parameters are displayed in Appendix A.1.

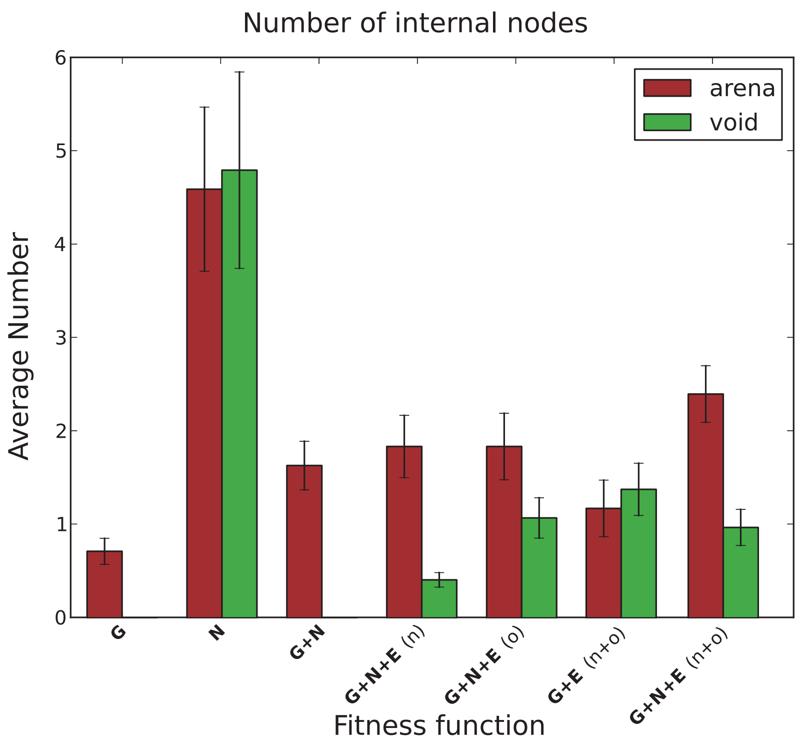


Figure 5.12: Number of internal nodes for the different fitness setups.

5.3 Results: Emergence of Internal Representations under Environmental Pressures

5.3.1 Impact of Selection Pressures

This section details the influence of the different selective pressures, as well as the two different environments, on the emergence of internal structures and internal representations. Globally, most results focus on the Arena environment, as it provides more interesting insights than the Void environment. Comparison between Arena and Void internal nodes and representations are detailed in Section 5.3.1.

Number of internal nodes Figure 5.12 displays the average number of internal nodes obtained for all the different selective pressures and environments. The novelty search (**N**) clearly produces the largest number of internal nodes, a very similar number for both Void and Arena environment. This can be explained by the fact that novelty search rewards trajectories which are very different from previously found trajectories. Internal neurons can help generate internal dynamics, delays or activity modifications which enables for novel trajectories.

Overall, the environmental pressure **E** tends to slightly increase the average number of internal neurons (the difference is statistically significant for the void environment, but not for the arena one).

Interestingly, absolutely no internal nodes were found in the Void environment

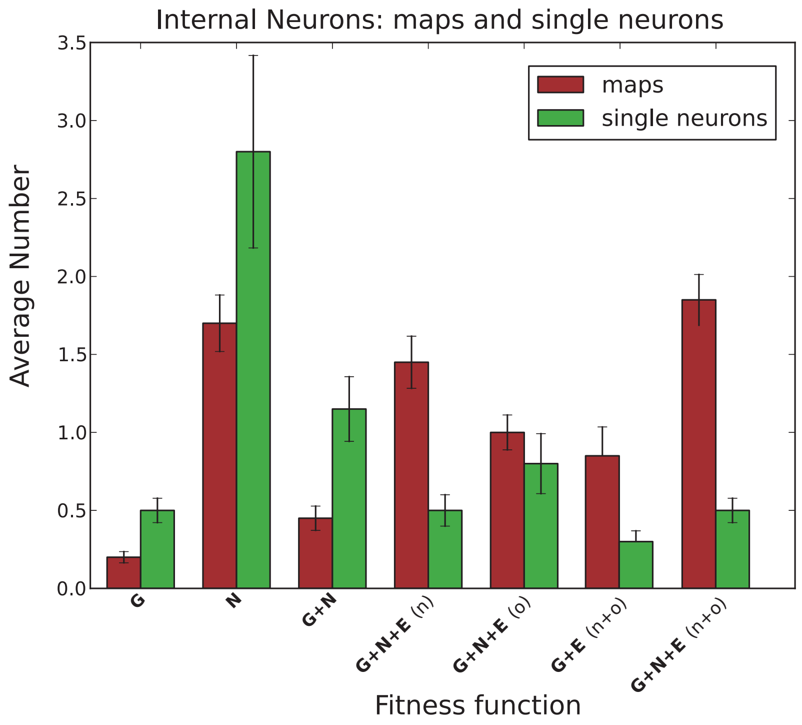


Figure 5.13: Type of internal nodes for the different fitness setups, on the Arena Environment.

with **G**, and **G+N** selective pressures. This can be explained by the fact that for the Void environment with little constraints, the evolutionary process converges very quickly (In less than a hundred generations — see figure 5.6). As the evolutionary process is then stopped upon convergence, there is very little evolutionary time for evolving internal structures. This confirms that there is no need for internal nodes to converge when using **G**, and **G+N** selective pressures.

In other setups, runs require at least 1000 generations to converge, and often require all the 4000 generations. In those runs, some internal nodes appear: in the Void environment with **G+N+E (n+o)** for instance, there is in average of 1 internal node. This does not mean, however, that internal nodes are required for solving the task with these selective pressures. Instead, internal nodes can emerge just because of random drift (and eventually with diversity) during a long evolutionary process (See later tests for more explanations).

Type of internal node Figure 5.13 details the node nature, in average, at generation 4000 for each setup and for the Arena environment.

Globally, the number of single neurons is higher when no environmental pressure **E** is used, and lower otherwise. The environmental pressures then drive towards internal maps (this conclusion will be explained in later results).

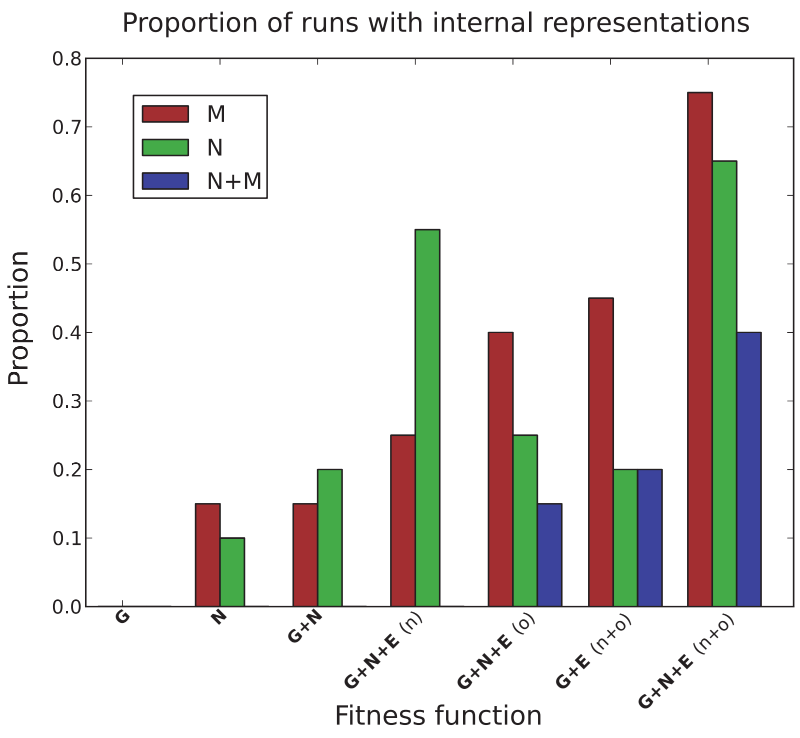


Figure 5.14: Proportion of runs with Internal Representations within the Arena Environment, with different fitness functions

Representations in the different fitness setups The number of internal nodes does not relate all information about the emergence of internal representations. Figure 5.14 displays the proportion of runs, for each selective pressures, that have at least one Internal Representation — on the Arena environment. A run with a **N+M** representation is also included in **N** and **M** categories.

The presence of internal representations of any types is strongly dependent on the selective pressures. No representation at all emerge in the **G** only setup, and only 10−20% of runs result in controllers with representations with the **N** and **G+N** setups. Furthermore, **N+M** representations only emerge in the setups including the environmental pressure **E**.

Unsurprisingly, more **N** representations emerge when subjected to environmental pressure **E (n)**, while more **M** representations emerge when subjected to **E (o)**. In other words, the pressure towards having consistent behaviors when subjected to noise leads to more internal representations with noise resistance, while the pressure towards having consistent behaviors under visual occlusions leads to more internal representations with memory.

When both environmental pressures are present, the proportion of runs with controllers having representations is significantly higher. Table 5.2, 5.3, 5.4, detail the statistical significance of the differences between all setups.

Table 5.2: P-values associated with the presence of **M** internal representations , as displayed on figure 5.14. P-values are calculated using the Mann-Whitney U test. Bold numbers represent statistical significance ($p < 0.05$).

	G	**N**	**GN**	**GNE (n)**	**GNE (o)**	**GNE (no)**	**GE (no)**
G		**0.0403**	**0.0403**	**0.0097**	**0.0010**	**1.2e-6**	**0.0004**
N			0.4480	0.3208	0.0813	**0.0010**	0.0504
GN				0.2541	**0.0450**	**0.0001**	**0.0228**
GNE (n)					0.1275	**0.0004**	0.0649
GNE (o)						**0.0165**	0.3683
GNE (no)							**0.0382**
GE (no)							

Table 5.3: P-values associated with the presence of **N** internal representations , as displayed on figure 5.14. P-values are calculated using the Mann-Whitney U test. Bold numbers represent statistical significance ($p < 0.05$).

	G	**N**	**GN**	**GNE (n)**	**GNE (o)**	**GNE (no)**	**GE (no)**
G		0.0813	**0.0200**	**0.0001**	**0.0099**	**1.0e-6**	**0.0200**
N			0.2498	**0.0052**	0.1584	**0.0013**	0.2498
GN				**0.0117**	0.3148	**0.0023**	0.4923
GNE (n)					0.0509	0.2715	**0.0117**
GNE (o)						**0.0147**	0.3148
GNE (no)							**0.0023**
GE (no)							

Influence of Novelty The novelty objective **N** significantly increases the presence of **N** and **M** representations, when added to goal objective only **G**, as well as goal objective + environmental pressures **G+E (n+o)**. There are more runs resulting in **N+M** representations (8 for **G+N+E**, and 4 for **G+E**). The most successful setup, in terms of representations, is then the **G+N+E (n+o)** setup.

Main result Environmental Pressures (**E (n)**, **E (o)** and **E (n+o)**) significantly increase the proportion of runs with **N**, **M** and **N+M** representations. Using a Novelty objective **N** further increases this proportion.

Influence of the Environment Figure 5.15 showcases the differences between the Arena environment and the Void environment, for the best setup **G+N+E (n+o)**. The proportions are statistically higher for the Arena environment with

Table 5.4: P-values associated with the presence of **N+M** internal representations , as displayed on figure 5.14. P-values are calculated using the Mann-Whitney U test. Bold numbers represent statistical significance ($p < 0.05$).

	G	**N**	**GN**	**GNE (n)**	**GNE (o)**	**GNE (no)**	**GE (no)**
G		x	x	x	**0.0403**	**0.0010**	**0.0200**
N			x	x	**0.0403**	**0.0010**	**0.0200**
GN				x	**0.0403**	**0.0010**	**0.0200**
GNE (n)					**0.0403**	**0.0010**	**0.0200**
GNE (o)						0.0515	0.3563
GNE (no)							0.1027
GE (no)							

Table 5.5: Number of runs with internal representations **M**, **N** and **N+M**, as well as P-value of the difference between the Void and Arena Setup displayed on Figure 5.15.

	# Void	# Arena	P-value
M	9	**15**	**0.01359**
N	6	**13**	**0.0098**
M+N	5	**8**	0.1443

regards to **M** or **N** representations. For **N+M** Representations, the difference is not significant; this is probably due to the small number of runs with representations (the number of runs resulting in **N+M** representations is 8 for Arena, and 5 for the Void environment). Table 5.5 shows statistical significance.

The presence of walls, which makes the task much more difficult, seem to have a positive impact on the emergence of internal representations.

Representations and Generalization In order to assess the importance of these representations, the generalization ability of controllers — with and without representations — is computed. The resulting controllers of the runs of each setup are placed under one of the four categories:

- **N+M**: have at least one **N+M** representation
- **M**: have no **N+M** representation, and at least one **M** representation
- **N**: have none of the above, and at least one **N** representation
- **None**: have none of the above (no representation at all).

In figure 5.16, each run is plotted in the line corresponding to its category, with its generalization ability in the x-axis. The runs resulting in architectures presenting

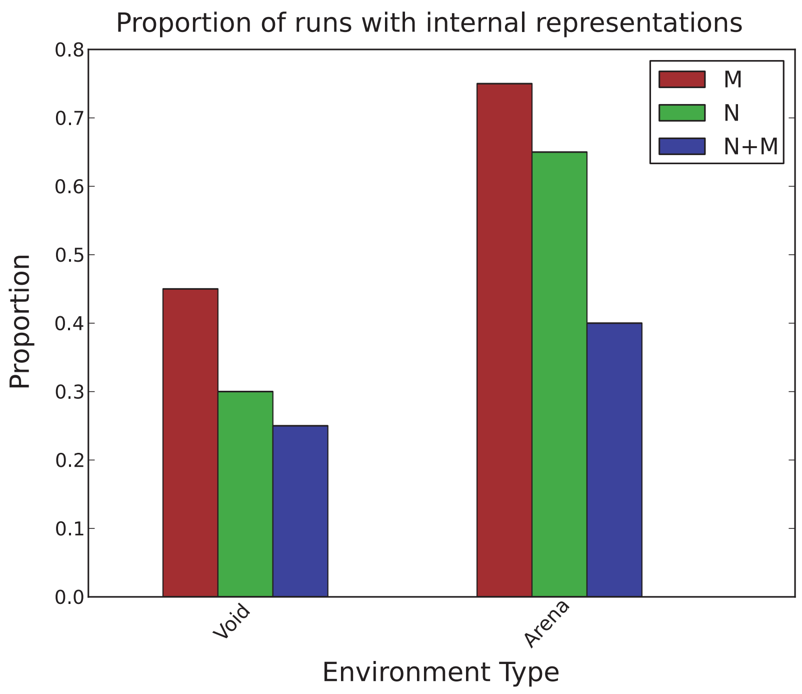


Figure 5.15: Proportion of runs with Internal Representations with the fitness setup **G+N+E (n+o)**

Table 5.6: Point-biserial Pearson Correlation between generalization score and the presence of representations, as displayed on figure 5.16.

M+N Representation	0.93
N or **M** Representation	0.52

at least a **N+M** representation have a 0.56 average generalization score, while the runs resulting in no representation at all have a 0.07 generalization score. The runs having at least a **N** or **M** representation have a 0.29 average generalization score. This shows a strong correlation between the presence of representations and the generalization ability. Table 5.6 details the Pearson's Correlation score between the generalization score and the presence of these representations.

These runs however, have very different selective pressures and thus were not evolved in the same conditions. As the generalization ability is evaluated by testing the controllers with noise and occlusions, runs with environmental pressures **E (n+o)** have a natural advantage in generalization.

In order to reduce this bias, only runs with environmental pressures **E (n+o)** were selected in figure 5.17. This figure shows the exact same tendencies, and the correlations are still very strong (See Table 5.7).

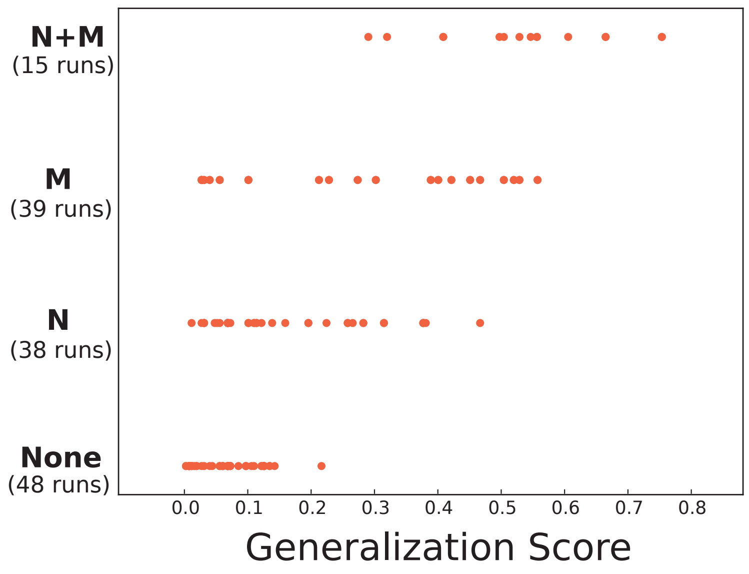


Figure 5.16: Scatter-plot of 140 runs (20 runs for each fitness setup) in the Arena environment. Each run is placed under one of the four categories: 1) has at least one **N+M** representation 2) has at least one **M** representation 3) has at least one **N** representation 4) has none of the above.

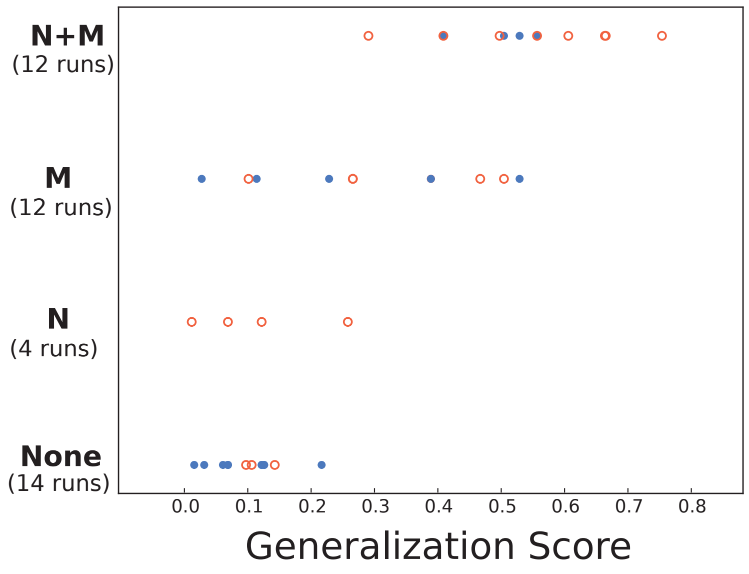


Figure 5.17: Scatter-plot of 40 runs (20 runs for **GNE (n+o)** — red hollow dots, and 20 for **GE (n+o)** —blue full dots) in the Arena environment. Each run is placed under one of the four categories: 1) has at least one **N+M** representation 2) has at least one **M** representation 3) has at least one **N** representation 4) has none of the above.

Table 5.7: Point-biserial Pearson Correlation between generalization score and the presence of representations for the 40 runs with all environmental pressures, as displayed on Figure 5.17.

M+N Representation	0.91
N or **M** Representation	0.62

Conclusion The presence of Internal representations in controllers is strongly correlated to the ability to generalize to contexts unseen during evolution.

5.3.2 Resulting Architectures

Typical Behavior Figure 5.18 and 5.19 display typical behaviors of a controller generated by a **G+N+E (n+o)** run. This controller manages to reach all 6 goals when not influenced by noise nor occlusions, and therefore has a maximal goal objective **G** = 1. The robot also shows good behavior when in noisy conditions: its discretized trajectory is still very close to the one obtained without noise. In figure **??** the controller reacts poorly when its visual input is muted: its trajectory is strongly affected by occlusions. The controller corresponding to trajectory in figure 5.19 has perfect score, even with noise and occlusions — its controller is displayed in Appendix A.3.

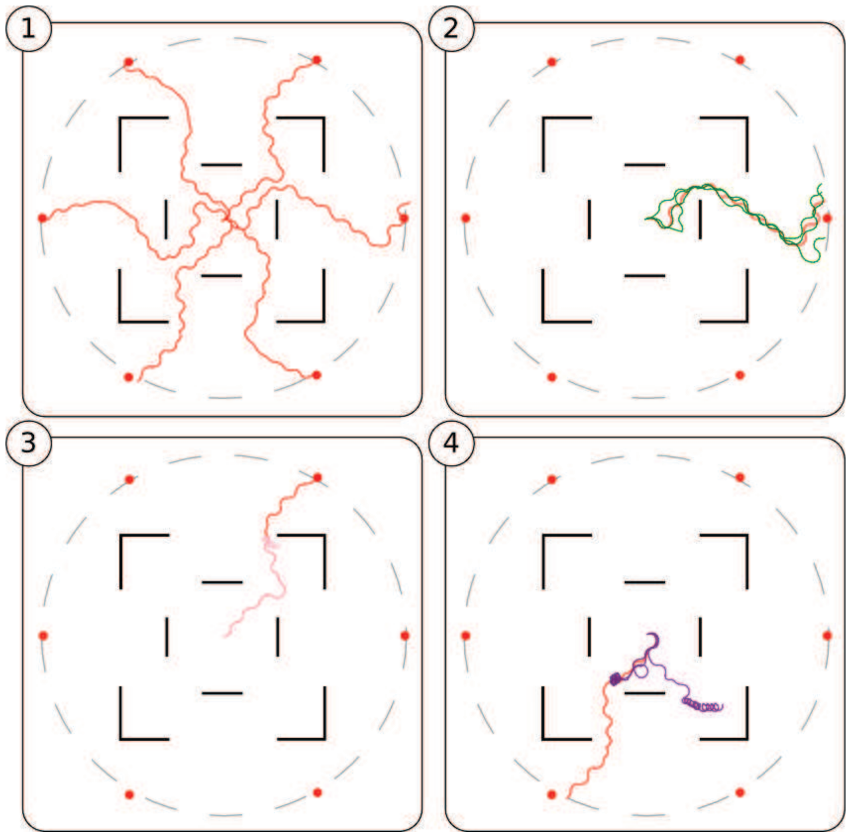


Figure 5.18: Typical behavior of a resulting controller with **G+N+E (n+o)** selective pressures. 1) First 6 trajectories with no noise nor occlusions. 2) (dark green) Trajectories in noisy conditions. 3) (pink) Trajectories when subjected to occlusions after 50 time-steps. 4) (purple) Trajectories when subjected noise and occlusions after 50 time-steps.

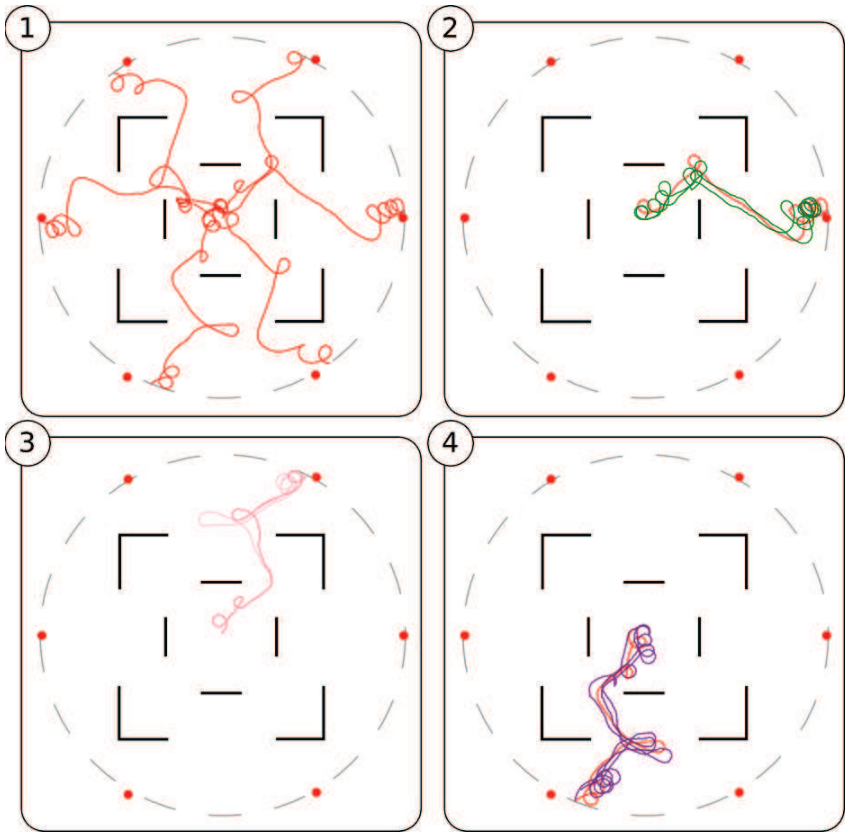


Figure 5.19: Typical behavior of a resulting controller with **G+N+E (n+o)** selective pressures. 1) First 6 trajectories with no noise nor occlusions. 2) (dark green) Trajectories in noisy conditions. 3) (pink) Trajectories when subjected to occlusions after 50, 100 and 150 time-steps. 4) (purple) Trajectories when subjected noise and occlusions after 50 time-steps.

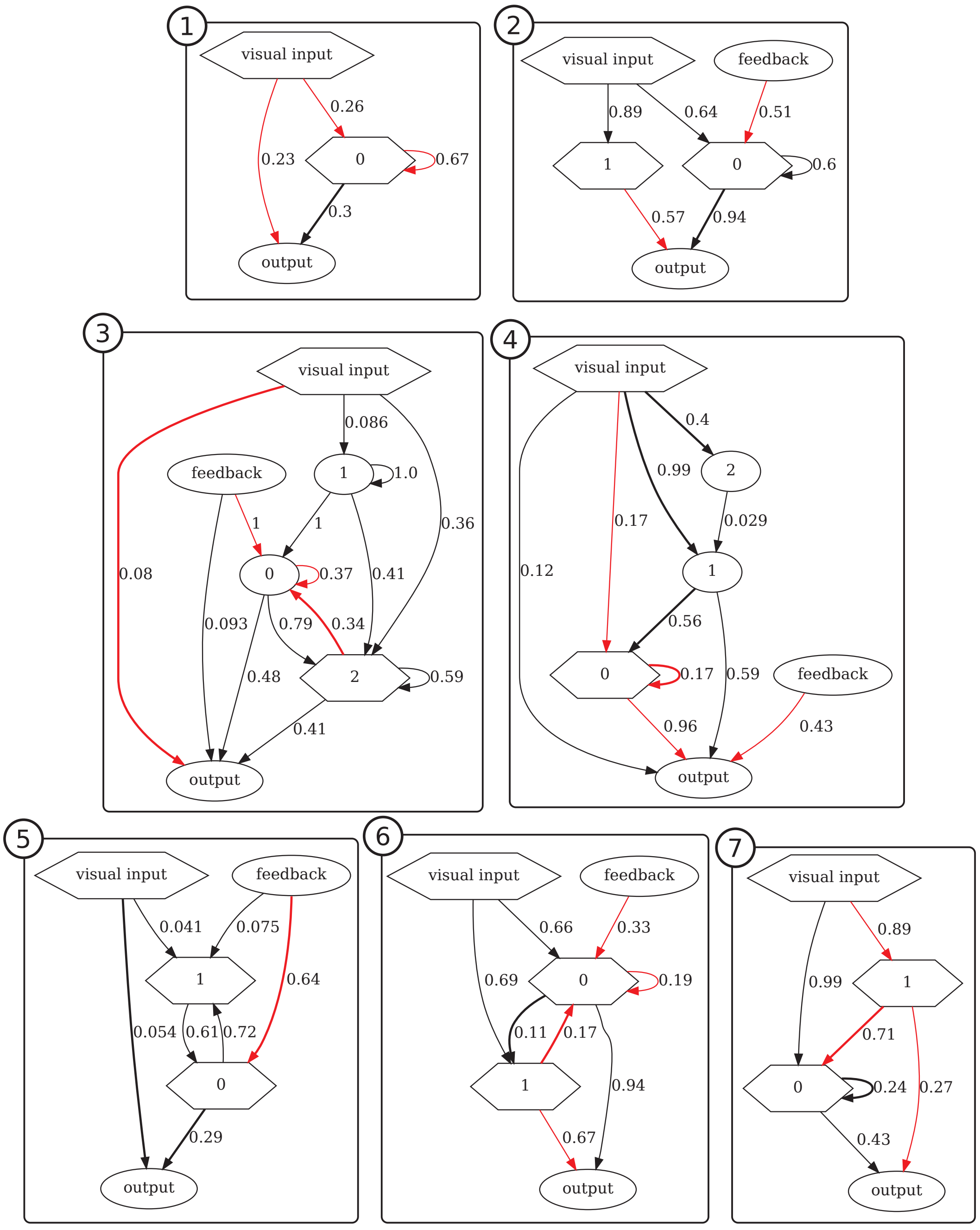


Figure 5.20: Parts of neural networks connected to the visual input, obtained at generation 4000 in **G+N+E (n+o)** runs. Numbers on the edges represent the connection weights, and red arrows are inhibitory connections. Ellipses are single neurons, and hexagons are maps of 20 neurons. Bold arrows represent Gaussian connections (the associated number corresponds then to σ of the Gaussian). Biases and offsets are not represented. 1) present a **N** Representation, 2) a **M** Representation, and all others at least one **N+M** Representation.

While most runs converge to maximal goal objective **G**, the consistency objective **E** does not always converge to its maximal value. This is especially true for the **E (o)** and **E (n+o)** as occlusions strongly affect the behavior, and few controllers are unaffected by the loss of vision. Still, the selective pressures **E (o)** and **E (n+o)** have a strong influence on the whole population as it increases the emergence of internal representations.

Internal structures The neural networks presenting internal representations have various architectures. Figure 5.20 displays several neural structures (partial neural networks, not including lasers, nor the acceleration output). Those structures are resulting from the **G+N+E (n+o)** setup. It is noticeable that none of these structures are feed-forward networks: they all have at least one recurrent connection.

Many networks include many more connections with the lasers and eventually other internal neurons. The analysis of the behavior of these networks show no easily recognizable internal activity, such as the activity of a sub-part of a Head Direction model. In particular, no internal map could effectively reproduce the input when the input completely disappears. Instead the internal maps were able to encode *some* information, which enabled the robots to have consistent behavior, but not easily understandable by the experimenter. Two profiles of such map activity can be found in Appendix A.4.

In the evolutionary experiment by Gigliotta et al. (2011), the authors describe the emergence of predictive models when the robot is subjected to occlusions. This means that an internal map of neurons is able to reproduce the missing input. The neural architecture able to realize this predictive model was however *ad hoc* and not modified through evolution. In the present experiment such an internal structure did not appear spontaneously, even though it would be very effective to cope with occlusions and noise. However, it is not known whether such a model can be easily built with EvoNeuro2 building blocks.

Connectivity Pattern Although the internal behaviors are difficult to understand from a human point of view, a few networks showed specific connectivity patterns between internal maps. Figure 5.21 displays a sub-part of an evolved neural network, in which both maps are considered as **N+M** Representations. The two maps are connected with Gaussian connections, each with a different σ, one inhibitory and one excitatory. These two maps compute a difference of Gaussian (DoG). As explained in figure 5.22, this DoG have similarities with a "Mexican-hat" connectivity (figure 5.22, right) in a single circular map (figure 5.22, left) — if the time-steps necessary to transmit the activity from one map to the other is neglected. The "Mexican-hat" connectivity is well known in computational neuroscience because of its two components:

- A strong local excitation: A strong activity in the map results in a self-excitation. Thus information is self-maintained in the map.

- A weak global inhibition: A strong activity in the map inhibits other activities. Thus competing activity or noise are lowered in the map.

Such a connectivity may lead to memory and noise resistance in the internal maps: the local excitation can maintain information and thus provide memory, while the inhibition can result in lowering the noise effects. Among the controllers found with **G+N+E (n+o)**, two other networks had similar connectivity as the one displayed in figure 5.21. Other motifs did not always reduce to "Mexican-hat" connectivity patterns. However, all networks including a **M** representation included at least one positive recurrent pathway.

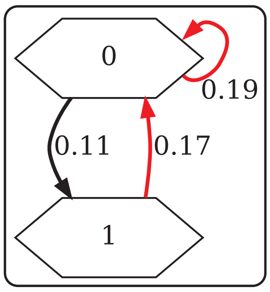


Figure 5.21: Motif found in a network with representation **N+M** (See network 6 of figure 5.20). Both connections are Gaussian, with different sigma.

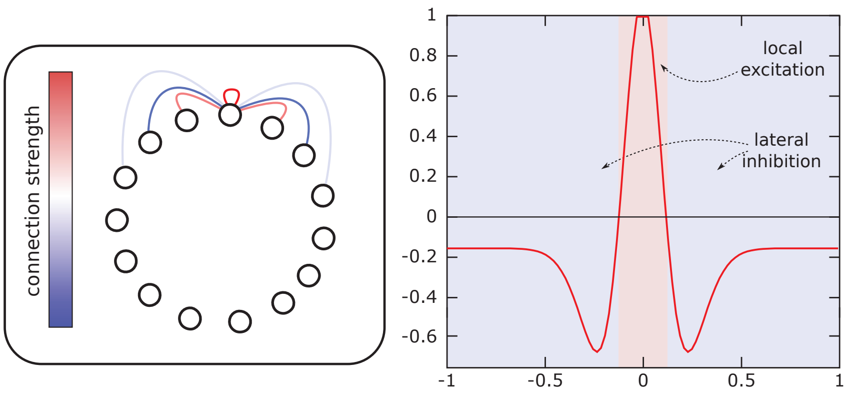


Figure 5.22: (Left) Example of a circular map with "Mexican-hat" lateral connectivity, similar to the ones found in Head Direction models (McNaughton et al., 2006). (Right) Lateral Connectivity of the map, corresponding approximately to the connections displayed in figure 5.21. The difference of Gaussian creates a lateral inhibition and a local excitation in the maps, which realize a "Mexican-hat" shaped connectivity.

5.3.3 Lineage and selection for Internal Representation

The genealogy of a successful solution provides information on how and when the representations emerged during the evolutionary process. The principle is the fol-

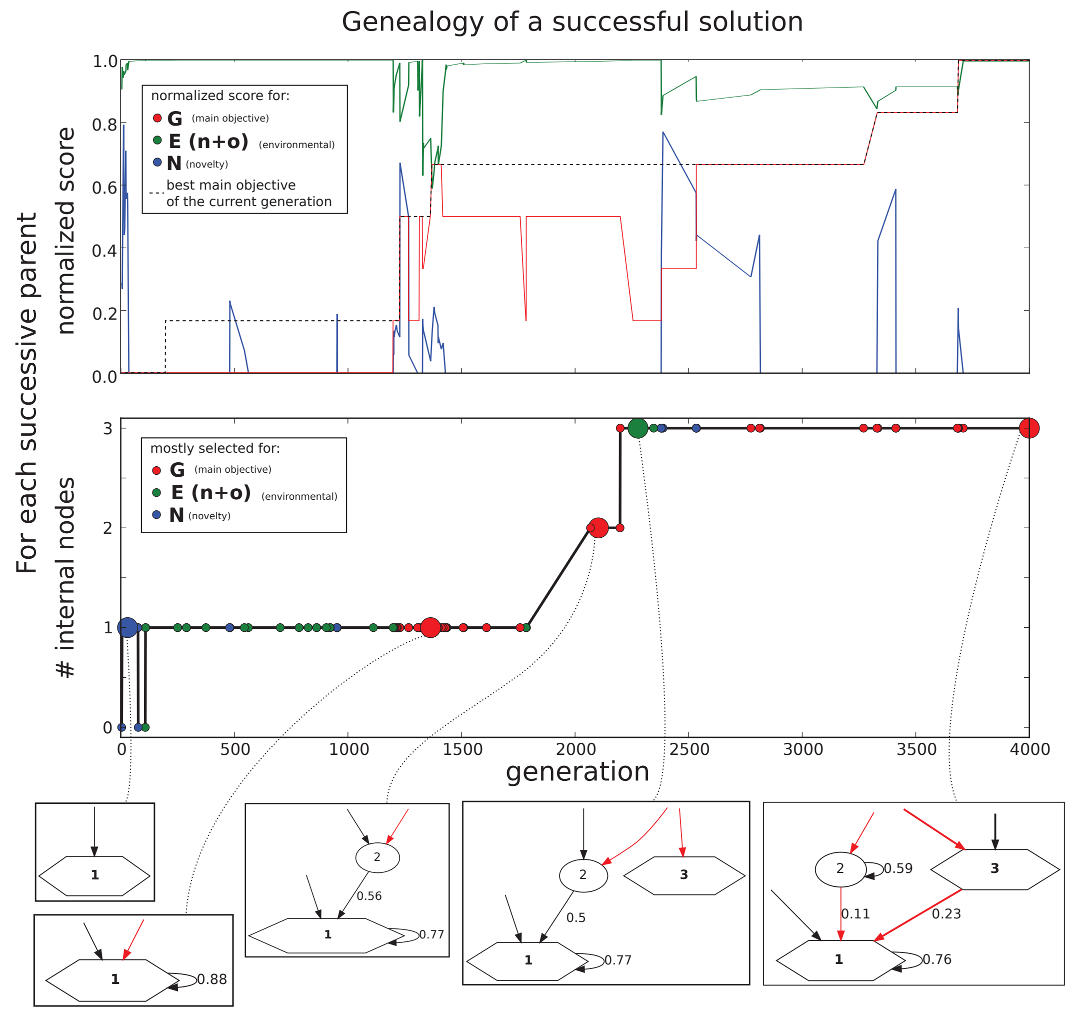


Figure 5.23: Genealogy of a successful solution obtained at generation 4000 with **G+N+E (n+o)**. Each of the parents of the successful solution is represented as a dot in the middle graph. (Top graph) Objective scores for the parents. The black dotted line represent the best main objective of the current generation. (Middle graph) Number of internal nodes for each of the successive parents. Each dot is colored in function of which objective it was most selected for (see text for further explanations). (Bottom) A few selected neural networks associated with the some of the parents. Only the internal nodes are represented.

lowing: 1) a successful run is selected among the runs with **G+N+E (n+o)** 2) The parent of the successful solution (obtained at generation 4000) is the controller from which the solution evolved, through one or more mutations. The successive parents are sought and displayed as dots on figure 5.23 (Middle graph), where the Y-axis represents the number of internal nodes of the controllers. Each of the parent score in the three objectives **G**, **N** and **E** is represented on the top graph. The dashed line represents the best individual in terms of goal objective **G** of each generation.

Discontinuous evolution The first noticeable aspect is the very discontinuous nature of the evolutionary process: the parent of a solution might be a controller found 500 generations earlier. This means that the controller has survived, unmodified, among the successive populations, during 500 generations . This is mainly due to the two following reasons:

- The mutations happen in the graph space of neural networks, and thus many mutations either strongly damage the behavior, or do not affect it at all.
- Due to elitism and the discrete aspect of the fitness landscape, there are many solutions with similar behaviors in the population. Their novelty objective **N** then falls to 0, but the other objectives maintain them in the populations.

On the other hand, some solutions have very close successive parents, only separated by one generation. This is usually linked with bursts of novelty objective **N**. This means that the population is rapidly moving towards a new area in the search space.

Relative influence of objectives The color of the dots in middle graph represents an approximation of which objective the parent was most selected for. To do so, each parent is ranked in the population of its current generation based on each objective. For instance, a parent at generation 2380 is ranked in its population (the lower the rank, the better; rank 1 is the best individual of the population with regards to the corresponding objective):

- **G**: 41/200
- **N**: 21/200
- **E**: 188/200

Therefore this parent was most selected for its novelty objective, and is displayed as blue in the middle graph. The early parents of the successful solutions (left part of the graphs) were mostly selected for **N** and **E**. Early generations show a very high score in the **E**, implying that the solutions have very consistent behavior under noise and/or occlusions. However, this is because the solutions have very simple behaviors (some are standing still), and are thus not completing the task at all (**G** = 0). As the solutions are selected for **G**, around generation 1300, the consistency score quickly drops. Later stages imply all three objectives, and then mostly the goal objective **G** for the last 1200 generations.

An analysis on 10 runs with **G+N+E (n+o)** show that the best solutions have an average of **101** $\pm$ 52 ancestors. These ancestors were selected because they were better on one or several objective than the others. On average, the ancestors were selected mostly because of:

- **G**: 53%
- **N**: 37%
- **E**: 10%

This shows that throughout the generations, all three objectives have strong impact on the evolutionary descent. The Goal and Novelty objective seem to have a stronger impact than the Environmental objective on the selection.

Network evolution The lower part of figure 5.23 displays the structural changes in the successive parents of the population. The bottom graph displays only the internal nodes rather than the whole neural network. There were very few structural changes throughout the genealogy of the successful solutions. Many of the mutations were only weight changes of little amplitude. Very early, an internal map appeared in the lineage. After 1300 generations, a recurrent connection was included (the map is then a **M** representation in that case). This representation remains almost unchanged over the following generations. Interestingly, the controller at 1300 having the representation was selected mostly because of goal objective **G**, and not the environmental selective pressure **E** objective. However, the **E** objective had an impact on earlier parents.

This graph represents the genealogy of a single run, and as the evolution is stochastic process with many random artifacts, the exact transitions are different for other runs. For instance, some runs display no evolutionary activity (flat graphs) for over 2000 generations due to the challenging nature of the search space. However, this gives a general tendency of the impact of the different objectives, which can be seen in any successful run. Appendix A.2 show an additional lineage graph.

5.4 Conclusion

Through the CMERP, the influence of several parameters on the emergence of internal representations has been shown:

- The presence of helper objectives **N** and **E** significantly increases the emergence of **M** and **N** representations.
- **E (n)** favored most **N** representations, **E (o)** **M** representations and **E (n+o)** all kinds of representations.
- There were globally more representations in the Arena environment than in the Void environment.

Furthermore, the presence of representations, both **N+M** or any of **N** or **M** is strongly correlated to the ability to generalize well in other contexts. The controllers found had recurrent connections, allowing them to resist noise, and have a form of memory (through positive recurrent feedback). A few controllers found have common points with “Mexican-hat” connectivity found in computational neuroscience models of Head Direction and neural fields. Last, the analysis of the genealogy of the representations on a single run with all three pressures **G**, **N**, and **E** show that all pressures contribute to the selection at different generations, and only a few structural changes occur during the lineage of controllers.

Predictive models Resisting to noise and occlusions can be achieved through a *predictive model* (Gigliotta et al., 2011): when the visual input is distorted by noise or muted by occlusions, the model generates a replacement of the input by *predicting* what the robot would see. Using selective pressures such as occlusions (blind phases), Gigliotta et al. (2011) are able to evolve two kind of solutions:

- Individuals which are able to solve the task even during blind phases, but with a modified behavior.
- Individuals which are able to solve the task during blind and normal phase with a similar behavior.

In the last category, the individuals are able to build an internal predictive model which predicts the next visual inputs when they are missing. Such predictive models could be very useful to solve the task presented in this chapter: they would resist to occlusions and eventually to noise. However, such a model did not emerge spontaneously during the experiments reported here:

- There are behavior changes (at least a slight change in trajectory) between the scenarios with and without occlusions.
- No internal map activity is similar to a prediction of the input.

Still, the networks with **N+M** representations were able to store *enough* information to have little variation when subjected to noise and occlusions.

Chapter 6

Discussion and Perspectives

6.1 Contributions

The aim of this thesis is to study the emergence of internal representations in evolutionary robotics. Three points are considered in this work:

- the definition of internal representations in an evolutionary robotics context, and the definition of a protocol to test for these representations;
- the proposal of a new method for adding constraints (seen as *selective pressures*) on the evolutionary process;
- the study of the emergence of internal representations in a simulated environment, under the influence of different selective pressures.

6.1.1 Definition of Internal Representations

Chapter 3 shows the wide use of the term "internal representation" in the cognitive science literature. This thesis focuses on how representations are used in Evolutionary Robotics, as well as briefly introducing their use in Computational Neuroscience models. The chapter defines a quantitative way to determine whether a given neural network has an internal representation or not.

Confronted to the different ways to qualify representations, the present work focuses on representations with specific functional properties:

- **N** Representations (resistance to noise): even if the input of the neural network is subjected to noise, the representation does not vary too much.
- **M** Representations (ability to memorize): even if the input of the neural network is muted, the representation "remembers" previous values.

These representations are associated with a test protocol, which can be conducted on any neural network. The protocol relies on the activity of the internal nodes when the network is subjected to different conditions (noisy conditions, loss of input, etc.). The protocol is tested on classic neural networks, and is able to discriminate networks with or without representations. A test on an evolutionary experiment —"Ball Collecting Task", shows that none of these representations spontaneously appear.

The **M** Representation test was used as the basis of a new *helper objective* in a T-maze memory experiment (Chapter 2). The main result is that solutions with memory were successfully generated by the evolutionary process. The memory helper objective, alongside a diversity objective has generated solutions with a robust memory. Interestingly, these helper objectives promoted solutions which generalize well to previously unseen contexts, even though they were not rewarded for their generalization ability.

6.1.2 Behavioral Consistency Method

Chapter 4 covers the definition of a method for creating a Behavioral Consistency Objective (BCO). The objective can be used as a single fitness function, or as an additional selective pressure alongside other objectives. The principle of the BC method is to add soft constraints on the behavior of the evolved system. Instead of directly rewarding a particular "target" behavior, the objective rewards individuals with a consistent behavior over predefined scenarios.

This method is applied to two experiments related to computational neuroscience. The original experiments aim at optimizing computational neuroscience models (a visual attention model, and an action section model) so that their behavior fits as well as possible a target behavior. In contrast, the BCO does not require the exact knowledge of a target behavior. Instead, functional constraints on the expected behavior are defined. Using the BCO in both experiments, successful solutions are evolved, and the solutions show a greater diversity of behaviors.

The Behavioral Consistency Method does not completely remove *a priori* knowledge on the target behaviors of systems. Instead, it shifts the required knowledge from *an exact description of a target behavior or parts of it*, to *a description of scenarios and how the behavior should compare in these scenarios*.

Additionally, the BCO can have several important advantages over a standard evaluation on the same scenarios —these advantages are detailed in the next section 6.2.2. These advantages are confirmed in the experiment in Chapter 5: the use of this objective strongly improved convergence results.

6.1.3 Emergence of Internal Representations under multiple selective pressures

Chapter 5 proposes a new ER protocol for studying the emergence of internal representations. The neural network of a robot controller is evolved (both the topology and the parameters of the network) to solve a navigation task in a circular arena. Two environmental selective pressures — noise and occlusions — are added to the selection process. The task itself with the goal objective only is not *representation hungry*, but representations become necessary when adding the environmental selective pressures.

The main result is that the number of internal representations significantly increases in presence of environmental selective pressures, even though the networks were not rewarded nor trained for having internal representations. The novelty helper objective further increases the number of runs in which the emergence occurs.

As expected, the noisy pressure promotes more internal **N** representations, while the occlusion pressure promotes the **M** representations. Using all selective pressures, 9 out of 20 runs could achieve **N+M** representations. In addition, the presence of a representation is strongly correlated with a higher generalization ability.

The analysis of the resulting architectures show the following points:

- As expected, the networks with representations have at least a recurrent connection, they are not feed-forward, reactive networks.
- The behavior of the internal maps is not easily understandable by a human being, and none correspond to a predictive model of the input.
- A few networks had a connectivity closely related to a difference of Gaussian functions, which can be found in Computational Neuroscience networks (often referred as a “Mexican-hat” connectivity).

The study of the lineage of the best individuals reveals interesting information. The evolutionary process is very discontinuous, and a good solution only has a limited number of ancestors, sometimes separated by thousands of generations. Furthermore, it revealed the relative importance of the different objectives throughout the evolutionary process, as well as their influence at different generations. All three objectives (Goal objective, Novelty, and Environmental pressures) had a significant influence on the selection process. The Goal and Novelty Objective were the objectives influencing the selection most. However, the Environmental objective had a very significant impact on the evolutionary process as a whole, even though it wasn’t the main influence. This technique could be useful to monitor other multi-objective evolutionary experiments.

Last, the evolutionary process generated successful controllers in a difficult ER task, which involve navigation through a (simple) continuous maze. The visual input is a continuous stimulus, —which is subjected to noise and occlusion. Depending on this stimulus, the robot must navigate towards different parts of the maze. The task is therefore more challenging than most navigation tasks reviewed by Nelson et al. (2009).

6.2 Discussion and Perspectives

This thesis opens several questions on the methods used:

- the classification of representations;
- the implications of the use of the Behavioral Consistency Method.

Additionally it brings several perspectives on closely related topics:

- emergence of Predictive Models;
- new Selective Pressures;
- neural Network Encodings.

6.2.1 Classification of representations and tests

The present definition of representations only account for very specific representations (**N** and **M**), not any kind of representations. However, these representations seem frequent in computational neuroscience models, and the presence of noise, or the necessity to have some memory are common in evolutionary robotics experiments (See Chapter 2 and 3 for more details). This opens the question of a further classification of representations, with additional functional properties of representations. A few leads might include:

- Representations which resist to distractors (or competing stimuli), which can be found in Neural Fields or Action Selection.
- A more continuous characterization of Memory representations: A Memory Representation may be able to store one bit of information (such as the one studied in the T-maze experiment), ore more. Using information theory, a better characterization of M representations might be possible.

Furthermore, the present definition of **N** and **M** representations require parameters (thresholds), and definition of scenarios. In order to have a more general definition, as well as making the protocol easier to use, the following approach could be considered:

- the use of information theory for a more general comparison of internal activity. For instance, the quantification of representation by Marstaller et al. (2012) uses mutual information to determine the level of representation included in an internal node. The use of such methods could remove the need for parameters such as thresholds.

Last, the representation tests where used on a new evolutionary protocol, the CMERP. Further work includes the use of the tests on the evolutionary robotics experiments from the literature. The robotic experiment defined by Gigliotta et al. (2011), or the minimally cognitive behaviors defined by (Beer, 1996) could then provide a framework for the study of the emergence of internal models in natural and artificial agents.

6.2.2 Behavioral Consistency Method versus standard evaluation

Chapter 4 and 5 show a successful use of BC method for applying selective pressures as helper objectives. However, in the case of environmental pressures, such as noise and occlusions, the selective pressure could have been applied in a more straightforward way (denoted **standard evaluation**):

- By evaluating the performance in terms of goal objective under scenarios with the noisy and vision deprived scenarios;
- by aggregating all the different scores into a global fitness score. The most intuitive way to aggregate the scores by averaging the fitness scores.

The standard evaluation is the most simple and biologically plausible way to apply the selective pressure. Unlike the BC method, it does not add any *a priori* knowledge on which part of the behavior should resist to noise/occlusions. Using such a fitness function would improve the strength of the results of chapter 5: the presence of noise and occlusion directly in the evaluation of a navigation task should lead to the emergence of internal representations in the controller architecture.

However, the use of such a standard evaluation yields much worse results: little convergence and less representations. The comparison between the two methods was not fully explored in this thesis, however preliminary comparison results can be found in Appendix A.5. This could be explained by the following points, each of them might have a large influence on the evolutionary process:

- **Stochastic effects in evaluation** By directly evaluating scenarios with noise, the noisy simulations can lead to wrongly positive evaluations. For instance, a single scenario can be successful in one instance of noise, but by running the simulation again, the solution might be almost always unsuccessful. Such a solution, very sensitive to noise, will however be selected by the standard evolutionary process. The BC method is less influenced by this problem, because only solutions with *little* noise effect on the behavior are chosen.

- **Evolutionary gradients** The BC objective and a standard evaluation reward different individuals in the population. The latter only rewards successful solutions (those which reach the goal), while the BCO can reward individuals which are very weak regarding the goal objective (they may not reach any goal), but have consistent behavior in the different scenarios. These solutions could be important stepping stones towards solutions which reach goals and have consistent behaviors. The evolutionary "path" towards the best solutions in the fitness landscape is different and might lead to the exploration of more diverse solutions with the BCO than with the standard evaluation scheme.

- **Computational cost** The standard evaluation of noisy scenarios, vision deprived scenarios, etc. can lead to an exponential number of evaluations (evaluate in many different instances of noises, for each possible goal point, for full vision or altered vision, etc...). In the preliminary tests with full evaluation, the number of scenarios evaluated to cover all different cases was 264 (see Appendix A.5). The BC method only requires few additional scenarios (a total of 15 scenarios evaluated), and still had better results.

At first glance the standard method seems more biologically plausible method to apply evolutionary pressures than the BCO. However, the intuition is that artificial evolution is not able to generate these constraints using standard evaluation, because it would require too complex and numerous evaluations. Furthermore, even with many evaluations, the aggregation of the evaluations might not drive efficiently the evolutionary search (for instance, averaging the partial fitness scores is very different from natural selection processes). The BCO can be seen as an efficient way to aggregate evaluations, and induce constraints on the agent behavior that can be similar to constraints seen natural systems.

6.2.3 Emergence of Internal Models

This thesis details the emergence of specific representations, which may be considered as a simplified model of the robot's perceptions. It does not include several possible actions — and thus the need of action selection — or a spatial representation of the world, etc.

This thesis is therefore a small step towards the emergence of more complex capacities and internal models. However, the challenge lies in the scaling of the capacities: how can more complex capacities — and thus potentially models — emerge ? How can several capacities emerge in a single robot ?

Many types of complex internal models have been studied in Cognitive Science (Johnson-Laird, 1980), and some of the most studied are the models for spatial navigation. They include spatial representations, as well as internal functionalities, such as self localizing, path planning, path integration, etc. (Trullier et al., 1997). Section 3.1.3 details some models involved in cognitive abilities, and more specifically, section 3.1.3.2 details spatial navigation models.

Several works have already tackled the emergence of more detailed internal models in evolutionary robotics:

- Nolfi (2005) developed a categorization experiment in which the agents discovers the categories itself.
- In Gigliotta and Nolfi (2008), a robot navigates in a maze, and creates its own representation of space.
- Gigliotta et al. (2010, 2011) work on a robot which follows a visual input, and learns to predict this input.

However, all these approaches are based on ad hoc networks, with built-in neurons for specific representations, and models. In the first two papers, the fitness also explicitly rewards the presence of representations. A long term goal following this work, is the automatic design of an autonomous agent with some cognitive capacities close to an animal. It can be hypothesized that internal models are necessary for an artificial agent to develop such capacities. However, in embodied views of cognition, such an hypothesis is not made, and therefore the models may emerge only if they are needed to solve the tasks. A possible lead would be to study the emergence of internal models, —without specifying the internal structure of the neural network— such that the evolutionary process discovers the model itself.

Furthermore, in order to increase the complexity of emerging internal models, the following possibilities are proposed:

- An approach based on exaptation. Exaptation (Gould and Vrba, 1982) refers to a shift in the function of a trait during natural evolution. Mouret and Doncieux (2009a) take inspiration from this process and define a helper objective that rewards the presence of a sub-network realizing a given sub-function. This concept could be applied to reward controllers in which a sub-part of the controller realize a cognitive function, such as action selection, or any other. The behavioral consistency method, with scenarios similar to those described in Chapter 4 could then be used to reward controllers with a form of action selection.

The work following this thesis could lead to results in two different domains:

- **engineering:** taking advantage of the emerging internal models for better robotic control: one can expect the controllers with internal models to generalize better, to have better properties in uncertain conditions, etc.

- **biology:** study the emergence of internal models and gain insight on biological phenomena, through artificial experiments (similarly to the experiments by Gigliotta et al. (2011)). Biologists have studied the evolution of cognition (Shettleworth, 2009; Platek, 2009). While these works provide insights on the origins of cognitive aspects, they often rely on the study of observable parameters, such as the size of the skull of animals. The origins of cognitive functions is therefore very difficult to study, and remain open questions today. Synthetic approaches such as evolutionary robotics may help these biological studies.

6.2.4 New Selective Pressures

Other selective pressures can be considered in order to promote the emergence of internal models in neural networks. Inspiration may be drawn from natural selective pressures which led to the neural networks currently observed on biological organisms.

Variability of the environment Körding et al. (2006) emphasized that the variations in environment and noise could be one of the reasons for the need of internal models for more robust motor control. The selective pressures proposed in Chapter 5 are already inspired from these real world conditions: the presence of noise, and the unreliability of information which can lead to occlusions in sensors.

This can be generalized in the following way: the variability of the world can be considered as different selective pressures. In natural environments, individuals are evaluated in very different conditions (different environments, different weather, different interactions with others, etc.). In particular a selective pressure can be derived from the ability to be efficient in many different contexts. In other words, the ability to generalize to other contexts (slightly different from the ones used for the main evaluation of the task) can be seen as a selective pressure.

Pinville et al. (2011) developed a method named ProGAb for promoting the generalization ability of ER experiments. This method can be applied as an additional objective, and computes an approximation of the generalization ability of individuals. In a T-maze experiment similar to the one presented in the first part of chapter 4, Pinville (2013) uses this objective as a selective pressure, and notices an increasing number of emerging of memory in the neural networks. Furthermore, the results of chapter 5 show a strong correlation between generalization ability and the emergence of internal representations.

This leads to the belief that a selective pressure promoting the generalization ability would have a significant effect on the emergence of internal representations.

Multiple selective pressures and NSGA-II This thesis advocates the addition of several selective pressures as objectives in ER experiments. However, as studied in Section 2.1.3.5, the performance in terms of selection of NSGA-II quickly drops when the number of objective is larger than 3 or 4. The use of many selective pressures might then require specific MOEAs which can deal with a large number of objectives (Ishibuchi et al., 2008).

Another worthy point is the fact that all objectives are considered as having the same importance. One could, for instance, give more weight to a specific objective: the goal objective could be of greater importance than a diversity objective and therefore have a more important role in the selection process. To do so, Clune et al. (2013) use an additional parameter for each objective: a probability p_i. p_i represents the probability that the objective i is taken into account in the selection process. If $\forall i$, $p_i = 1$, this corresponds to the traditional NSGA-II selection process. A lower probability means that an objective will have a lower impact in the selection process.

6.2.5 Neural Network Encodings

This thesis focuses on the *selective pressures* which drives the evolutionary search towards controllers with internal representations. This is a first step towards the evolution of cognitive abilities. However, the *encoding* of the controller is also of critical importance; and a direct encoding of the neural network parameters is expected to quickly reach its limit.

This can be illustrated by looking at the biological evolution of brains: brains can contain billions of neurons, and each of these neurons is not encoded by independent genes. Instead, the brain undergoes a development phase, in which the same genes are reused and determine, along with the environment, the biological parameters of many neurons. This shows that the encoding, which maps the genome into a fully developed neural network has a strong impact on the evolution of large and complex neural networks.

To address the challenge of the evolution of more complex agents, a substantial part of the ER community has focused on indirect encodings. EvoNeuro and EvoNeuro2 used in this thesis are already examples of indirect encodings from which larger neural networks can emerge. Section 2.3.4.2 details a few other indirect encodings such as HyperNEAT (Stanley, 2007; Stanley et al., 2009).

The intuition is that both *selective pressures* and *indirect encodings* will be major factors for the evolution of complex and cognitive artificial agents. This could be illustrated by a fitness landscape of an ER thought experiment. On one hand, the *encoding* modifies the geometry of this fitness landscape, and contracts it: the number of mutations required to move from a point to another is likely to become much lower with an appropriate encoding (but could also be larger). On the other hand, the *selective pressures* select promising mutations — not necessarily good in terms of performance, but which are good stepping stones for later solutions — and prevent to fall into local optima. In order to obtain controllers with complex and multiple cognitive abilities, it is likely that both *selective pressures* and *indirect encodings* have to be studied simultaneously. In (Mouret and Doncieux, 2009a), the authors show that using a specific encoding and specific selective pressures increase their results, but using both at the same time is significantly better than using only one of them.

Chapter 7

Conclusion

This thesis aims at studying the emergence of internal representations in Evolutionary Robotics, in the light of selective pressures. Through an ER experiment in Chapter 5, an hypothesis is tested: do environmental pressures such as the presence of *noise* and *occlusions* drive the evolutionary search towards internal representations ? To address this question, several tools and methods are defined.

In Chapter 2, a necessary background is introduced. The chapter details the state-of-the-art in Evolutionary Algorithms, in particular Multi-Objective Evolutionary Algorithms, and their use in Evolutionary Robotics. It also covers several recent advances in Evolutionary Robotics, such as the exploration of the behavioral space. Last, neural networks and their optimization through evolution —with direct or indirect encodings— is mentioned.

Chapter 3 reviews the uses of the term "internal representation" in Cognitive Science and Evolutionary Robotics, and then defines the scope of these representations in this thesis. The focus is made on **M** and **N** representations, which correspond to internal nodes which have functional properties (**M** — presence of memory and **N** — resistance to noise). A protocol to test whether a network has such a representation is defined.

Chapter 4 covers the definition of a new method for creating a Behavioral Consistency Objective (BCO). Its efficiency is asserted on two test problems taken from Computational Neuroscience. The objective can be used as a single fitness function, or as an additional *selective pressure* alongside other objectives. These *helper objectives* could be applied to many ER experiments, in order to enhance some properties of the evolved controllers, such as their robustness to noise.

Finally, Chapter 5 is the core of the thesis: using the test protocol defined in Chapter 3, and adding environmental selective pressures with BCOs, the hypothesis is confirmed: environmental pressures such as the presence *noise* and *occlusions* strongly increase the emergence of internal representations. Additionally, the Novelty objective, which promotes novel behaviors, further increases the emergence of these representations. The representations are useful for motor control: the presence of a representation is strongly correlated with the ability to generalize well to unseen contexts. The resulting networks show no sign of understandable internal models, but several networks have connectivity patterns which strongly resemble to a "Mexican-hat" connectivity, which is commonly seen in Computational Neuroscience models. Last, the evolutionary process generated successful controllers in a challenging Evolutionary Robotics task.

Appendix A

Appendix

A.1 DNN and EvoNeuro2 Parameters

- MOEA: NSGA-II (pop. size : 200)
- DNN (direct encoding) or EvoNeuro2 (indirect encoding):
  - prob. of changing weight/bias: 0.1
  - prob. of adding/deleting a conn.: 0.15/0.25
  - prob. of changing a conn.: 0.1
  - prob. of adding/deleting a neuron: 0.025/0.025
  - activation function for neurons:
    $y_i = \varphi\left(\sum_j w_{ij} x_j\right)$ where $\varphi(x) = \frac{1}{1+\exp(b-kx)}$

A.2 Additional Lineage

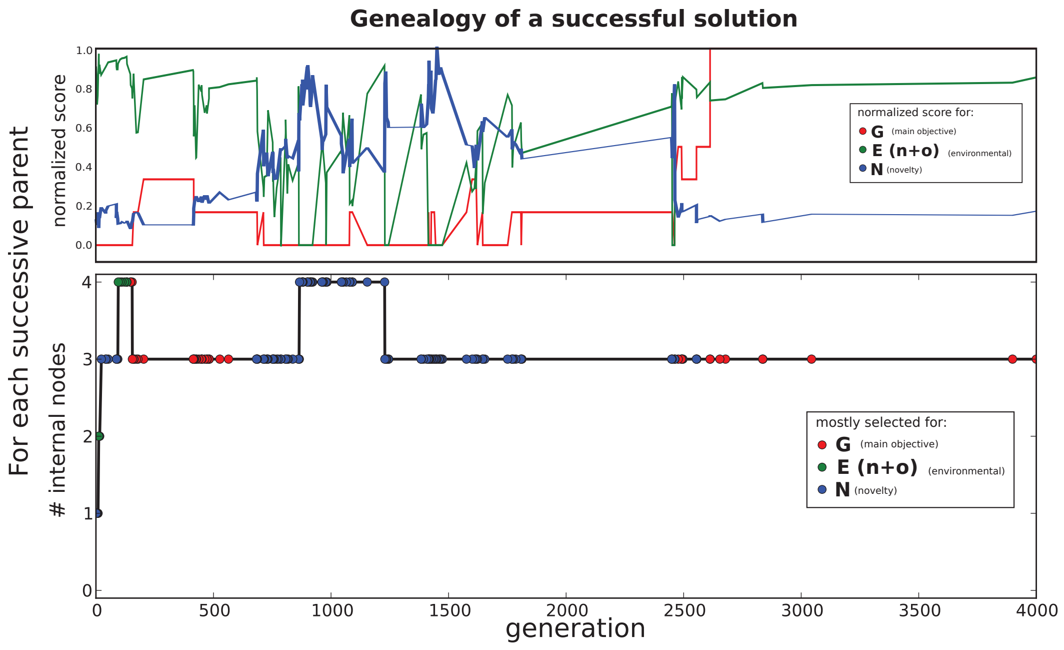


Figure A.1: Genealogy of a solution obtained at generation 4000 with **G+N+E (n+o)**. Each of the parents of the successful solution is represented as a dot in the middle graph. (Top graph) Objective scores for the parents. The black dotted line represent the best main objective of the current generation. (Bottom graph) Number of internal nodes for each of the successive parents. Each dot is colored in function of which objective it was most selected for.

A.3 Neural Networks

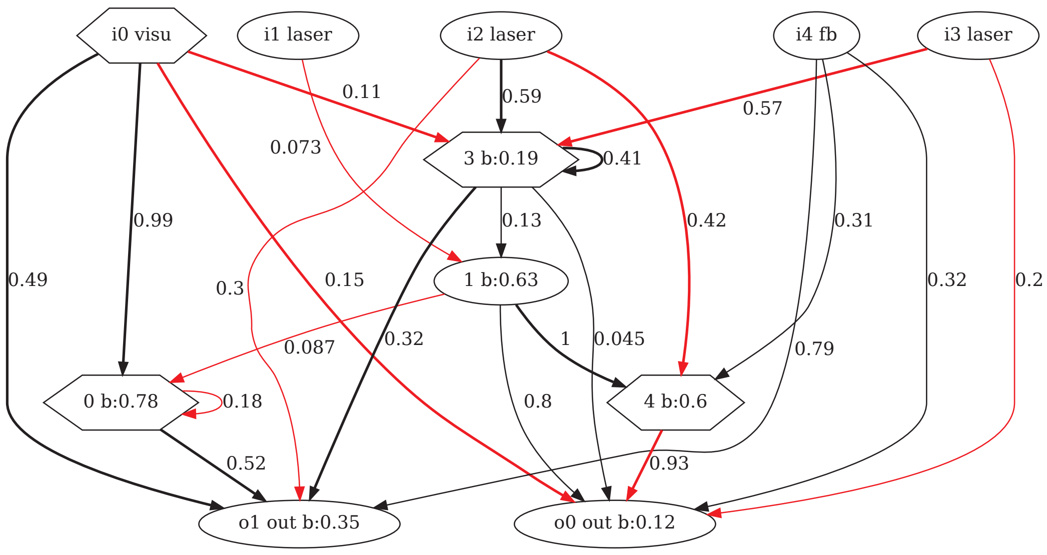


Figure A.2: Neural network obtained with fitness **G+N+E (n+o)**. Red arrows represent inhibitory connections (i.e. with negative weights); bold arrows represent Gaussian weights connections. Biases and offsets are not represented. The trajectory resulting from this controller is displayed on Figure 5.19.

A.4 Chronograms

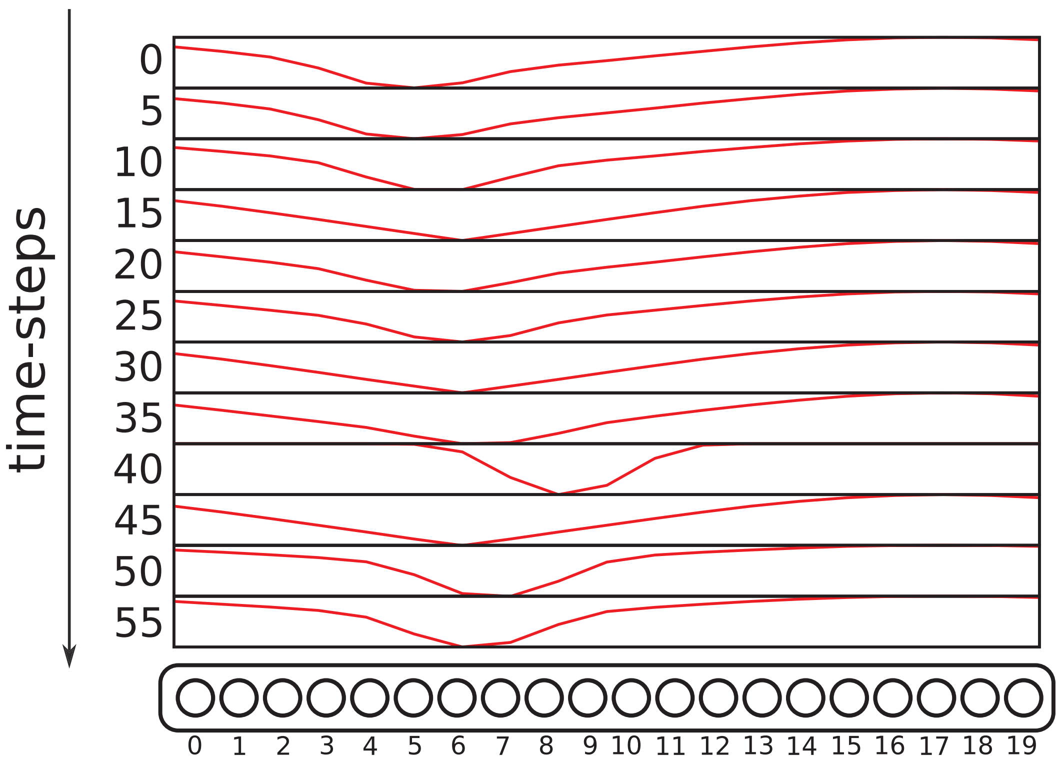


Figure A.3: Evolution of an internal map activity of controller displayed in A.2. The activity of the map is taken right after an occlusion (which means the visual input is set to 0.

A.5 Comparison between BCO and Standard Evaluation

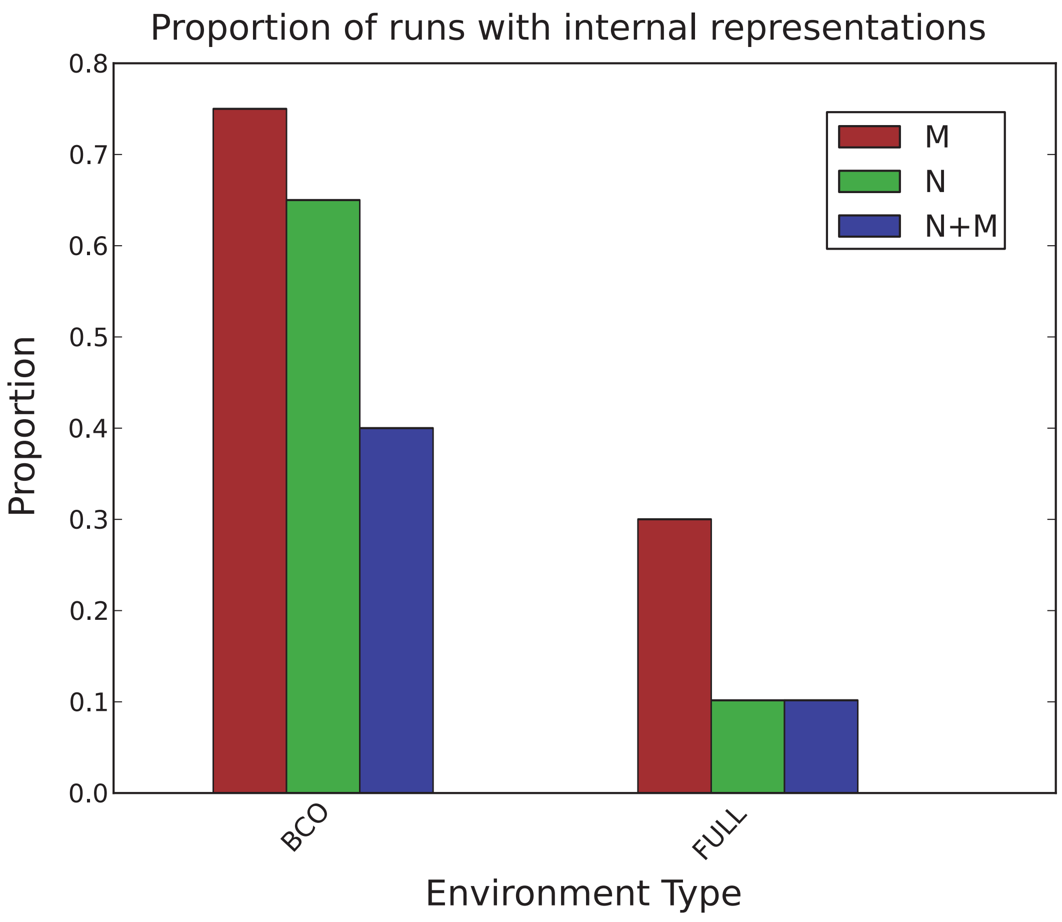


Figure A.4: Comparison between the use of BCO (evaluation over 15 contexts) and Full Evaluation (standard evaluation over 264 contexts: the contexts are the same as the ones used for the generalization test, see 5.2.3). . In all runs —standard and BCO—, the number of generation is 4000, the population size is 200. Therefore, the total number of context evaluations for a run is 1.2×10^7 for the BCO, 2.1×10^8 for Full evalutation. Only 10 runs with full evaluation were performed, because the runs were very time-consuming.

Bibliography

Alexander, G. E., Crutcher, M. D., et al. (1990). Functional architecture of basal ganglia circuits: neural substrates of parallel processing. *Trends in Neuroscience*, 13(7):266–271. (Cited page 39.)

Altenberg, L. (2004). Modularity in Evolution : Some Low-Level Questions. In *Modularity, Understanding the Development and Evolution of Natural Complex Systems*, pages 1–32. (Cited page 37.)

Amari, S.-i. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 87:77–87. (Cited pages 30, 49, 65 and 90.)

Antonelo, E., Schrauwen, B., Dutoit, X., Stroobandt, D., and Nuttin, M. (2007). Event detection and localization in mobile robot navigation using reservoir computing. *Artificial Neural Networks–ICANN 2007*, pages 660–669. (Cited page 33.)

Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. Oxford University Press New York. (Cited page 8.)

Bäck, T. and Hoffmeister, F. (1991). Extended selection mechanisms in genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 92–99. Morgan Kaufmann. (Cited page 26.)

Bäck, T., Hoffmeister, F., and Schwefel, H. P. (1991). A survey of evolution strategies. *Proceedings of the 4th International Conference on Genetic ALgorithms and their Applications*. (Cited page 10.)

Bäck, T. and Schwefel, H. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23. (Cited page 9.)

Baddeley, A. (1992). Working memory. *Science*, 255(5044):556–559. (Cited pages 49 and 59.)

Bader, J. and Zitzler, E. (2011). HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*. (Cited page 19.)

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21. (Cited page 11.)

Banzhaf, W., Koza, J., Ryan, C., Spector, L., and Jacob, C. (2000). Genetic programming. *IEEE Intelligent Systems*, 15(3):74–84. (Cited page 10.)

Beer, R. D. (1995). On the Dynamics of Small Continuous-Time Recurrent Neural Networks. *Adaptive Behavior*, 3(4):469–509. (Cited pages 32 and 33.)

Beer, R. D. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. *From animals to animats*. (Cited pages 48, 52, 53, 102 and 134.)

Beer, R. D. (2003). The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior*, 11(4):209–243. (Cited pages 53, 59 and 102.)

Beer, R. D. (2006). Parameter space structure of continuous-time recurrent neural networks. *Neural Computation*, 18(12):3009–3051. (Cited page 32.)

Berg, H. C., Brown, D. A., et al. (1972). Chemotaxis in escherichia coli analysed by three-dimensional tracking. *Nature*, 239(5374):500–504. (Cited page 47.)

Bergfeldt, N. and Linaker, F. (2002). Self-organized modulation of a neural robot controller. In *Proceedings of the International Joint Conference on Neural Networks*, pages 495–500. Citeseer. (Cited page 54.)

Beume, N., Naujoks, B., and Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669. (Cited page 19.)

Beume, N. and Rudolph, G. (2006). Faster S-Metric Calculation by Considering Dominated Hypervolume as Klee's Measure Problem. In *Proceedings of the Second IASTED Conference on Computational Intelligence (CI 2006)*, pages 231—-236. (Cited page 19.)

Billard, A. and Ijspeert, A. J. (2000). Biologically inspired neural controllers for motor control in a quadruped robot. In *Proceedings of the International Joint Conference on Neural Networks IJCNN 2000*, volume 6, pages 637–641. IEEE. (Cited page 55.)

Blynel, J. and Floreano, D. (2003). Exploring the T-maze: Evolving learning-like robot behaviors using CTRNNs. *Lecture notes in computer science*, pages 593–604. (Cited page 54.)

Bongard, J. (2011a). Innocent until proven guilty: Reducing robot shaping from polynomial to linear time. *Evolutionary Computation, IEEE Transactions on*, pages 1–15. (Cited page 24.)

Bongard, J. (2011b). Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences*, pages 1234–1239. (Cited page 32.)

Bongard, J. and Pfeifer, R. (2002). Relating Neural Network Performance to Morphological Differences in Embodied Agents. In *Proceedings of the Sixth International Conference on Cognitive and Neural Systems*, Boston. (Cited page 33.)

Braitenberg, V. (1986). *Vehicles: Experiments in synthetic psychology.* MIT press. (Cited pages 2 and 21.)

Braver, T., Cohen, J., and Servan-Schreiber, D. (1995). A computational model of prefrontal cortex function. *Advances in neural information processing systems, Nips*, pages 141–148. (Cited page 75.)

Bredeche, N., Montanier, J.-M., Liu, W., and Winfield, A. F. (2012). Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):101–129. (Cited page 33.)

Brindle, A. (1981). *Genetic algorithms for function optimization*. PhD thesis, University of Alberta, Department of Computer Science. (Cited page 11.)

Brockhoff, D. and Zitzler, E. (2007). Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. *Evolutionary Computation, 2007. CEC*, pages 2086–2093. (Cited page 19.)

Brooks, R. (1991a). Intelligence without reason. *Artificial intelligence: critical concepts*, pages 569–595. (Cited page 2.)

Brooks, R. A. (1991b). Intelligence without representation. *Artificial intelligence*, 47(1):139–159. (Cited pages 2, 3 and 46.)

Buhmann, M. D. (2003). *Radial basis functions: theory and implementations*, volume 12. Cambridge university press. (Cited page 30.)

Capi, G. and Doya, K. (2005). Evolution of recurrent neural controllers using an extended parallel genetic algorithm. *Robotics and Autonomous Systems*, 52:148–159. (Cited pages 4, 55 and 59.)

Charnes, A. and Cooper, W. W. (1961). *Management Models and Industrial Applications of Linear Programming, vol. 1*. John Wiley. (Cited page 13.)

Cheng, K., Collett, T., Pickhard, A., and Wehner, R. (1987). The use of visual landmarks by honeybees: Bees weight landmarks according to their distance from the goal. *Journal of Comparative Physiology A*, 161(3):469–475. (Cited page 47.)

Chomsky, N. (1965). *Aspects of the Theory of Syntax*, volume 11. The MIT press. (Cited page 2.)

Clark, A. and Toribio, J. (1994). Doing without representing? *Synthese*, 101(3):401–431. (Cited pages 2, 4, 46 and 52.)

Cliff, D., Husbands, P., Harvey, I., and Others (1993). Evolving visually guided robots. *From animals to animats*, 2:374–383. (Cited page 21.)

Clune, J., Beckmann, B. E., Ofria, C., and Pennock, R. T. (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 2764–2771. IEEE. (Cited page 37.)

Clune, J., Mouret, J.-B., and Lipson, H. (2012). The evolutionary origins of modularity. *arXiv preprint arXiv:1207.2743*, pages 1–17. (Cited page 37.)

Clune, J., Mouret, J.-B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological Sciences*, 280(1755). (Cited page 138.)

Cochocki, A. and Unbehauen, R. (1993). *Neural networks for optimization and signal processing.* John Wiley & Sons, Inc. (Cited page 32.)

Coello, C. A. (2006). Evolutionary multi-objective optimization: a historical view of the field. *Computational Intelligence Magazine, IEEE*, 1(1):28–36. (Cited page 20.)

Coello, C. A., Christiansen, A. D., and Aguirre, A. H. (1995). Multiobjective Design Optimization of Counterweight Balancing of a Robot Arm using Genetic Algorithms. In *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, TAI '95, pages 20—-23, Washington, DC, USA. IEEE Computer Society. (Cited page 13.)

Coello, C. A. and Lamont, G. B. (2004). *Applications of multi-objective evolutionary algorithms*, volume 1. World Scientific Publishing Company Incorporated. (Cited page 20.)

Corne, D., Jerram, N. R., Knowles, J. D., Oates, M. J., and J, M. (2001). PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001*, pages 283–290. Morgan Kaufmann Publishers. (Cited page 16.)

Corne, D. W., Knowles, J. D., and Oates, M. J. (2000). The Pareto Envelope-Based Selection Algorithm for Multi-objective Optimisation. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 839–848, London, UK, UK. Springer-Verlag. (Cited pages 15 and 16.)

Craik, K. J. W. (1943). *The nature of explanation.* Cambridge University Press. (Cited pages 2, 46 and 47.)

Cramer, N. L. (1985). A Representation for the Adaptive Generation of Simple Sequential Programs. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 183–187, Hillsdale, NJ, USA. L. Erlbaum Associates Inc. (Cited page 8.)

Dacke, M., Nilsson, D.-E., Scholtz, C. H., Byrne, M., and Warrant, E. J. (2003). Animal behaviour: insect orientation to polarized moonlight. *Nature*, 424(6944):33–33. (Cited page 47.)

Darwin, C. (1859). *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life.* (Cited page 7.)

Davidor, Y. (1991). *Genetic Algorithms and Robotics: A heuristic strategy for optimization*, volume 1. World Scientific Pub Co Inc. (Cited page 22.)

Dayan, P. and Abbott, L. F. (2005). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press. (Cited page 30.)

De Jong, K. (2006). *Evolutionary computation: a unified approach*. MIT Press. (Cited page 8.)

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, Ann Arbor, MI, USA. (Cited page 8.)

Deb, K. (2001). *Multi-objectives optimization using evolutionnary algorithms*. Wiley. (Cited pages 76, 78, 93 and 95.)

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *Parallel Problem Solving from Nature PPSN VI*, pages 849–858. Springer. (Cited pages 3, 15 and 17.)

Deb, K., Mohan, M., and Mishra, S. (2005). Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary Computation*, 13(4):501–525. (Cited pages 15, 16 and 17.)

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197. (Cited pages 15 and 17.)

Deco, G., Rolls, E. T., et al. (2005). Attention, short-term memory, and action selection: a unifying theory. *Progress in neurobiology*, 76(4):236–256. (Cited page 49.)

Deneve, S., Latham, P., and Pouget, A. (2001). Efficient computation and cue integration with noisy population codes. *Nature neuroscience*, 4(8):826–831. (Cited page 51.)

Deneve, S., Latham, P. E., and Pouget, A. (1999). Reading population codes: a neural implementation of ideal observers. *Nature neuroscience*, 2(8):740–745. (Cited pages 51, 61 and 112.)

Doncieux, S. and Mouret, J.-B. (2010a). Behavioral diversity measures for Evolutionary Robotics. In *Proceedings of IEEE-CEC'10*, pages 1303–1310. (Cited pages 22, 26 and 107.)

Doncieux, S. and Mouret, J.-B. (2010b). Behavioral diversity measures for evolutionary robotics. In *Proc. of IEEE CEC*, volume 2010, pages 1–8. (Cited pages 26 and 27.)

Doncieux, S. and Mouret, J.-B. (2013). Behavioral diversity with multiple behavioral distances. *IEEE Congress on Evolutionary Computation*, pages 1–8. (Cited page 26.)

Doncieux, S., Mouret, J.-B., Bredeche, N., and Padois, V. (2011). *Evolutionary Robotics: Exploring New Horizons*, pages 3–25. Springer. (Cited page 24.)

Drchal, J., Kapral, O., Koutník, J., and Šnorek, M. (2009). Combining Multiple Inputs in HyperNEAT Mobile Agent Controller. In *Proceedings of the 19th International Conference on Artificial Neural Networks*, pages 775–783, Berlin. Springer. (Cited page 37.)

Edlund, J. A., Chaumont, N., Hintze, A., Koch, C., Tononi, G., and Adami, C. (2011). Integrated information increases with fitness in the evolution of animats. *PLoS computational biology*, 7(10):e1002236. (Cited pages 53, 57 and 103.)

Eiben, A., Sprinkhuizen-Kuyper, I., and Thijssen, B. (1998). Competing crossovers in an adaptive ga framework. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 787–792. IEEE. (Cited page 9.)

Eiben, A. E. and Schippers, C. A. (1998). On evolutionary exploration and exploitation. *Fundamenta Informaticae*, 35(1):35–50. (Cited pages 9 and 10.)

Eiben, A. E. and Smith, J. E. (2003). *Introduction to evolutionary computing*. Springer. (Cited page 3.)

Elman, J. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211. (Cited pages 5, 32 and 42.)

Everson, R. and Fieldsend, J. (2002). Full elite sets for multi-objective optimisation. *Adaptive Computing in*, (Acdm):1–12. (Cited page 19.)

Filliat, D., Kodjabachian, J., and Meyer, J.-A. (1999). Incremental evolution of neural controllers for navigation in a 6-legged robot. *on Artificial Life and Robots*. (Cited page 55.)

Fleischer, M. (2003). The measure of pareto optima applications to multi-objective metaheuristics. In *Evolutionary Multi-Criterion Optimization*, pages 74–74. Springer. (Cited page 15.)

Floreano, D. (1994). Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. *From animals to animats*. (Cited pages 3, 4 and 33.)

Floreano, D. and Mattiussi, C. (2008). *Bio-inspired artificial intelligence: Theories, methods, and technologies*. (Cited page 29.)

Floreano, D. and Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *Systems, Man, and Cybernetics, Part*, 26(3):396–407. (Cited page 4.)

Floreano, D. and Nolfi, S. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Bradford Book. (Cited pages 3, 21 and 24.)

Fogel, D. (2000). Introduction to evolutionary computation. *Evolutionary Computation 1: Basic algorithms and operators*, 1:1. (Cited page 8.)

Fogel, D. B. (1991). *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling*. Ginn Press. (Cited page 8.)

Fogel, D. B. (2006). Evolutionary computation: toward a new philosophy of machine intelligence. *Evolutionary Computation*. (Cited page 10.)

Fogel, L. J. (1962). Autonomous automata. *Industrial Research*, 4:14–19. (Cited page 8.)

Fonseca, C. M. and Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation : Discussion and Generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. (Cited pages 3 and 14.)

Fonseca, C. M., Paquete, L., and López-Ibáñez, M. (2006). An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator. In *Congress on Evolutionary Computation (CEC 2006)*, pages 1157 —-1163. (Cited page 19.)

Fraser, A. S. (1957). Simulation of Genetic Systems by Automatic Digital Computers. I. Introduction. *Australian Journal of Biological Sciences*, 10:484–491. (Cited page 7.)

Gallagher, J. (1999). Evolution and analysis of dynamical neural networks for agents integrating vision, locomotion, and short-term memory. In *Proceedings of the Genetic and Evolutionary Computation Conference*. (Cited pages 34 and 53.)

Gardner, M. and Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)–a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636. (Cited page 32.)

Gigliotta, O. and Nolfi, S. (2008). On the coupling between agent internal and agent/environmental dynamics: Development of spatial representations in evolving autonomous robots. *Adaptive Behavior*, 16(2-3):148–165. (Cited page 136.)

Gigliotta, O., Pezzulo, G., and Nolfi, S. (2010). Emergence of an internal model in evolving robots subjected to sensory deprivation. *From Animals to Animats 11*, pages 575–586. (Cited pages 102 and 136.)

Gigliotta, O., Pezzulo, G., and Nolfi, S. (2011). Evolution of a predictive internal model in an embodied and situated agent. *Theory in biosciences*, 130(4):259–276. (Cited pages 5, 53, 59, 102, 125, 130, 134, 136 and 137.)

Girard, B., Tabareau, N., Pham, Q. C., Berthoz, A., and Slotine, J. J. (2008). Where neuroscience and dynamic system theory meet autonomous robotics: a contracting basal ganglia model for action selection. *Neural Networks*, 21(4):628–641. (Cited pages 30, 31, 37, 42, 50 and 76.)

Goldberg, D. E. (1987). Simple genetic algorithms and the minimal, deceptive problem. *Genetic algorithms and simulated annealing*, 74:88. (Cited page 3.)

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley. (Cited page 8.)

Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multi-modal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154. (Cited page 26.)

Gomez, F. J. (2009). Sustaining diversity using behavioral information distance. In *Proceedings of GECCO'09*, pages 113–120. ACM. (Cited pages 22, 26 and 27.)

Gould, S. J. and Vrba, E. S. (1982). Exaptation-a missing term in the science of form. *Paleobiology*, pages 4–15. (Cited pages 26 and 136.)

Grosan, C. and Oltean, M. (2004). Improving the performance of evolutionary algorithms for the multiobjective 0/1 knapsack probl em using epsilon-dominance. *Computational Science-ICCS 2004*, pages 674–677. (Cited page 16.)

Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. *Robotics, IEEE Transactions on*, 22(6):1115–1130. (Cited page 4.)

Gruau, F. (1994). Automatic definition of modular neural networks. *Adaptive Behavior*, pages 1–44. (Cited page 36.)

Gruau, F. (1995). Automatic Definition of Modular Neural Networks. *Adaptive Behaviour*, 3(2):151–183. (Cited page 5.)

Gurney, K., Prescott, T. J., and Redgrave, P. (2001a). A computational model of action selection in the basal ganglia. I. A new functional anatomy. *Biological cybernetics*, 84(6):401–410. (Cited page 37.)

Gurney, K., Prescott, T. J., and Redgrave, P. (2001b). A computational model of action selection in the basal ganglia. i. a new functional anatomy. *Biological cybernetics*, 84(6):401–410. (Cited page 50.)

Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., and Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806. (Cited pages 50, 51 and 59.)

Haldane, J. B. S. (1932). The causes of evolution. (Cited page 7.)

Hansen, N. and Koumoutsakos, P. (2003). Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11:1–18. (Cited page 11.)

Hansen, N. and Ostermeier, a. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–95. (Cited page 11.)

Hansen, T. F. (2003). Is modularity necessary for evolvability? Remarks on the relationship between pleiotropy and evolvability. *Bio Systems*, 69(2-3):83–94. (Cited page 37.)

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346. (Cited pages 2 and 46.)

Hartland, C., Bredeche, N., and Sebag, M. (2009). Memory-enhanced evolutionary robotics: the echo state network approach. In *In proc. of IEEE-CEC'09*, number section IV, pages 2788–2795. (Cited page 33.)

Hartwell, L. H., Hopfield, J. J., Leibler, S., and Murray, A. W. (1999). From molecular to modular cell biology. *Nature*, 402(6761 Suppl):C47—-52. (Cited page 36.)

Harvey, I. and Husbands, P. (1992). Evolutionary robotics. In *Proceedings of IEE Colloquium on Genetic Algorithms for Control Systems Engineering*, pages 1–4. (Cited pages 3, 4 and 21.)

Harvey, I., Husbands, P., Cliff, D., and Thompson, A. (1997). Evolutionary robotics: the Sussex approach. *Robotics and autonomous systems*, 20:205–224. (Cited page 24.)

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation, 2nd edition.* Prentice Hall PTR. (Cited page 29.)

Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544. (Cited page 31.)

Hoffmann, F. and Pfister, G. (1996). Evolutionary learning of a fuzzy control rule base for an autonomous vehicle. *Management of Uncertainty in Knowledge-Based.* (Cited page 4.)

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* Number 53. University of Michigan Press. (Cited page 8.)

Horn, J., Nafpliotis, N., and Goldberg, D. (1994). A niched Pareto genetic algorithm for multiobjective optmization. In *First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Volume 1.* (Cited page 14.)

Hornby, G. S. (2005). Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1729–1736. AcM. (Cited page 36.)

Hornby, G. S. and Lipson, H. (2001). Evolution of generative design systems for modular physical robots. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, pages 4146—-4151. (Cited page 36.)

Hornby, G. S. and Pollack, J. B. (2002). Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3):223–246. (Cited page 36.)

Hornby, G. S., Takamura, S., Yokono, J., Hanagata, O., Yamamoto, T., and Fujita, M. (2000). Evolving robust gaits with AIBO. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 3040–3045. IEEE. (Cited pages 4, 55, 56 and 59.)

Huband, S., Hingston, P., and While, L. (2003). An evolution strategy with probabilistic mutation for multi-objective optimisation. *Evolutionary*. (Cited page 19.)

Hutchins, E. (1995). Cognition in the Wild. *MIT press*. (Cited page 55.)

Igel, C., Hansen, N., and Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary computation*, 15(1):1–28. (Cited page 19.)

Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2008). Evolutionary many-objective optimization: A short review. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2419–2426. IEEE. (Cited pages 21 and 138.)

Ishiguro, A., Tokura, S., Kondo, T., Uchikawa, Y., and Eggenberger, P. (1999). Reduction of the gap between simulated and real environments in evolutionary robotics: a dynamically-rearranging neural network approach. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 3, pages 239–244. IEEE. (Cited page 55.)

Jaeger, H. (2002). *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach.* GMD-Forschungszentrum Informationstechnik. (Cited pages 32 and 33.)

Jaeger, H., Lukoševičius, M., Popovici, D., and Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352. (Cited page 30.)

Jakobi, N. (1997). Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive behavior*, 6(2):325–368. (Cited page 54.)

Jakobi, N. (1998). Running across the reality gap: Octopod locomotion evolved in a minimal simulation. *Evolutionary Robotics*. (Cited page 22.)

Jansen, T. and Wegener, I. (1999). On the analysis of evolutionary algorithms - a proof that crossover really can help. *Algorithms-ESA'99*, page 700. (Cited page 9.)

Jensen, M. (2003). Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. *Evolutionary Computation, IEEE Transactions on*, 7(5):503–515. (Cited page 19.)

Jensen, M. T. (2004). Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimisation. *Journal of Mathematical Modelling and Algorithms*, 3(4):323–347. (Cited page 25.)

Jin, Y. (2005). Evolutionary Optimization in Uncertain Environments. *IEEE Transactions on Evolutionary Computation*, 2610(1):264–317. (Cited page 24.)

Jin, Y., Olhofer, M., and Sendhoff, B. (2001). Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 1042–1049. (Cited page 13.)

Johnson-Laird, P. N. (1980). Mental models in cognitive science. *Cognitive science*, 4(1):71–115. (Cited pages 2, 46, 47 and 136.)

Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness.* Number 6. Harvard University Press. (Cited pages 2 and 46.)

Jordan, M. I. (1986). Serial order: A parallel distributed processing approach. *Advances in Connectionist Theory Speech*, 121(ICS-8604):471–495. (Cited page 32.)

Kashtan, N. and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13773–8. (Cited page 37.)

Keijzer, F. (2002). Representation in dynamical and embodied cognition. *Cognitive Systems Research*, 3(3):275–288. (Cited pages 2 and 46.)

Khare, V., Yao, X., and Deb, K. (2003). Performance scaling of multi-objective evolutionary algorithms. In *Evolutionary Multi-Criterion Optimization*, pages 72–72. Springer. (Cited page 21.)

Kim, D. (2004). Evolving internal memory for T-maze tasks in noisy environments. *Connection Science*, 16(3):183–210. (Cited page 54.)

Knowles, J. and Corne, D. (2003). Properties of an adaptive archiving algorithm for storing nondominated vectors. *Evolutionary Computation, IEEE Transactions on*, 7(2):100–116. (Cited page 19.)

Knowles, J. D. (2002). *Local-search and hybrid evolutionary algorithms for Pareto optimization*. PhD thesis. (Cited pages 15 and 19.)

Knowles, J. D. and Corne, D. (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary computation*, 8:149—-172. (Cited pages 15 and 16.)

Knowles, J. D., Watson, R., and Corne, D. (2001). Reducing local optima in single-objective problems by multi-objectivization. In *EMO'01: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, volume 1993 of *LNCS*, pages 269–283. Springer. (Cited pages 25 and 76.)

Koos, S., Mouret, J.-B., and Doncieux, S. (2012). The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics. *IEEE Transactions on Evolutionary Computation*, 1:1. (Cited page 54.)

Körding, K. P., Wolpert, D. M., et al. (2006). Bayesian decision theory in sensorimotor control. *Trends in cognitive sciences*, 10(7):319. (Cited pages 102 and 137.)

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, volume 229. MIT Press. (Cited pages 4 and 8.)

Kyriacou, T. (2011). Using an evolutionary algorithm to determine the parameters of a biologically inspired model of head direction cells. *Journal of computational neuroscience*. (Cited page 104.)

LaDage, L., Roth, T., Cerjanic, A., Sinervo, B., and Pravosudov, V. (2012). Spatial memory: are lizards really deficient? *Biology Letters*, 8(6):939–941. (Cited page 104.)

Larranaga, P. and Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer. (Cited page 8.)

Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–82. (Cited pages 15, 16 and 17.)

Laumanns, M., Thiele, L., and Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942. (Cited page 13.)

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551. (Cited pages 32 and 33.)

Lee, Y. (1991). Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. *Neural computation*, 3(3):440–449. (Cited page 30.)

Lehman, J. (2008). Exploiting open-endedness to solve problems through the search for novelty. *Proceedings of Alife XI*, pages 71–80. (Cited pages 5, 22, 24, 26, 27 and 101.)

Lehman, J., Risi, S., D'Ambrosio, D. B., and Stanley, K. O. (2012). Rewarding Reactivity to Evolve Robust Controllers without Multiple Trials or Noise. In *Artificial Life*, volume 13, pages 379–386. (Cited page 52.)

Lehman, J. and Stanley, K. O. (2010). Abandoning Objectives: Evolution through the Search for Novelty Alone. *Evolutionary computation*, pages 189–223. (Cited pages 24 and 27.)

Liénard, J., Guillot, A., and Girard, B. (2010). Multi-Objective Evolutionary Algorithms to Investigate Neurocomputational Issues : The Case Study of Basal Ganglia Models. In Meyer, J.-A., Guillot, A., and Hallam, J., editors, *From animals to animats 11*, LNAI. (Cited page 94.)

Linaker, F. and Jacobsson, H. (2001). Mobile robot learning of delayed response tasks through event extraction: A solution to the road sign problem and beyond. In *International Joint Conference On Artificial Intelligence*, volume 17, pages 777–782. (Cited page 54.)

Lippmann, R. P. (1989). Pattern classification using neural networks. *Communications Magazine, IEEE*, 27(11):47–50. (Cited page 32.)

Lund, H. H. and Miglino, O. (1996). From simulated to real robots. In *Proceedings of IEEE International Conference on Evolutionary Computation*. (Cited page 52.)

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560. (Cited page 33.)

Maniadakis, M. and Tani, J. (2009). Acquiring Rules for Rules: Neuro-Dynamical Systems Account for Meta-Cognition. *Adaptive Behavior*, 17(1):58–80. (Cited page 54.)

Maniadakis, M., Trahanias, P., and Tani, J. (2010). Self-organized executive control functions. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. (Cited page 34.)

Marr, D. (1982). Vision: A computational investigation into the human representation and processing of visual information, henry holt and co. *Inc., New York, NY*. (Cited pages 2, 46 and 47.)

Marstaller, L., Hintze, A., and Adami, C. (2012). Cognitive systems evolve complex representations for adaptive behavior. *arXiv preprint arXiv:1206.5771*. (Cited pages 53, 57, 100, 103 and 134.)

Marstaller, L., Hintze, A., and Adami, C. (2013). The evolution of representation in simple cognitive networks. (Cited page 100.)

Mataric, M. and Cliff, D. (1996). Challenges in evolving controllers for physical robots. *Robotics and autonomous systems*, 19:67–83. (Cited page 24.)

Mattiussi, C. (2005). *Evolutionary Synthesis of Analog Networks*. PhD thesis, École Polytechnique Fédérale de Lausanne. (Cited page 36.)

Mattiussi, C. and Floreano, D. (2007). Analog Genetic Encoding for the Evolution of Circuits and Networks. *IEEE Transactions on Evolutionary Computation*, 11(5):596–607. (Cited page 36.)

Mayer, H., Gomez, F., Wierstra, D., Nagy, I., Knoll, A., and Schmidhuber, J. (2008). A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *Advanced Robotics*, 22(13-14):1521–1537. (Cited page 34.)

Mayr, E. (1942). *Systematics and the origin of species, from the viewpoint of a zoologist*. Harvard Univ Pr. (Cited page 7.)

McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. (1955). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. *AI Magazine*, 27(4):12. (Cited page 1.)

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133. (Cited page 29.)

McNally, L., Brown, S. P., and Jackson, A. L. (2012). Cooperation and the evolution of intelligence. *Proceedings. Biological sciences / The Royal Society*, 279(1740):3027–34. (Cited pages 56 and 100.)

McNaughton, B. L., Barnes, C. a., Gerrard, J. L., Gothard, K., Jung, M. W., Knierim, J. J., Kudrimoti, H., Qin, Y., Skaggs, W. E., Suster, M., and Weaver, K. L. (1996). Deciphering the hippocampal polyglot: the hippocampus as a path integration system. *The Journal of experimental biology*, 199(Pt 1):173–85. (Cited page 50.)

McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., and Moser, M.-B. (2006). Path integration and the neural basis of the'cognitive map'. *Nature Reviews Neuroscience*, 7(8):663–678. (Cited pages 41, 50 and 126.)

Meeden, L. and Kumar, D. (1998). Trends in evolutionary robotics. *Soft computing for intelligent robotic*. (Cited page 24.)

Mendel, G. (1865). Experiments in plant hybridization (1865). In *Read at the meetings of February 8th, and March 8th*. (Cited page 7.)

Meyer, J.-A., Husbands, P., and Harvey, I. (1998). Evolutionary robotics: A survey of applications and problems. *Evolutionary Robotics*, pages 1–22. (Cited pages 3, 21 and 24.)

Miglino, O., Denaro, D., Tascini, G., and Parisi, D. (1998). Detour Behavior in Evolving Robots: Are Internal Representations Necessary? In *Evolutionary Robotics*, pages 59—-70. (Cited page 52.)

Miglino, O., Nafasi, K., and Taylor, C. E. (1994). Selection for wandering behavior in a small robot. *Artif. Life*, 2(1):101–116. (Cited page 34.)

Miller, B. L. and Goldberg, D. E. (1995). Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Evolutionary Computation*, 4:113—-131. (Cited page 11.)

Mitchell, M. (1998). *An introduction to genetic algorithms.* The MIT press. (Cited page 8.)

Moriguchi, H. and Honiden, S. (2011). Sustaining Behavioral Diversity in NEAT. In *Proceedings of the 12th Annual conference on Genetic and evolutionary computation - GECCO '11.* (Cited page 35.)

Mouret, J.-B. (2008). *Pressions sélectives multiples pour l'évolution de réseaux de neurones destinés à la robotique.* Thèse de doctorat, Université Pierre et Marie Curie (UPMC), 4 place jussieu, 75005 PARIS. (Cited pages 34 and 37.)

Mouret, J.-B. (2011). Novelty-based Multiobjectivization. In *New Horizons in Evolutionary Robotics: Extended Contributions from the 2009 EvoDeRob Workshop*, volume 341 of *Studies in Computational Intelligence*, pages 139–154. Springer. (Cited pages 25, 67, 76 and 107.)

Mouret, J.-B. and Doncieux, S. (2006). Incremental evolution of target-following neuro-controllers for flapping-wing animats. In *From Animals to Animats 9*, pages 606–618. (Cited page 34.)

Mouret, J.-B. and Doncieux, S. (2008a). Incremental evolution of animats' behaviors as a multi-objective optimization. *From Animals to Animats 10*, pages 210–219. (Cited pages 25, 34 and 37.)

Mouret, J.-B. and Doncieux, S. (2008b). {MENNAG}: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars. *Evolutionary Intelligence*, 1:187–207. (Cited page 37.)

Mouret, J.-B. and Doncieux, S. (2009a). Evolving modular neural-networks through exaptation. In *Proc. of CEC 2009.* (Cited pages 26, 136 and 139.)

Mouret, J.-B. and Doncieux, S. (2009b). Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *Proceedings of the Congress on Evolutionary Computation 2009 (to appear).* (Cited page 24.)

Mouret, J.-B. and Doncieux, S. (2010). Sferesv2: Evolvin' in the Multi-Core World. In *IEEE Congress on Evolutionary Computation 2010 CEC 2010*. (Cited page 113.)

Mouret, J.-B. and Doncieux, S. (2012a). Encouraging Behavioral Diversity in Evolutionary Robotics: An Empirical Study. *Evolutionary computation*, 20(1):91–133. (Cited pages 4, 5, 26, 43, 56, 59, 67 and 101.)

Mouret, J.-B. and Doncieux, S. (2012b). Encouraging Behavioral Diversity in Evolutionary Robotics: an Empirical Study. *Evolutionary Computation*. (Cited pages 10, 22, 24 and 76.)

Mouret, J.-B., Doncieux, S., and Girard, B. (2010). Importing the Computational Neuroscience Toolbox into Neuro-Evolution—Application to Basal Ganglia. In *Proceedings of GECCO'10*. (Cited pages 37, 38, 39, 40, 41, 42, 94 and 96.)

Müller, N. G., Mollenhauer, M., Rösler, A., and Kleinschmidt, A. (2005). The attentional field has a mexican hat distribution. *Vision Research*, 45(9):1129–1137. (Cited page 50.)

Muller, R. U., Stead, M., and Pach, J. (1996). The hippocampus as a cognitive graph. *The Journal of general physiology*, 107(6):663–694. (Cited page 50.)

Nakamura, H., Ishiguro, A., and Uchikawa, Y. (2000). Evolutionary construction of behavior arbitration mechanisms based on dynamically-rearranging neural networks. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00*, pages 158–165. (Cited page 55.)

Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370. (Cited pages 4, 22, 24, 56, 87, 106 and 133.)

Nelson, A. L. and Grant, E. (2007). Aggregate selection in evolutionary robotics. In *Mobile Robots: The Evolutionary Approach*, volume 50, pages 63–87. (Cited page 24.)

Nolfi, S. (2002). Power and the limits of reactive agents. *Neurocomputing*, 42(1):119–145. (Cited pages 4, 55 and 100.)

Nolfi, S. (2005). Categories formation in self-organizing embodied agents. *Handbook of categorization in cognitive science*, pages 869—-889. (Cited page 136.)

Nolfi, S., Floreano, D., Miglino, O., and Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics. *Artificial Life IV*. (Cited page 22.)

Nolfi, S. and Tani, J. (1999). Extracting regularities in space and time through a cascade of prediction networks: The case of a mobile robot navigating in a structured environment. *Connection Science*, 11(2):125–148. (Cited pages 53 and 102.)

O'Keefe, J. and Nadel, L. (1978). *The hippocampus as a cognitive map*, volume 3. Clarendon Press Oxford. (Cited pages 48, 50 and 59.)

Ollion, C. and Doncieux, S. (2011). Why and how to measure exploration in behavioral space. In *Proceedings of GECCO'11*. ACM. (Cited pages 22, 26, 27, 56 and 67.)

Ollion, C. and Doncieux, S. (2012). Towards Behavioral Consistency in Neuroevolution. In *From Animals to Animats: Proceedings of the 12th International Conference on Adaptive Behaviour (SAB 2012)*, pages 1–10. (Cited pages 73 and 74.)

Ollion, C., Pinville, T., and Doncieux, S. (2012). With a little help from selection pressures: evolution of memory in robot controllers. In *Proc. Alife XIII*, pages 1–8. (Cited pages 42, 73 and 74.)

O'Reilly, R. C. and Frank, M. J. (2006). Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*, 18(2):283–328. (Cited page 37.)

Pareto, V. (1897). *Cours d'Economie Politique*, volume 2. F. Rouge & Cie., Lausanne, Switzerland. (Cited page 11.)

Parker, G. and Georgescu, R. (2006). Using cyclic genetic algorithms to evolve multi-loop control programs. *Mechatronics and Automation,*. (Cited page 52.)

Pfeifer, R. and Bongard, J. C. (2006). *How the body shapes the way we think: a new view of intelligence*. MIT press. (Cited pages 2, 3 and 46.)

Pfeifer, R., Iida, F., and Bongard, J. (2003). New robotics: design principles for intelligent systems. *Artificial life*, 11(1-2):99–120. (Cited page 24.)

Pinville, T. (2013). *Robotique évolutionniste : influence des pressions de sélection sur l'émergence d'une forme de mémoire interne*. These de doctorat, ISIR / UPMC, 4 Place JUSSIEU 75005 Paris - France. (Cited pages 54, 55, 77, 100, 101, 102, 106 and 138.)

Pinville, T. and Doncieux, S. (2010). Automatic synthesis of working memory neural networks with neuroevolution methods. *Neurocomp 2010*. (Cited pages 41 and 75.)

Pinville, T., Koos, S., Mouret, J.-B., and Doncieux, S. (2011). How to Promote Generalisation in Evolutionary Robotics: the ProGAb Approach. *Proc. of GECCO'11*. (Cited pages 25, 34, 56, 74, 76, 102, 112 and 137.)

Platek, S. (2009). *Foundations in evolutionary cognitive neuroscience*. Cambridge University Press. (Cited page 137.)

Pratihar, D. K. (2003). Evolutionary robotics — A review. *Sadhana*, 28(6):999–1009. (Cited page 24.)

Prusinkiewicz, P. and Lindenmayer, A. (1990). *The algorithmic beauty of plants*. Springer-Verlag. (Cited page 36.)

Quagliarella, D. and Vicini, A. (1997). Coupling genetic algorithms and gradient based optimization techniques. *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science - Recent advances and industrial applications*, pages 289–309. (Cited page 13.)

Quinton, J. (2010a). Exploring and optimizing dynamic neural fields parameters using genetic algorithms. *IEEE World Congress on Computational Intelligence ICJNN*, pages 0–6. (Cited pages 90 and 91.)

Quinton, J.-C. (2010b). Exploring and optimizing dynamic neural fields parameters using genetic algorithms. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–7. IEEE. (Cited pages 50, 65 and 66.)

Ranjithan, S., Eheart, J., and Liebman, J. (1992). Incorporating fixed-cost component of pumping into stochastic groundwater management: A genetic algorithm-based optimization approach. *Eos Transactions AGU*, 74:125. (Cited page 13.)

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. In *Royal Aircraft Establishment Translation No. 1122, B. F. Toms, Trans.* Ministry of Aviation, Royal Aircraft Establishment, Farnborough Hants. (Cited page 8.)

Rechenberg, I. (1973). *Evolutionsstrategie – Optimierung technisher Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, Stuttgart, GER. (Cited pages 8 and 16.)

Redgrave, P., Prescott, T. J., Gurney, K., et al. (1999). The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience*, 89(4):1009–1023. (Cited pages 50 and 59.)

Redish, A. (1994). *Contributions to a Computational Neuroscience Theory of Rodent Navigation.* PhD thesis. (Cited pages 48 and 50.)

Reil, T. and Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *Evolutionary Computation, IEEE Transactions on*, 6(2):159–168. (Cited pages 55 and 59.)

Risi, S. (2011). Enhancing ES-HyperNEAT to Evolve More Complex Regular Neural Networks. In *Proceedings of the 2011 anual conference on Genetic and evolutionary computation: GECCO'11.* (Cited page 37.)

Risi, S. and Stanley, K. O. (2010). Indirectly Encoding Neural Plasticity as a Pattern of Local Rules. *SAB*, pages 533–543. (Cited page 37.)

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(1958). (Cited page 32.)

Rougier, N. P. and Vitay, J. (2006). Emergence of attention within a neural population. *the official journal of the International Neural Network Society*, 19(5). (Cited pages 49, 50, 59, 65, 90 and 92.)

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536. (Cited page 32.)

Rylatt, R. M. and Czarnecki, C. A. (2000). Embedding Connectionist Autonomous Agents in Time: The 'Road Sign Problem'. *Neural Processing Letters*, 12(2):145–158. (Cited pages 54 and 75.)

Salinas, E. and Abbott, L. (1994). Vector reconstruction from firing rates. *Journal of computational neuroscience*, 1(1-2):89–107. (Cited pages 61 and 112.)

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA. (Cited page 8.)

Seok, H., Lee, K., and Zhang, B. (2000). An on-line learning method for object-locating robots using genetic programming on evolvable hardware. *Symposium on Artificial Life and Robotics*. (Cited page 52.)

Shettleworth, S. J. (2009). *Cognition, evolution, and behavior*. Oxford University Press, USA. (Cited pages 1, 2, 4, 47, 48 and 137.)

Singer, W. and Gray, C. M. (1995). Visual feature integration and the temporal correlation hypothesis. *Annual review of neuroscience*, 18(1):555–586. (Cited page 31.)

Slocum, A. C., Downey, D. C., and Beer, R. D. (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *In*, pages 430–439. MIT Press. (Cited page 53.)

Smith, T. F. and Waterman, M. S. (1981). Comparison of biosequences. *Advances in Applied Mathematics*, 2(4):482–489. (Cited page 28.)

Srinivas, N. and Deb, K. (1994). Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248. (Cited pages 3 and 14.)

Stanley, K. O. (2006). Exploiting regularity without development. *Proceedings of the AAAI Fall Symposium on Developmental Systems*. (Cited page 37.)

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162. (Cited pages 37 and 138.)

Stanley, K. O., Ambrosio, D. D., and Gauci, J. (2009). A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks. *Artif. Life*, 15(2):1–39. (Cited pages 5, 37 and 138.)

Stanley, K. O. and Miikkulainen, R. (2002a). Evolving Neural Networks Through Augmenting Topologies. *Evolutionary Computation*, 10(99–127). (Cited page 5.)

Stanley, K. O. and Miikkulainen, R. (2002b). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127. (Cited pages 10, 26, 34 and 35.)

Stanley, K. O. and Miikkulainen, R. (2003). Evolving adaptive neural networks with and without adaptive synapses. *GECCO*. (Cited pages 34 and 35.)

Suchorzewski, M. (2011). A novel generative encoding for evolving modular, regular and scalable networks. *GECCO'11*, pages 1523–1530. (Cited pages 36 and 37.)

Tani, J. and Nolfi, S. (1999). Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Networks*, 12(7):1131–1141. (Cited pages 53 and 102.)

Taube, J. S., Muller, R. U., and Ranck, J. (1990). Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis. *The Journal of Neuroscience*, 10(2):420–435. (Cited pages 50, 51 and 59.)

Teo, J. and Abbass, H. (2005). Multiobjectivity and complexity in embodied cognition. *IEEE Transactions on Evolutionary Computation*, 9(4):337–360. (Cited page 24.)

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological review*, 55(4):189–208. (Cited pages 2, 4, 46, 47, 48 and 50.)

Tonelli, P. (2012). *Relations entre plasticité synaptique et régularité des codages en neuro-évolution.* Thèse de doctorat, Université Pierre et Marie Curie, 4 place Jussieu, 75005 Paris. (Cited pages 36 and 41.)

Touretzky, D. S., Redish, A. D., and Wan, H. S. (1993). Neural representation of space using sinusoidal arrays. *Neural Computation*, 5(6):869–884. (Cited pages 48 and 50.)

Trujillo, L., Olague, G., Lutton, E., and de Vega, F. (2008). Discovering several robot behaviors through speciation. *Applications of Evolutionary Computing*, pages 164–174. (Cited pages 22, 26 and 27.)

Trullier, O., Wiener, S. I., Berthoz, A., and Meyer, J. A. (1997). Biologically based artificial navigation systems: Review and prospects. *Progress in neurobiology*, 51(5):483–544. (Cited pages 4, 48, 50 and 136.)

Ulbricht, C. (1996). Handling time-warped sequences with neural networks. In *From animals to animats*, pages 180–192. The MIT Press. (Cited pages 33 and 54.)

Van Dartel, M., Sprinkhuizen-Kuyper, I., Postma, E., and Van Den Herik, J. (2005). Reactive agents and perceptual ambiguity. *Adaptive Behavior*, 13(3):227–242. (Cited page 100.)

Varela, F. J., Thompson, E., and Rosch, E. (1991). *The embodied mind: Cognitive science and human experience.* (Cited pages 2 and 46.)

Variano, E. A., McCoy, J. H., and Lipson, H. (2004). Networks, dynamics, and modularity. *Physical review letters*, 92(18):188701. (Cited page 36.)

Vickerstaff, R. and Di Paolo, E. (2005a). An evolved agent performing efficient path integration based homing and search. *Advances in Artificial Life*, pages 221–230. (Cited pages 55 and 59.)

Vickerstaff, R. J. and Di Paolo, E. A. (2005b). Evolving neural models of path integration. *Journal of Experimental Biology*, 208(17):3349–3366. (Cited page 55.)

Walker, J., Garrett, S., and Wilson, M. S. (2003). Evolving Controllers for Real Robots: A Survey of the Literature. *Adaptive Behavior*, 11(3):179–203. (Cited page 24.)

Wang, S. L., Shafi, K., Lokan, C., and Abbass, H. A. (2013). An agent-based model to simulate and analyse behaviour under noisy and deceptive information. *Adaptive Behavior*, 21(2):96–117. (Cited page 102.)

Watson, R. (2002). Embodied Evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18. (Cited page 52.)

Wehner, R. and Srinivasan, M. V. (2003). Path integration in insects. (Cited page 47.)

Whiteson, S., Stone, P., Stanley, K. O., Miikkulainen, R., and Kohl, N. (2005). Automatic feature selection in neuroevolution. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1225–1232, New York, NY, USA. ACM. (Cited page 34.)

Wiener, S. I. and Taube, J. S. (2005). *Head Direction Cells and the Neural Mechanisms of Spatial Orientation (Bradford Books).* The MIT press. (Cited page 51.)

Wilson, S. W. (1991). The animat path to AI. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 15–21, Cambridge, MA, USA. MIT Press. (Cited page 1.)

Wolpert, D., Ghahramani, Z., and Jordan, M. (1995). An internal model for sensorimotor integration. *Science*, 269(5232):1880. (Cited pages 2 and 46.)

Woolley, B. G. and Stanley, K. O. (2011). Evolving a single scalable controller for an octopus arm with a variable number of segments. *Parallel Problem Solving from Nature–PPSN XI*, pages 270–279. (Cited page 36.)

Yamauchi, B. and Beer, R. D. (1996). Spatial learning for navigation in dynamic environments. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 26(3):496–505. (Cited page 32.)

Yang, Q. and Ding, S. (2007). Novel algorithm to calculate hypervolume indicator of Pareto approximation set. *ICIC 2007*. (Cited page 19.)

Yujian, L. and Bo, L. (2007). A normalized levenshtein distance metric. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1091–1095. (Cited page 27.)

Ziemke, T. (1996). Towards adaptive behaviour system integration using connectionist infinite state automata. In *From animals to animats 4*. (Cited page 33.)

Ziemke, T. (1999). Remembering how to behave: Recurrent neural networks for adaptive robot behavior. *Recurrent neural networks: Design and applications*, pages 355–389. (Cited page 34.)

Ziemke, T., Bergfeldt, N., Buason, G., Susi, T., and Svensson, H. (2004). Evolving cognitive scaffolding and environment adaptation: a new research direction for evolutionary robotics. *Connection Science*, 16(4):339–350. (Cited pages 4, 54 and 55.)

Ziemke, T. and Thieme, M. (2002). Neuromodulation of Reactive Sensorimotor Mappings as a Short-Term Memory Mechanism in Delayed Response Tasks. *Adaptive Behavior*, 10(3-4):185–199. (Cited pages 4, 34, 54, 59 and 75.)

Zitzler, E. and Künzli, S. (2004). Indicator-based selection in multiobjective search. *Parallel Problem Solving from Nature-PPSN VIII*, (i):1–11. (Cited page 19.)

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *Computer Engineering*, TIK-Report(103):1–21. (Cited pages 15 and 16.)

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE*, 3(4):257–271. (Cited pages 15 and 19.)

Zufferey, J.-c., Floreano, D., Leeuwen, M. V., and Merenda, T. (2002). Evolving Vision-Based Flying Robots. *Evolutionary Computation*, pages 592–600. (Cited pages 52 and 59.)