DOCTOR OF SCIENCES

Challenging the Emergence of Modularity in Evolutionary Neural Networks

Nicolay, Delphine

Award date:
2018

Awarding institution:
University of Namur

UNIVERSITÉ DE NAMUR

FACULTÉ DES SCIENCES

DÉPARTEMENT DE MATHÉMATIQUE

Challenging the Emergence of Modularity in Evolutionary Neural Networks

Thèse présentée par
Delphine Nicolay
pour l'obtention du grade
de Docteur en Sciences

Composition du Jury :

Mauro BIRATTARI
Timoteo CARLETTI (Promoteur)
Renaud LAMBIOTTE (Promoteur)
Alexandre MAYER (Président du Jury)
Andrea ROLI

Août 2018

Graphisme de couverture : ©Presses universitaires de Namur

Rempart de la Vierge, 13
B-5000 Namur (Belgique)

Imprimé en Belgique

ISBN : 978-2-39029-012-4
Dépôt légal : D/2018/1881/18

Université de Namur
Faculté des Sciences
rue de Bruxelles, 61, B-5000 Namur (Belgique)

Discussion sur l'émergence de modularité dans les réseaux de neurones évolutionnaires

par Delphine Nicolay

Résumé : La modularité est une caractéristique très souvent observée dans les réseaux biologiques et artificiels. Elle rend les réseaux plus facilement modifiables, donc plus rapidement capables de s'adapter à de nouveaux environnements, et finalement plus résistants. Malgré ses avantages, cette propriété reste assez controversée et un débat sur la nature des modules et sur les raisons de leur émergence reste ouvert. L'objectif de ce travail est d'étudier la modularité dans les réseaux de neurones évolutionnaires, c'est-à-dire dans des réseaux de neurones artificiels entraînés à accomplir certaines tâches prédéfinies. Cet entraînement est réalisé au moyen de techniques d'optimisation évolutionnaire telles que les algorithmes génétiques qui imitent les principes d'évolution naturelle de Darwin. Pour cela, nous développons des applications assez simples dans les domaines de la robotique évolutionnaire et la reconnaissance d'images. Pour chacune d'elles, nous entamons notre analyse par la comparaison des performances atteintes par des réseaux modulaires et non-modulaires. Nous nous concentrons ensuite sur la recherche de modularité et nous essayons d'identifier les conditions nécessaires à son émergence en considérant différentes hypothèses souvent citées dans la littérature

Challenging the emergence of modularity in evolutionary neural networks

by Delphine Nicolay

Abstract: Modularity is a widespread feature of biological and artificial networks. It makes networks more easily evolvable, i.e. capable of rapidly adapting to new environments, and eventually more resilient. Despite its advantages, this characteristic is a controversial issue and the debate remains open on the nature of existing modules and the reasons for their emergence. In this work, we study modularity in evolutionary neural networks, i.e. artificial neural networks evolved to accomplish some predefined tasks. This evolution is performed by means of evolutionary optimization techniques such as genetic algorithms that mimic the Darwinian principles of natural evolution. With this aim, we develop simple applications in the frameworks of evolutionary robotics and pattern recognition. For each of them, we start our analysis by comparing the performance of modular and non-modular networks. We then focus on the search for modularity and we try to identify the conditions that cause its emergence by considering different hypotheses often met in the literature.

Thèse de doctorat en Sciences Mathématiques (Ph.D. thesis in Mathematics)
Date: 27/08/2018
Département de Mathématique
Promoteurs (Advisors): Timoteo CARLETTI, Renaud LAMBIOTTE

Remerciements

La réalisation d'une thèse de doctorat n'est pas un long fleuve tranquille. C'est un travail prenant et parfois difficile, ponctué de moments de joie mais aussi de moments de doute et de découragement. Arrivée au terme de celui-ci, j'aimerais remercier toutes les personnes qui, par leurs connaissances scientifiques, leurs conseils, leurs encouragements ou leur simple présence, ont contribué à la réussite de ce projet.

Je tiens avant tout à remercier mes promoteurs, Timoteo Carletti et Renaud Lambiotte, de m'avoir encadrée pendant ces six années de recherche. Teo, je te remercie de m'avoir soutenue dans mes choix, que ce soit dans la thèse ou dans d'autres projets comme la préparation de l'agrégation. Je te remercie également pour ton indulgence et ta confiance en mes capacités, qui a souvent (si pas toujours) été plus grande que la mienne... Merci aussi pour tes nombreuses relectures, ton aide et tes innombrables conseils. J'aimerais enfin te remercier pour ton incroyable disponibilité. Que tu sois à l'autre bout du globe (Australie, Japon, etc.), en vacances ou même à l'hôpital (pour ton genou ou suite aux acrobaties de ton fils), tu as toujours trouvé du temps pour répondre à mes questions, et je t'en suis infiniment reconnaissante! Renaud, merci pour tes commentaires toujours pertinents.

Je souhaite aussi remercier Andrea Roli, sans qui je n'aurais jamais commencé cette thèse. Notre rencontre lors de mon stage de master à Cesena a été déterminante dans mon choix. Je te remercie pour le temps que tu m'as consacré, pour nos discussions, toujours enrichissantes, et pour ton suivi régulier tout au long de ce travail.

Merci au dernier membre de mon comité d'accompagnement, André Füzfa, pour ses remarques lors de l'épreuve de confirmation et ses encouragements. Mes remerciements vont également à Mauro Birattari et Alexandre Mayer, qui ont accepté de faire partie de mon jury de thèse. Merci à vous pour vos remarques constructives qui ont grandement contribué à l'amélioration de ce manuscrit. Merci aussi à André Hardy pour la relecture attentive de mes interprétations statistiques.

J'aimerais à présent remercier tous les membres de ce département si chaleureux. En particulier, merci à nos formidables secrétaires, Pascale et Alice, sans qui le département ne tournerait pas rond. Nos discussions (enfants et cuisine pour l'une, péripéties en voiture et bêtises de Tom pour l'autre) ont toujours été un réel plaisir!

Merci à mes collègues de bureau, Céline, Pauline, et Ambi. Ambi, merci pour tes nombreux encouragements, parfois excessifs mais tellement bons pour le moral. Céline, merci pour ton écoute, ta bonne humeur constante et ton rire reconnaissable entre tous! Pauline, merci pour nos conversations et nos confidences. Merci aussi pour ton inimitable distraction qui fait ressortir mon côté "mère poule" (d'ailleurs, as-tu tes clés, ton gsm et tes lunettes?) et qui me fait si souvent rire! Merci à Anne-Sophie C. et Charlotte, mes "grandes sœurs" au département, de m'avoir prise sous leur aile. Le département n'a plus été le même après votre départ. Merci aux filles du bureau 135, Eve et Mara, pour l'animation (musiques, fous rires, chants de Noël, bulles de savon, décorations, etc.) apportée dans le couloir du premier étage. Merci aux frères Dehaye, Jérémy et Jon, pour leurs avis bien tranchés et leur compagnie si divertissante. Merci aux habitués de la salle café, Audrey, Emilie, le groupe cosmo (Sandrine, Sébastien et Marie L.), Carole, Anne-Sophie L. et Alexandre pour ces nombreux temps de midi passés ensemble. Merci à Martin et Nico pour leur humour et leurs jeux de mots. Merci aux amateurs de l'Arsenal, les inséparables Marie P. et Morgane, Manon, Joanna, Jojo, Julien B., Francois S., François L. et Arnaud pour ces débats passionants, amusants et parfois tellement surréalistes. Merci à Julien P. pour ses passages réguliers au bureau. Merci aussi à vous tous pour les nombreux moments partagés en dehors du département, que ce soient les soirées films, les matchs de foot, les concerts, les barbecues et repas en tout genre, ainsi que ces inoubliables sorties kayak et Pairi Daiza!

Merci à tous mes amis proches, que ce soient ceux du secondaire, de l'unif ou du tennis de table de m'avoir accompagnée jusqu'à ce jour. Merci à vous, Amandine, Aurélie D. (la Petite), Aurélie H. (la Grande), Cédric, Christophe, Elodie, Isabelle, Perrine et Rudi.

Merci à mes parents pour leur amour et leur soutien dans toutes mes décisions, même celle, si surprenante pour eux, de faire des études de mathématiques... Merci de m'avoir donné les moyens de réaliser ce travail et d'avoir toujours cru en moi. Merci également à mes frères et sœur, Sébastien, Céline et Quentin, ainsi qu'à leur conjoint, Coralie et Laurent, de me supporter (dans tous les sens du terme) au jour le jour. Merci à toi, Marraine, pour cette complicité qui nous lie depuis si longtemps.

Ces dernières années, j'ai peu à peu découvert le bonheur (et les responsabilités...) lié aux mots "tantine" et "marraine". Je voudrais donc remercier Louise, Anaïs, Mathys, Arthur et Achille. Mes loulous, vos sourires, vos câlins et les moments passés à vos côtés m'ont très souvent permis de décompresser et de me déconnecter du travail présenté dans ces pages.

A vous tous, merci du fond du coeur.

Delphine

Contents

Introduction

Context

In the last decades, networks theory has attracted considerable attention and has been applied with success for the mathematical representation of complex systems as diverse as biological, social and information systems (metabolic reaction and ecosystems networks, graph of scientific citations and collaborations, the internet and the world wide web, etc.) (see e.g. Albert and Barabási, 2002; Dorogovtsev and Mendes, 2013). The observation of these empirical networks reveals the presence of a common feature. Actually, many of them are modular, which means that they can be separated into units in which intra connections are dense while inter connections are sparse. In some cases, these units can also be seen as elementary blocks assembled in more complex structures and performing independently (see e.g. Hartwell et al., 1999; Wagner and Altenberg, 1996).

This widespread feature of networks has important consequences in case of evolution. Indeed, the subdivision of networks into modules makes them more evolvable, i.e. more capable of rapid adaptation when they are confronted with new environments. It also offers computational advantages, intuitively considering that it is easier and less costly to rewire subunits than a whole network. Moreover, it makes networks more resilient, as the damaging of one particular module responsible for one given goal does not affect the achievements of other goals performed by other modules.

Despite its advantages, modularity remains a controversial issue, with disagreement concerning the nature of the modules that exist as well as over the reason of their appearance in real networks (see e.g. Wagner et al., 2007). It has also been abundantly studied in the context of systems design with current models of biological evolution, in which systems are evolved to attain defined goals by means of evolutionary techniques. However, networks obtained with these techniques are usually intricately wired and non-modular (see e.g. Eick et al., 2001; Thompson, 2012) and there is no consensus about the necessary conditions for the emergence of modularity.

Whereas most hypotheses assume indirect selection for evolvability, authors of some articles (see e.g. Bullinaria, 2007) suggest that the emergence of modularity might depend on different external factors such as the learning algorithm, the effect of physical constraints and the tasks themselves. For example, Lipson et al. (2002) have found that switching between goals leads to the spontaneous evolution of modular networks. Nevertheless, this condition is not strong enough for Kashtan and Alon (2005) who suggest that modularity appears when the learning process switch between two tasks with common subgoals. On the contrary, modules are lacking when the second task is random, and, consequently, independent from the first one. As for Clune et al. (2013), they claimed that the pressure to reduce the cost of connections between network nodes causes the emergence of modular networks. On his side, Valverde (2017) studies the so called breakdown of modularity theory to understand the abrupt transition from high to low modularity in evolved networks. He asserts that the decrease of modularity takes place when the evolutionary goal is non-separable, i.e. when the computation of the outputs requires the interaction of several inputs. He also claims that evolution can not design modular networks under functional and cost constraints.

In this work, we study the possible emergence of modularity in the domain of evolutionary learning in which artificial neural networks, considered as computational systems, are trained by genetic algorithms in order to learn particular tasks from experience. With this aim, we focus on two simple applications in evolutionary robotics and pattern recognition and we investigate them by comparing the performance of modular and non-modular networks and by deeply analyzing modularity in evolved networks.

Structure of the thesis

The manuscript is organized as follows. The three first chapters present the theoretical background of this work while the two following chapters deal with the presentation of our applications and results.

Chapter 1 presents artificial neural networks. We introduce their biological inspiration and their general functioning. Then, we present features often met in these networks before focusing on the model of perceptron used throughout this work. This chapter also gives us the opportunity to provide some tools for the analysis of modularity in evolved neural networks.

In Chapter 2, we introduce genetic algorithms. We first describe the general principle of these algorithms and their operators for one objective function. Then, we present two families of multiobjective genetic algorithms. The first one consists in optimizing one single fictive fitness built as the weighted sum of the different initial objectives. The second one is based on the notion of Pareto optimality. We finally get some tools for analyzing the efficiency of our algorithms.

Chapter 3 makes a connection between the tools developed in the two first chapters by leading the reader in the field of evolutionary learning. We detail the concepts of learning and evolutionary learning. As neural networks can be trained by other optimization techniques than genetic algorithms, we provide two of these algorithms

that will be used for comparisons throughout our work. We close the chapter by some comments on general aspects of our numerical simulations.

Chapter 4 proposes a deep study of a trial application in the domain of evolutionary robotics. We first describe the tasks to achieve and the experimental settings. We then study in detail results got with two different neural networks models. This study is divided in three parts. The first part consists in a description of preliminary results on the learning process and a comparison between different multiobjective genetic algorithms. The second part focuses on a comparison of learning sequences. In the last part, we study the emergence of modularity in evolved networks.

Chapter 5 presents a second application in the field of pattern recognition. In a first time, we describe the problem, which has already been studied in previous works (Clune et al., 2013; Kashtan and Alon, 2005). Then we try to replicate and to extend results of the latter reference article. As results are not those expected, we present and discuss them in detail.

We close this work by giving some concluding remarks and providing some interesting perspectives for future work.

Contributions

As previously mentioned, this thesis brings several contributions to the domain of evolutionary learning by studying applications in the frameworks of evolutionary robotics and pattern recognition. These main contributions are summarized in the following.

Each application has been studied thanks to numerical codes we personally built. We have thus implemented GA adapted to our different models of neural networks, as well as routines for the initialization of experimental settings and the evaluation of performance functions. Many routines have also been developed in order to analyze the resulting networks.

In previous work, Calabretta et al. (2008) have shown that a non-modular neural network can be trained so as to achieve the same performance as a modular one in accomplishing multiple classification tasks. In Chapter 4, we extend this work in the more general and the more complex context of robotic tasks. Chapter 4 also proposes a comparison between distinct multiobjective genetic algorithms and a study about the possible ways of decreasing the number of links in the evolved networks.

Chapters 4 and 5 provide a deep analysis of modularity emergence. For the application described in Chapter 4, whose tasks are more complex than those generally studied in the literature, it is shown that evolved networks are non-modular and that even initial modular networks rapidly evolves into non-modular ones. In Chapter 5, we have tried to replicate the research of Kashtan and Alon (2005) in which they show that a repeated switch between different goals with common subgoals lead to the emergence of modularity. However, we have obtained opposite results and we have concluded that the emergence of modularity is very sensitive to the experimental settings.

Key results appearing in this document have been published in different papers, whose a detailed list can be found below.

- D. Nicolay and T. Carletti. Neural networks learning: Some preliminary results on heuristic methods and applications. In *DWAI@AI*IA*, pages 30-40, 2013.
- D. Nicolay, A. Roli, and T. Carletti. Learning multiple conflicting tasks with artificial evolution. In *Advances in Artificial Life and Evolutionary Computation*, volume 445 of *Communications in Computer and Information Science*, pages 127-139, Springer International Publishing, 2014.
- D. Nicolay, A. Roli, and T. Carletti. Does training lead to the formation of modules in threshold networks? In *Proceedings of ECCS 2014*, pages 181-192, Springer International Publishing, 2016.

Simultaneously to this main research, we have worked on two additional projects. Firstly, we have studied the possibility to make use of neural networks endowed with quantum gates linked together to design quantum operators and algorithms. We have succeeded in reproducing simple well-known quantum algorithms and quantum gates starting from a set of universal gates. We have also highlighted the limitations of our method. Results related to this research have led to the publication of one paper and are presented in Appendix A.

Secondly, we have collaborated with Alexandre Mayer, from the department of Physics, with the aim to develop a genetic algorithm for the optimization of the light-extraction efficiency of GaN light-emitting diodes (LED). Thereafter, a multiobjective version of this GA has also been used for the optimization of solar thermal collectors. This collaboration enables us to profit from the experience of other researchers in the development of GA and led to the publications of two papers. Our contribution has been limited to discussions with Alexandre about the concept of genetic algorithms and their operators. Articles resulting from this collaboration are inserted in Appendix B.

Papers presenting results of these two research projects are cited below.

- A. Mayer, A. Bay, L. Gaouyat, D. Nicolay, T. Carletti and O. Deparis. Genetic algorithms used for the optimization of light-emitting diodes and solar thermal collectors. In *Proceedings of SPIE*, 3987, 10 p., 2014.
- A. Mayer, L. Gaouyat, D. Nicolay, T. Carletti and O. Deparis. Multi-objective genetic algorithm for the optimization of a flat-plate solar thermal collector. In *Optics Express*, volume 22, pages A1641-A1649, 2014.
- D. Nicolay and T. Carletti. Quantum neural networks achieving quantum algorithms. In *Advances in Artificial Life and Evolutionary Computation*, volume 830 of *Communications in Computer and Information Science*, pages 3-15, Springer International Publishing, 2018.

Chapter 1

Artificial Neural Networks

This chapter deals with the introduction of artificial neural networks (ANN). These computational tools draw their inspiration from the functioning of biological neural networks. They receive information from their environment and produce answers according to their connections scheme and the magnitude of these weighted connections. Neural networks have important capabilities of learning and generalization. Moreover, they are fault-noise tolerant, which means that they provide accurate prediction in the presence of uncertain data and measurements errors, and that they continue operating in case of neurons or connections lost. Their attractiveness also comes from their non-linearity and high parallelism features. They are exploited in various domains such as optimization, process control, broker valuation, forecasting or approximation of complex functions (see e.g. Basheer and Hajmeer, 2000; Pfeifer and Scheier, 2001).

The first section of this chapter consists in a simplified description of biological neural networks on which the artificial neural networks are based (see e.g. Peretto, 1992; Pfeifer and Scheier, 2001; Rojas, 1996). This introduction gives us the opportunity to highlight the essential constituents of these models. Then, we introduce artificial neural networks. The last section is dedicated to the presentation of modularity, the key feature of networks we study in detail in this work.

1.1 Biological Source

The human brain is a very complex structure which consists of 10^{11} neuronal cells, called neurons. They are highly interconnected and interact to control body response to environmental changes. Although their shape varies according to their function, a common structure can be depicted (see Figure 1.1). The main components of neuronal cells are the dendrites, the cell body (soma), the axon and the synapses. The dendrites receive signals from other neurons and pass them over to the cell body which integrates them. If this accumulation is sufficiently strong, the cell body is said to fire and transmits a response to the axon. This response is then relayed to the

dendrites of neighboring neurons through the synapses.

Information is transmitted by electrical impulses. When a neuron is at rest, i.e. when it does not receive any signal, the difference of potential between the intra and extra-cellular fluid is stable. Upon arrival at synapses, an impulse leads to the release of chemical transmitters. According to their nature, the transmitters can increase or decrease the cell potential and two different reactions are then observable. Either the synapse excites the cell and helps to propagate the signal, or the probability of firing is decreased for a while and the synapse is said inhibitory. The signal goes on and propagates if the difference of potential in the interior and the exterior of the neuronal cell overcomes a certain threshold. Indeed, a displacement of ions is required to get back to equilibrium and prompts a new electrical impulse. This one, also called action potential, is represented as a wave which crosses the neuron by carrying the signal along it. Let us emphasize that an action potential is prompted by several signals with different intensities.

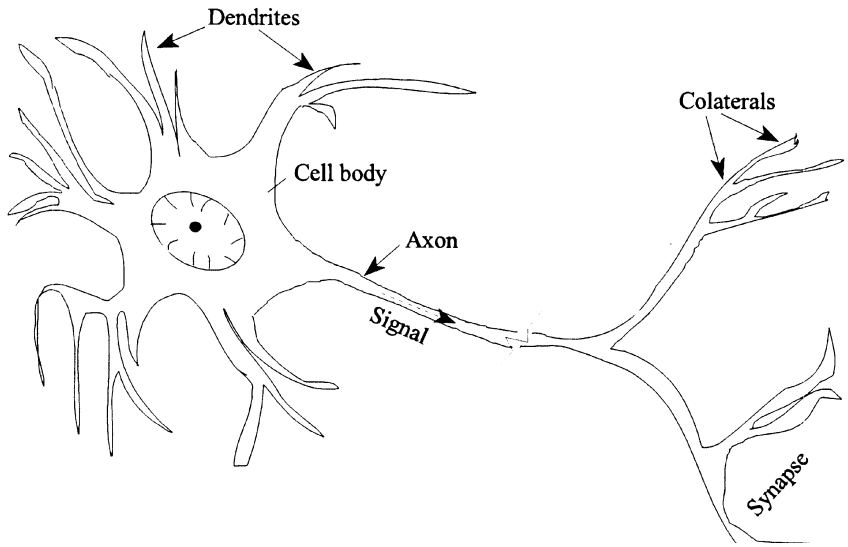


Figure 1.1: Schematic of biological neuron (Basheer and Hajmeer, 2000)

Information supplied by senses and conveyed to neuronal cells enable interactions between humans and their environment. Thanks to it, humans can learn to deal with everyday life situations. This learning is stimulated by the brain plasticity, i.e. the ability of the brain to change according to its environment (see e.g. Gerstner and Kistler, 2002). This plasticity appears in two ways with different time scales. First, neuronal connections are created, strengthened or broken depending on their usefulness. Second, the excitement threshold of neurons and the weights of synapses are modified to adapt to new situations.

To sum up, four elements are essential in the composition of artificial neurons.

These elements are the inputs, the outputs, the body of the cell and a function to decide if the neuron fires and then if the information is transmitted. Moreover, these neurons require to be linked by oriented connections to respect the directional flow of information.

1.2 Model

As mentioned in the previous section, there exists a great variety of neurons in natural systems whose shapes vary according to their functions. This diversity is reflected by the significant number of neural networks models that exist in the literature. This boom in ANN models is also explained from an engineering point of view, where more attention is dedicated to the problem solving aspect than to the possible generalization features. In fact, a model functioning well for a specific application can be ineffective in other situations. As a result, this huge quantity of models has been developed with the aim of covering a very large amount of real-world problems. Among them, we can cite precursor models such as the neuron model of McCulloch and Pitts, the associative memories of Taylor and the perceptron developed by Rosenblatt. We can also cite other models such as the Hopfield networks, the Boltzmann machines and the deep neural networks which have aroused great interest in recent years (see e.g. Prieto et al., 2016).

In this section, we limit ourselves to present the model of the perceptron and its generalizations, which are used as a reference for our ANN model. Interested readers can find more details about other existing models in the books of Rojas (1996) and Marsland (2015). First, we present a generic model and features often met in the literature. Then, we focus on the perceptron. Let us remark that this model is quite old compared to the history of ANN. Nevertheless, it is sufficiently complex to manage the applications we study.

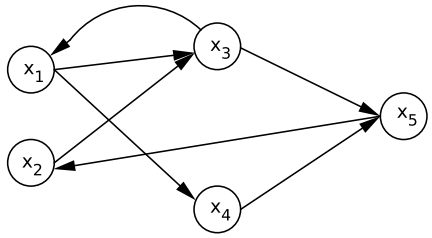
1.2.1 General Model

An artificial neuron is made of a body that receives inputs and produces an output according to its activation function. Inputs and outputs can be binary or real values. The input connections can be weighted and are summed up before being passed as argument of the activation function, which can be linear, step or sigmoid. If the input connections are weighted, negative weights correspond to inhibitory connections while positive weights match excitatory arcs. In this case, inhibition is said relative in opposition to total inhibition which is characterized by the immediate inhibition of the neuron in front of an inhibitory connection.

Artificial neural networks are built by connecting artificial neurons. They can be seen as directed graphs with nodes and arcs. They are then represented by a connectivity matrix W where w_{ij} is different from zero if the node j is linked to i (Pfeifer and Scheier, 2001). An example of such a network with its connectivity matrix is presented in Figure 1.2. Let us note that the connectivity matrix is defined as the

transpose of the adjacency matrix which is well known in graph theory. Some authors therefore use this more general concept to describe their ANN.

These networks can be classified in two ways. First, they can be separated between feedforward and recurrent networks, i.e. between networks without and with cycles. In the feedforward case, information is transmitted from the inputs to the outputs without visiting twice the same neuron. If a network is recurrent, as in Figure 1.2, some neurons can be visited more than once. The neuronal states of the network are then dynamic and the flow of information evolves before reaching its stable point. Second, networks are either synchronous if neurons fire simultaneously or asynchronous if they give their answers at random times.



(a) Neural network

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

(b) Connectivity matrix

Figure 1.2: Example of an unweighted neural network and its connectivity matrix

As their biological counterparts, the connections and weights of artificial neural networks can evolve to model a particular function. The algorithms dedicated to this evolutionary process are called learning algorithms and are divided into three classes, namely supervised, unsupervised and reinforcement learning (see e.g. MacKay, 2003). In the supervised learning, also called learning with a teacher, a series of input vectors are presented to the network that computes the corresponding outputs. These outputs are then compared to the expected answers called targets, and the weights are modified according to the errors. On the contrary, unsupervised learning is used when the targets are unknown, as in classification problems. In this case, the algorithm tries to categorize inputs by identifying similarities between them. As for the reinforcement learning, this is somewhere between supervised and unsupervised learning. Indeed, the algorithm receives a signal when the output is wrong but does not know the target. Consequently, it has to explore and try out different possibilities until it gets the right answer.

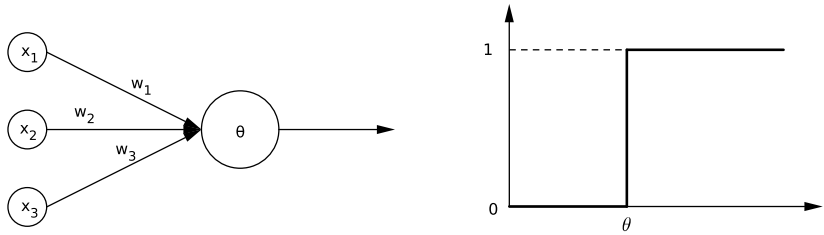
1.2.2 Perceptron

The perceptron has been introduced by Rosenblatt in 1958. This model of neuron is characterized by real valued inputs (x_i) and a binary output (y). Each input connection has a different weight (w_i), which can be positive or negative. According to its sign, the link is either excitatory or inhibitory.

The output is computed by a step activation function, which is expressed as

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i < \theta \end{cases}$$

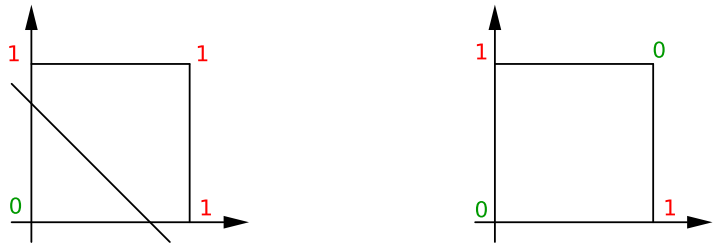
where n is the number of inputs and θ the threshold of the neuron. An example of such a neuron and its activation function is shown in Figure 1.3.



(a) Neuron (b) Step function

Figure 1.3: Representation of a perceptron and its activation function

The perceptron is suitable for solving classification problems with two classes. But it can only perform accurately with linearly separable classes, i.e. classes that can be separated by a hyperplane (Minsky and Papert, 1969). As an example, we can consider the implementation of boolean functions with two input variables. The perceptron is able to model functions *and* and *or* but can not deal with *xor* (see Figure 1.4).



(a) OR (b) XOR

Figure 1.4: Input space representation of linearly separable *or* and nonlinearly separable *xor* boolean functions

The need to cope with nonlinearly separable problems leads to the discovery of the multilayer perceptron architecture (see Figure 1.5). This network is feedforward and is made of one input layer, some additional layers, called hidden layers, and one

output layer. This model of networks is synchronous as all nodes of one layer transmit their outputs simultaneously.

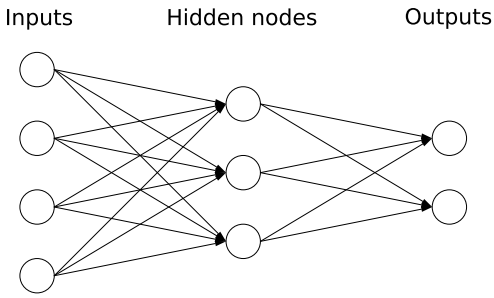


Figure 1.5: Multilayer perceptron architecture

Standard multilayer feedforward networks have been proved to form a class of universal approximators (Hornik et al., 1989). It means that the multilayer perceptron is capable of approximating any function to desired degree of accuracy providing that the number of hidden nodes is sufficiently large.

The weights of perceptrons and multilayer perceptrons can be optimized by supervised learning methods which are designed for these models, namely the perceptron rule (Rosenblatt, 1961) and the backpropagation learning (Rumelhart et al., 1986). More recently, these algorithms have been outperformed by other optimization techniques such as the stochastic gradient descent (see e.g. Shalev-Shwartz and Ben-David, 2014). As all these learning methods are not employed in this work, we refer interested readers to the above cited references for detailed presentation of these algorithms.

1.3 Modularity

As already mentioned, modularity is a widespread feature of biological and artificial networks linked to the notion of modules, also called communities. From a topological point of view, modules are sets of nodes highly connected to each other and slightly connected to the rest of the network. Figure 1.6 represents a modular network made of three topological modules marked out by dashed lines. Functional modules are sets of node that behave in a coherent and coordinated way and that loosely interact with the remainder of the network.

Topological and functional modules can match in networks, with different groups of nodes performing different functions. Nevertheless, this correspondence is not automatic and it is relevant to study them separately.

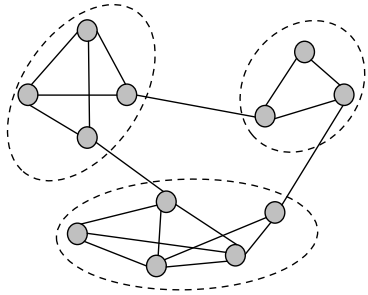


Figure 1.6: Network with three topological modules

1.3.1 Topological Modules

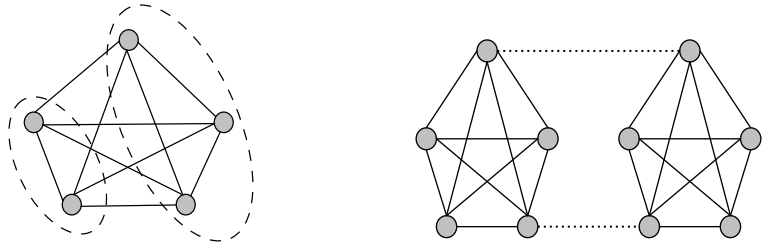
From a topological point of view, modularity is a measure of the density of links inside modules as compared to links between modules (Newman, 2006). In the case of weighted and undirected networks, the measure is defined as

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \, \delta(C_i, C_j)$$

where A_{ij} represents the weight of the edge between i and j, $k_i = \sum_j A_{ij}$ is the sum of the weighted edges linked to node i, C_i is the community to which node i is assigned and $m = \frac{1}{2} \sum_{ij} A_{ij}$. The δ function yields one if the nodes i and j are in the same community and zero otherwise. The basic idea of this measure is that random graphs are not expected to exhibit modular structures. Modularity is then computed by comparing the density of edges in communities, given by the terms A_{ij}, and the corresponding density for a random network with the same degree distribution as the original graph, computed as $\frac{k_i k_j}{2m}$. This measure returns a real value in the interval $[-1, 1]$. Values close to zero mean that there are no modules and values close to one that there are no interactions between modules. As for negative values close to minus one, they may hint to the existence of multipartite structures, i.e. subgroups with very few internal edges and many edges lying between them (Fortunato, 2010).

As an example, we compute the modularity values of the unweighted and undirected graphs drawn in Figure 1.7. The network on the left is a complete graph with five nodes (K_5). It is clearly non-modular as each node is connected to all other nodes. If we compute its modularity value by considering one unique community, we obtain a value equal to zero. If we consider two modules, represented by dashed line in the figure, the modularity is negative (-0.12), which means that it is better to keep one unique community. On the right, the network is made of two separate K_5 subgraphs. The network is then clearly modular and the modularity value is equal to 0.5 if we consider each subgraph as a community. By adding the dotted connections, the network is always modular but the modularity value is slightly lower (0.41). By comparison,

the modularity measure of a random graph with the same degree distribution is around 0.15.



(a) Complete K_5 graph (b) Separate K_5 subgraphs

Figure 1.7: Example of simple non-modular and modular networks

There are many methods and algorithms to study and compute topological modularity in a network (see e.g. Fortunato, 2010). In this work, we decided to use the Louvain method (Blondel, Guillaume, Lambiotte, and Lefebvre, 2008), illustrated in Figure 1.8.

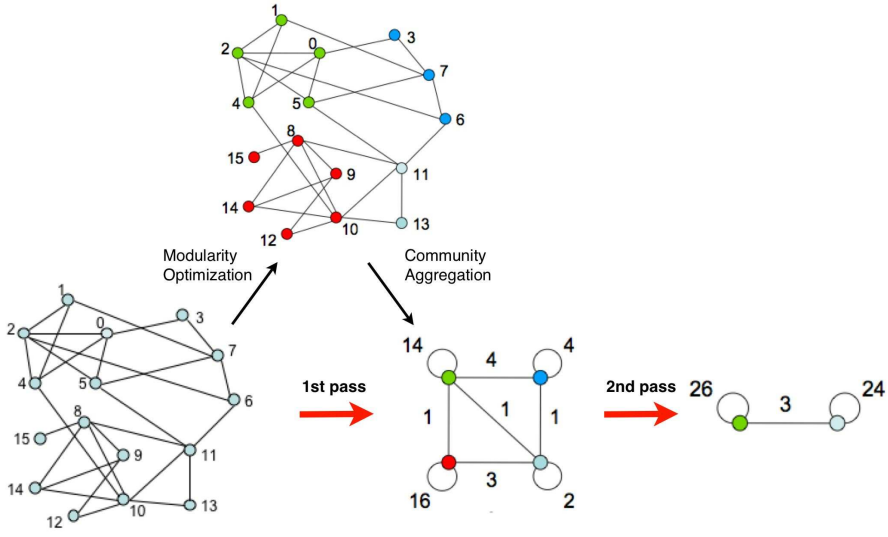


Figure 1.8: Illustration of the Louvain method (Blondel et al., 2008)

This algorithm is divided into two steps repeated sequentially. Each combination of these two steps is called a pass. Initially, all nodes are assigned to a different

community. Then, for each node i, the gain coming from putting i in a community C is computed as

$$\Delta Q = \left[\frac{\sum_{in} + 2k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right]$$

where $\sum_{in}$ is the sum of the weights of the links inside C, $\sum_{tot}$ is the sum of the weights of the links incident to node i and $k_{i,in}$ is the sum of the weights of the links from i to nodes in C. A similar expression is used to evaluate the change of modularity when i is removed from its own community and this value is substracted from the previous one to obtain the real gain. The node i is then placed in the community with the largest real gain as long as this gain is positive. This process is repeated sequentially until a local maximum of the modularity is attained. The second step consists in building a new network whose nodes are the communities found during the first phase. The weights of the links in this new network are assigned to the sum of the weighted connections between nodes of the two communities. Once this second step is completed, another pass can be applied to the resulting network. The algorithms stops when a maximum of modularity is reached.

1.3.2 Functional Modules

The study of functional modularity aims at grouping together neurons that have similar dynamic behaviors. Once more, there exist different methods to perform this analysis (see e.g. Bullmore and Sporns, 2009). In this work, we decided to use the dynamical cluster index (DCI) (Villani et al., 2013, 2015). This index is a generalization of the cluster index (CI) introduced by Tononi, McIntosh, Russell, and Edelman (1998) and relying on information theory. The difference lays in the data on which the measure is applied. Indeed, the CI is applied on fluctuations of a system around a stationary state while DCI is used on time series of data generated by a dynamical model. The difference makes the DCI able to study dynamical networks while CI is only able to analyze dynamical networks close to stationary states.

Let us consider a network U composed of n nodes whose states can assume a finite set of discrete values. The cluster index is based on the notion of entropy of single elements and sets of elements in U. The entropy of an element x_i is a measure of the unpredictability of the state, which is expressed as

$$H(x_i) = -\sum_{v \in V_i} p(v) \log p(v)$$

where V_i is the set of possible values of x_i and $p(v)$ the probability of occurrence of v. Entropy is equal to zero when one outcome is certain to occur and is high if the outcome is strongly uncertain, as in case of equiprobable states. Let us remark that, when data are furnished by observations, probabilities are estimated by means of relative frequencies. In the same way, the entropy of a pair of elements x_i and x_j, also called joint entropy, is defined thanks to their joint probabilities

$$H(x_i, x_j) = -\sum_{v \in V_i} \sum_{w \in V_j} p(v,w) \log p(v,w)$$

and this definition can be extended to sets of k elements.

The dynamical cluster index $C(S)$ of a set S of k elements is defined as the ratio between the integration $I(S)$ of S and the mutual information $M(S;U-S)$ between S and the rest of the network $(U-S)$. The integration measures the deviation from statistical independence of the k elements in S by subtracting the joint entropy of the whole set S to the sum of the single-node entropies

$$I(S) = \sum_{j \in S} H(x_j) - H(S).$$

If the k elements are completely independent, the integration is zero. If, on the other hand, they show strong deviation from statistical independence, integration is high. On its side, the mutual information is a measure of the mutual dependence between two sets. It is defined by the sum of the entropies of the two sets to which we substract their joint entropy

$$M(S;U-S) \equiv H(S) + H(U-S) - H(S,U-S).$$

This measure is equal to zero if and only if the sets are independent. The DCI is then expressed as

$$C(S) = \frac{I(S)}{M(S;U-S)}.$$

Good candidates as functional subsets are those that maximize the integration while minimizing the mutual information.

The cluster index scales with the size of S. Therefore, a normalization step is required before comparing indices of sets from different sizes. It is performed by comparing the indices with those of sets having the same size but belonging to a system which is expected to be non-modular. To build such a system given a series of data, the frequency of each value is first computed. Then, a new random series is generated considering that each value keeps the same probability of occurrence. This system, called homogeneous system, enables to calculate normalization constants for each set size. Given that $< I_h >$ and $< M_h >$ are the average integration and mutual information, the normalized cluster index of S is expressed as

$$C'(S) = \frac{I(S)/ < I_h >}{M(S;U-S)/ < M_h >}.$$

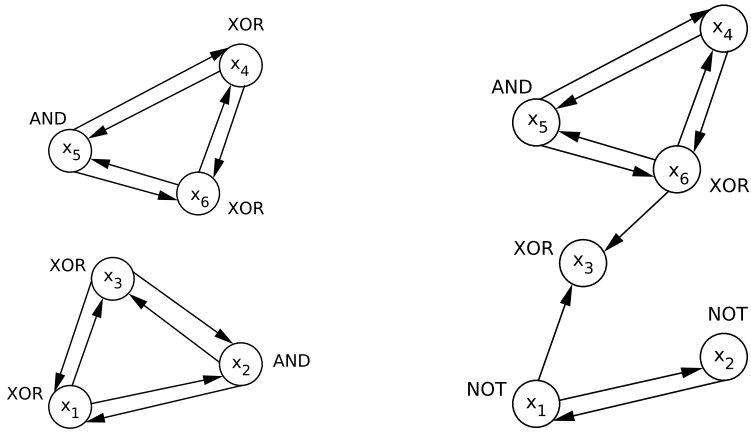
The search for functional modules consists in optimizing the dynamical cluster index. In principle, the DCI of all possible sets should be computed but this is only feasible for a limited size of sets in a reasonable time. Consequently, the idea is to sample the configurations instead of analyzing all of them. Moreover, this random sampling is assisted by a heuristic search to avoid missing sets with high index. The principle is that, once the samples of size k are evaluated, the random samples of size $k+1$ are completed with the $(k+1)$-size neighbors of the k-size sets that present the highest DCI values.

Once all the samples for size up to $n-1$ are evaluated, they are ranked thanks to statistical significance index T_c expressed as

$$T_c(S) = \frac{C'(S) - < C'_h >}{\sigma(C'_h)}$$

where $< C'_h >$ and $\sigma(C'_h)$ are the average and the standard deviation of the population of normalized cluster indices with the same size of S from the homogeneous system.

The DCI can be applied to any kind of networks, as it only uses dynamical states to find the functional modules. In the example drawn from (Villani et al., 2015) that we present here, it is used to find relevant subsets in boolean networks (see e.g. Gershenson, 2002), which are frequently used to model genetic regulatory networks. These boolean networks are made of nodes whose internal states take boolean values. The way in which nodes affect each other is determined by their connections and by logic functions. The updating of the nodes is synchronous. The dynamic evolves according to the updating functions and scheme from an initial random state. For this kind of networks, the data used to determine the relative frequencies that are necessary to compute the entropies are obtained from the state of the various attractors, each one weighted according to its basin of attraction. In this example, the DCI is analyzed for two particular boolean networks represented in Figure 1.9. The network on the left ($BN1$) is made of two independent components, while on the right ($BN2$), the components interact.



(a) $BN1$ (b) $BN2$

Figure 1.9: Boolean networks

Results of the analysis are presented in Table 1.1. Each horizontal line is associated to a subset of the whole system, and nodes present in this subset are marked by a cross. We can observe that the two separated subnetworks of $BN1$ are correctly identified. For $BN2$, five of the six nodes are clustered together, leading to the conclusion that nodes of this network are interdependent.

x_1	x_2	x_3	x_4	x_5	x_6	T_C
			×	×	×	110
×	×	×				102
	×		×	×	×	8

(a) $BN1$

x_1	x_2	x_3	x_4	x_5	x_6	T_C
	×	×	×	×	×	110
	×	×		×	×	6
	×	×		×		5

(b) $BN2$

Table 1.1: Functional modules obtained with the DCI

Chapter 2

Genetic Algorithms

In this chapter, we introduce the second mathematical tool used all along this work. Genetic algorithms (GA) are heuristic optimization methods which mimic the biological phenomenon of species evolution. This evolutionary process rests on the natural selection of individuals as well as on their reproduction and mutation. These three principles ensure essential features in the research for global optima. On one hand, they warrant the species diversification, which corresponds to a good exploration of the solutions space. On the other hand, they guarantee the selection of best adapted individuals, i.e. reaching solutions that globally optimize the problem.

Initially, we focus on the description of the simple GA and its operators. Then, we present two kinds of multiobjective genetic algorithms. The principle of the first one is to optimize a single function obtained as a weighted sum of the different initial objectives. As for the second one, it is based on the notion of Pareto optimality. A last section will then be dedicated to an analysis of the GA efficiency according to the objective function of the problem.

2.1 Simple GA

The aim of a genetic algorithm is to maximize an objective function f on its feasible set Ω.

$$\max_{x \in \Omega} f(x)$$

This optimization technique is detailed in numerous textbooks (see e.g. Eiben and Smith, 2015; Goldberg, 1989; Haupt and Haupt, 2004). In a first stage, we introduce the terminology and the general scheme of the algorithm. We also present its advantages and drawbacks. Then, we focus on each of these successive operators and detail their most known variants.

2.1.1 General Functioning

As previously mentioned, GA are inspired by biological process. It is then common to use biological terms to mention their components. A population is defined as a set of individuals. Each individual is made of a chromosome, which is generally a vector containing the parameters representing a potential solution. These parameters, also called genes, can be coded in binary form or in real form according to the type of problems. The different values taken by genes are called alleles. In terms of optimization, individuals and chromosomes are both used to mention a solution. The viability is the capacity for an individual to get used to its environment. It is expressed by the fitness, i.e. the objective function of the individual.

The general scheme of a GA is presented in Figure 2.1. An initial population is randomly generated and the fitness of each individual is assessed. A selection is then performed to choose the individuals that are considered as parents. New individuals, called children, are created by applying genetic operators of reproduction and mutation to these parents. Afterward, the next generation is constituted by selecting surviving individuals among parents and their offspring. The same procedure is iterated until the termination condition is reached. These iterations are called generations. The aim is to produce better individuals at each generation and finally reach the optimum.

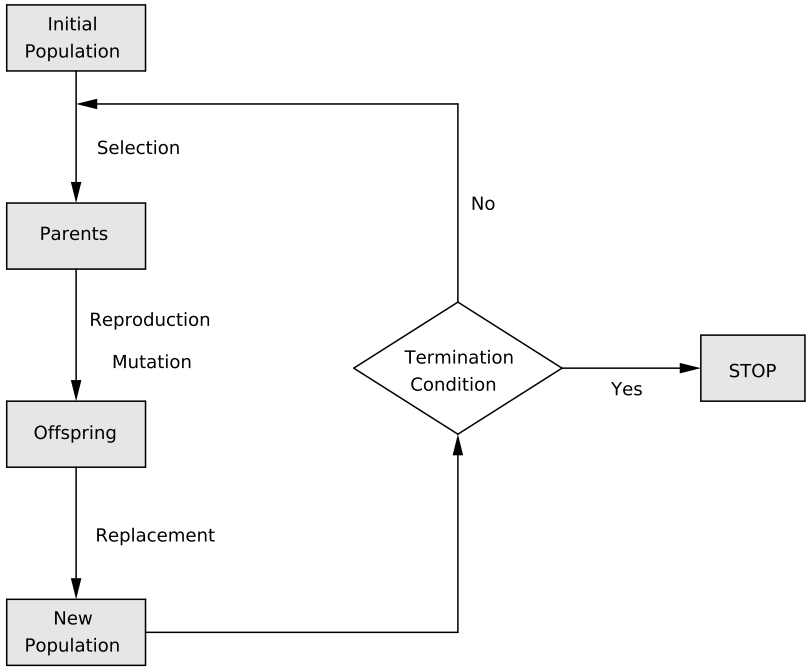


Figure 2.1: General scheme of a genetic algorithm

According to Haupt and Haupt (2004), genetic algorithms display numerous advantages. Indeed, they can deal successfully with a wide range of problem areas such as continuous, discrete or combinatorial problems. Contrary to many classical optimization methods, they do not require any knowledge about the first and second derivatives of the optimized function. Moreover, they can be applied on multimodal, noisy or strongly nonlinear functions. They also enable to solve multiobjective and constrained optimization problems. Nevertheless, genetic algorithms has a big drawback with the absence of theoretical proof of convergence. Indeed, although the literature abounds with applications that demonstrate the method effectiveness, the underlying theory is less understood and the convergence is only assured in probability (see e.g. Beasley et al., 1993a; Eiben and Smith, 2015). Consequently, GA are more costly than classical optimization methods as they require multiple simulations to obtain a satisfying approximate solution.

2.1.2 Operators

All genetic algorithms follow the scheme presented in Figure 2.1. But they can be quite different according to the ways they perform each step of the process. We detail here the most classical methods for these different steps (see e.g. Beasley et al., 1993a,b; Eiben and Smith, 2015).

Selection

The step of selection consists in choosing the best individuals for the reproduction with the aim to get individuals having better fitness over the generations. Classical processes for the selection are the roulette wheel, the stochastic remainder and the tournament.

With the roulette wheel, individuals are selected proportionally to their fitness. Each individual j has a probability p_j to be drawn which is defined as

$$p_j = \frac{F_j}{\sum_{k=1}^{m} F_k}$$

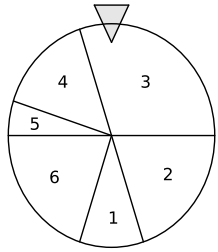
where F_j represents the fitness of individual j and m is the size of the population. To simulate such a draw, each individual j is matched with a segment of the interval $[0,1]$ whose length L_j is equal to its probability. The selection of individuals is then performed by drawing a random number between 0 and 1 and by choosing the individual associated to the segment containing the drawn number. This process is comparable to a roulette where each case represents an individual and has a width proportional to its fitness. Such a roulette is presented in Figure 2.2 for a fictive population of six individuals whose fitness are given in the table on the left.

This selection is repeated as many times as the number of individuals to select. Even if the roulette is quite simple, it has some drawbacks. On one hand, the use of small populations can make the draw badly representative and can lead to the loss of good individuals. On the other hand, this method of selection is very sensitive to

the fitness variance. Indeed, if the variance is small, the roulette is nearly similar to a random draw whereas if it is too high, the selection of best individuals can be too large and drive to premature convergence towards a local optimum. To avoid this kind of problems, individuals can be sorted according to their fitness before receiving a well-distributed fictive fitness used for the selection. For example, this fictive fitness can be their rank in the sorting. In this case, the individuals are uniformly spaced from $\frac{1}{M}$ where $M = \sum_{i=1}^{m} i$. The probability obtained with this fictive fitness is also represented in Figure 2.2.

	Fitness	Probability with fitness	Probability with sorting
1	15	0.1	0.095
2	30	0.2	0.238
3	45	0.3	0.286
4	22.5	0.15	0.143
5	7.5	0.05	0.048
6	30	0.2	0.190

(a) Fitness and probability



(b) Roulette

Figure 2.2: Roulette wheel selection

The stochastic remainder selection is an hybrid process that partially removes the drawbacks of the roulette wheel by selecting a part of the parents in a deterministic way. The number of times an individual is used during the reproduction is calculated as the integer part of the ratio between its fitness and the mean fitness on the current population. The remaining parents are then selected with the roulette wheel selection.

The tournament method consists in picking two individuals randomly. The individual with the best fitness is assigned a probability $p \in \left(\frac{1}{2}, 1\right]$ to be chosen, which means that the other one has a probability of $1-p$. There exist variants to this method. For example, the number of individuals involved in the tournament can be superior to two. The more they are, the more the selection pressure is large. In another variant, winners of tournaments are picked up by roulette wheel instead of randomly.

Reproduction

The reproduction is the operator which enables the combination of genetic information from parents. It is performed by a method called the crossover whose aim is to generate two children from two parents by crossing the chromosomes of parents, in such a way that children hold genes from both parents. There exist different types of crossover such as the 1-point crossover, the n-point crossover and the uniform crossover.

The 1-point crossover is the simplest version of the crossover in which parents chromosomes are cut at a random point and exchange their end to create children. This crossover is represented in Figure 2.3. The process can be generalized to n cutting

points. The chromosomes are then divided into $n+1$ fragments whose one on two is exchanged to obtain children. This process is logically called the n-point crossover.

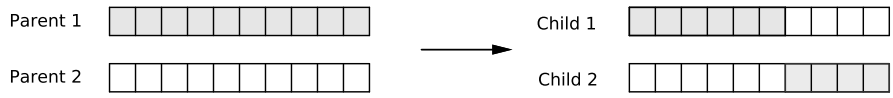


Figure 2.3: 1-point crossover

Concerning the uniform crossover, each gene of the first child is created by copying the corresponding gene from one of the two parents according to a randomly generated binary crossover mask. If the value corresponding to a gene is one, the gene is copied from the first parent. Otherwise, it is copied from the second parent. The process is then repeated after exchanging the value associated to each parent for the creation of the second child. An example of such a crossover is presented in Figure 2.4.

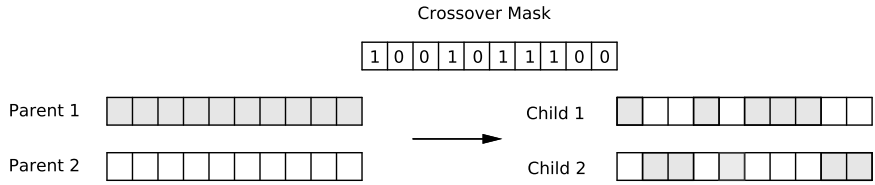


Figure 2.4: Uniform crossover

The effect of the reproduction operator is not always beneficial. Indeed, if an individual is close to the optimal solution and the other is very far from it, their crossover can lead to average children. To reduce this negative effect, selected individuals have a given probability to be crossed, which is called the crossover rate.

Mutation

The mutation operator slightly transforms the chromosomes in order to explore the solutions space and keep diversity in the population. If the genes are coded in binary form, each bit has a probability p to flip which is assumed to be independent from other bits. It means that the probability of mutation for a bit follows a Bernouilli law with parameter p. Consequently, the total number of mutations follows a Binomial law with parameters N_G and p, where N_G is the total number of bits calculated as the product between the size of the population, the number of genes in one chromosome and the number of bits on which each gene is coded. The mean number of mutations

completed at each step is then approximated by

$$n_{mutation} = N_G \times p$$

The mutation probability of genes is called the mutation rate. A similar process can be applied if genes are coded in real form. In this case, the mutation can consists in replacing the gene with a random value from the permissible set or in slightly modifying the gene by adding a small positive or negative value.

Replacement step

The reproduction and the mutation steps can be applied on the same or on different selected individuals. Assuming that the population is constant along the optimization process, it is important to choose which individuals among parents and children will be kept for the next generation. Contrary to parents selection, the survivors selection, or replacement step, is often deterministic. It can be based on the fitness values or on the concept of age. In the first case, a common method consists in ranking the unified set of parents and offspring and selecting the m best among them. In the second case, the idea is that the new generation, i.e. the offspring, entirely replaces the old one. In this case, the concept of elitism can be used to avoid the disappearance of best individuals by keeping them and inserting them into the next generation.

Termination condition

Ideally, the algorithm terminates when the optimal solution, or a good approximation, is achieved. But genetic algorithms are stochastic and reaching such an optimum is not guaranteed. Therefore, the termination condition has to be extended with a criterion that surely stops the algorithm. Criteria commonly used for this purpose are limited numbers of generations or fitness evaluations and maximally allowed CPU time. It is also considered to stop the algorithm when the fitness stagnates for a given number of generations.

2.2 Multiobjective GA

In many real-world applications, the optimization handles more than one objective function. For example, the aim of a firm is to maximize its income while minimizing the production and transportation costs. These different objectives often conflict and the search for solutions is equated with the investigation of a tradeoff between them. There exist different kinds of GA to solve such multiobjective problems. They are split into two main approaches (see e.g. Deb, 2001; Fonseca and Fleming, 1995).

In the first approach, the different objective functions are combined to create a new function. It is performed by normalizing the objectives and by assigning a weight to each of them before calculating their sum. The new function f_{new} is expressed as

$$f_{new} = w_1 f'_1 + w_2 f'_2 + \ldots + w_n f'_n$$

where f_i' is the i-th normalized function and $\sum_{i=1}^{n} w_i = 1$ The optimization is then performed on this new function by applying a simple GA.

The second approach is based on the notions of dominance and Pareto optimality. A candidate solution x is said to dominate a candidate y if x is as good as y for all objective functions and if x is strictly better than y for one of them. In mathematical terms, it is expressed as

$$x \succ y \quad \Leftrightarrow \quad \begin{array}{ll} \forall\, i = 1,\ldots,k & f_i(x) \geq f_i(y) \\ \exists\, j \in \{1,\ldots,k\} & f_j(x) > f_j(y) \end{array}$$

in the case of a maximization. A candidate solution belongs to the Pareto optimal set if it is not dominated by any other candidates. To this set corresponds a set in the objective functions space called the Pareto front and illustrated in Figure 2.5. Candidate solutions A and B are non-dominated and belongs to the Pareto optimal set. The Pareto front is represented by red circles.

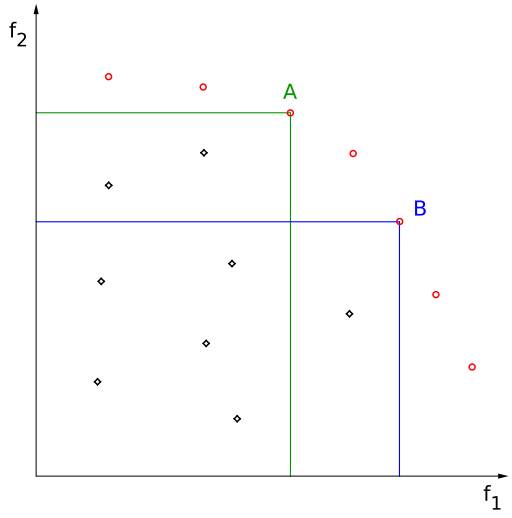


Figure 2.5: Pareto optimality front

Different methods have been developed to promote the generation of multiple non-dominated solutions. Among them, some methods make indirect use of the Pareto optimality definition (Schaffer, 1985; Kursawe, 1991). Others use the principle of non-dominated sorting (Goldberg, 1989; Fonseca and Fleming, 1993; Srinivas and Deb, 1994) which consists in giving a rank to individuals according to their non-dominance before assigning them a fitness, point from which the methods differ. For this work, we decided to use the NSGA developed by Srinivas and Deb (1994) and detailed below.

Individuals are first sorted by rank. Non-dominated individuals have a rank of 1. Individuals that are dominated by individuals of rank 1 and that dominate the rest of

the population have a rank of 2 and so forth. The fitness is then computed by rank. Firstly, individuals of rank 1 receive a temporary fitness equal to the size of the population. In a second step, individuals that are close to other individuals with the same rank are penalized by a sharing function to avoid a premature convergence towards a unique solution of the Pareto optimality set. After the use of the sharing function, i.e. once the final fitness have been computed for individuals of rank 1, individuals of rank 2 are assigned a temporary fitness inferior to the smallest final fitness assigned to individuals of the previous rank. The sharing process is then performed again for this rank and so forth.

The sharing is based on the computation of the distance between pairs of individuals from the same rank. This distance is expressed as

$$d_{ij} = \sqrt{\sum_{p=1}^{P} \left(\frac{x_p^{(i)} - x_p^{(j)}}{x_p^u - x_p^l} \right)^2}$$

where $x_p^{(i)}$ is the allele of the p-th gene for individual i and P is the total number of genes in the chromosome of each individual. As for parameters x_p^u and x_p^l, they respectively represent the upper and lower bounds of values taken by x_p. These bounds are calculated on the entire population. Each distance is then compared to a parameter σ_{share} in order to compute its associated sharing function which is defined by

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right)^2 & \text{if } d_{ij} \leq \sigma_{share} \\ 0 & \text{otherwise} \end{cases}.$$

Assuming that the ongoing rank is composed of n individuals, a niche count is then computed for each of them as

$$m_i = \sum_{j=1}^{n} Sh(d_{ij})$$

and their final fitness is obtained by performing the following modification

$$f_i \rightarrow \frac{f_i}{m_i}.$$

This sharing process requires to assign a value to σ_{share}. According to Deb and Goldberg (1989), a good estimation of this parameter is given by

$$\sigma_{share} \approx \frac{0.5}{\sqrt[P]{q}}$$

where $q \approx 10$.

Let us note that the distance used for the sharing is based on the genes space. Therefore, penalized individuals are those that present similar alleles. It is also possible to consider a distance in the objective space. In this case, individuals are penalized if they have close objective functions.

The advantage of the functions aggregation approach is that a simple GA is sufficient to optimize the problem. But it is also a drawback as only one solution of the Pareto optimality set is reached for each running of the algorithm. Moreover, this solution is strongly dependent from the weights provided by the user. Consequently, it is necessary to run multiple simulations to avoid missing interesting solutions. The main advantage of the second approach is to converge towards a Pareto set which is close to the real one. It enables the user to compare the different solutions and to choose the most convenient for its needs. However, even if the steps of selection, reproduction and mutation are similar to those of classical GA, the running time is longer due to the sorting and the sharing process.

2.3 Fitness Analysis

With the no free lunch theorem, Wolpert and Macready (1997) showed that it does not exist an optimization algorithm better than random search for all existing problems. Different features can influence the results, such as the parameters of the optimization method, the size of the problem and so forth. The objective function has also a significant impact in the search for optimal solutions. We present here two measures linked to this objective function that can give us information about the GA efficiency (see e.g. Hoos and Stützle, 2004).

2.3.1 Fitness-distance Correlation

As previously mentioned, the objective function value of a candidate has an impact in the research for optimal solutions. Indeed, it determines the choice of parents and so, guides the optimization procedure. Therefore, the guidance is more effective if candidates with high objective functions are close from the optimal solution.

The fitness-distance correlation coefficient (FDC coefficient) is a measure of the correlation between the objective function of a candidate and its distance to the optimal solution (Jones and Forrest, 1995). Given a sample of m candidate solutions $\{s_1,\ldots,s_m\}$ and their associated fitness-distance pairs $\{(f_1,d_1),\ldots,(f_m,d_m)\}$, the FDC coefficient r_{fdc} is computed as

$$r_{fdc} = \frac{\widehat{Cov}(f,d)}{\widehat{\sigma}(f).\widehat{\sigma}(d)}$$

where $\widehat{Cov}(f,d)$ is the covariance of the pair (f_i,d_i) while $\widehat{\sigma}(f)$ and $\widehat{\sigma}(d)$ are the standard deviations of the sets $F=\{f_1,\ldots,f_m\}$ and $D=\{d_1,\ldots,d_m\}$ respectively. These three terms are mathematically defined as

$$\widehat{Cov}(f,d) = \frac{1}{m-1}\sum_{i=1}^{m}(f_i-\bar{f})(d_i-\bar{d})$$

$$\widehat{\sigma}(f) = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(f_i-\bar{f})^2} \qquad \widehat{\sigma}(d) = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(d_i-\bar{d})^2}$$

where $\bar{f}$ and $\bar{d}$ are the averages of F and D.

According to its definition, the FDC coefficient is a real value in the interval $[-1,1]$. A large positive value indicates that fitness and distance are correlated while a value close to zero means that the evaluation function does not provide any guidance towards optimal solutions. In case of negative value, the evaluation function is actually misleading (Hoos and Stützle, 2004).

2.3.2 Autocorrelation

The ruggedness of the search landscape, which describes the degree of variability between the objective functions of neighboring candidates, is another feature that influences the performance of genetic algorithms. It is intuitively related to the number of local optima, as landscape with a high density of local optima are very rugged while smooth landscapes are expected to have fewer local optima. Consequently, smoother landscapes enable genetic algorithms to explore larger parts of the search space before hitting local optima whereas rugged landscape are harder to optimize.

Different ways have been considered to define the concept of landscape ruggedness (Angel and Zissimopoulos, 2000; Stadler, 1995; Weinberger, 1990). The possibility presented in this work has been developed by Weinberger (1990) and measures the correlation between fitness of neighboring candidates at a fixed distance i. It is based on a random walk starting from a random initial candidate solution, moving successively to neighboring candidates and generating a sequence of m fitness values $(f_1,\ldots,f_m)$. The sequence of measures $r(i)$, called the autocorrelation function of the walk, is then determined as

$$r(i) = \frac{\frac{1}{(m-i)}\sum_{k=1}^{m-i}(f_k-\bar{f}).(f_{k+i}-\bar{f})}{\frac{1}{m}\sum_{k=1}^{m}(f_k-\bar{f})^2}$$

where $\bar{f}$ is the average of the observed fitness values.

The most important correlation is the first-order correlation, $r(1)$, because it represents the dependency between the fitness of a candidate and its direct neighbors. Values close to one indicate that neighboring candidates tend to exhibit similar fitness, which means that the landscape is smooth. On the contrary, values close to zero indicate the absence of correlation between fitness of neighboring candidates. In this case, the landscape is very rugged and the problem is hard to solve for a genetic algorithm. Other orders of correlation $r(i)$ with $i \neq 1$ bring information about the speed decay, which is generally exponential.

Chapter 3

Evolutionary Learning

Machine learning is a field of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed (Samuel, 1959). This chapter is dedicated to a presentation of this concept of learning. It also marks the transition between the part where we describe the tools we develop and their use and the chapters dealing with the presentation of our research work. It is organized as follows.

The aim of the first section is to make a link between artificial neural networks and genetic algorithms that have been described in the two previous chapters. For this, we begin by introducing the basic concepts of machine learning (see e.g. Alpaydin, 2010; Marsland, 2015; Mitchell, 1997) and illustrating them in our research areas, i.e. robotics and pattern recognition. Then, we describe the framework of evolutionary learning (EL), in which optimization techniques based on principles of natural evolution, called evolutionary algorithms (EA), are used to train ANN for predefined goals (see e.g. Yao, 1999).

In front of the promising results obtained in this framework, other global optimization techniques such as ant colony or tabu search have also been considered to train artificial neural networks (see e.g. Blum and Socha, 2005; Sexton et al., 1998). The second section is dedicated to the description of such optimization algorithms that will be used in this work.

Our research requires numerous computational simulations and analyses. Even if we have worked on different domains of applications, some aspects of these computations are constant throughout our work. We conclude this chapter by presenting these shared features. This presentation also give us the opportunity to differentiate our personal implementations from open source codes used in this work. In the same way, we make a distinction between results obtained by ourselves or thanks to our collaborators.

3.1 Machine Learning

Machine learning consists in programming computational systems in order to optimize performance criteria using example data or past experience. Indeed, a system is said to learn from experience E, with respect to a task T and a performance measure P, if its performance for task T, as measured by P improves with experience E (Mitchell, 1997). Once trained, the system is either predictive and can be used to make predictions on the future, or descriptive and gain knowledge from data, or both. This learning is required when human expertise does not exist or can not be explained, or when the system is in a changing environment.

In this thesis, we have worked on applications in areas of robotics (see e.g. Pfeifer and Scheier, 2001) and pattern recognition (see e.g. Bishop, 2007) by considering ANN as our computational systems. In the domain of robotics, ANN are used as controllers for robots which are made of sensors and actuators. Their sensors enable them to perceive their environment and transmit this information to the input nodes of the ANN. According to its architecture and its connections weights, the network computes output nodes and controls the robot actions. This process is sketched in Figure 3.1. Numerous experiments have been performed on various tasks such as collision-free navigation, homing, predator-prey coevolution and so on (see e.g. Floreano and Keller, 2010). In pattern recognition, input nodes of ANN receive inputs from a retina and give an answer according to the presence or lack of known patterns.

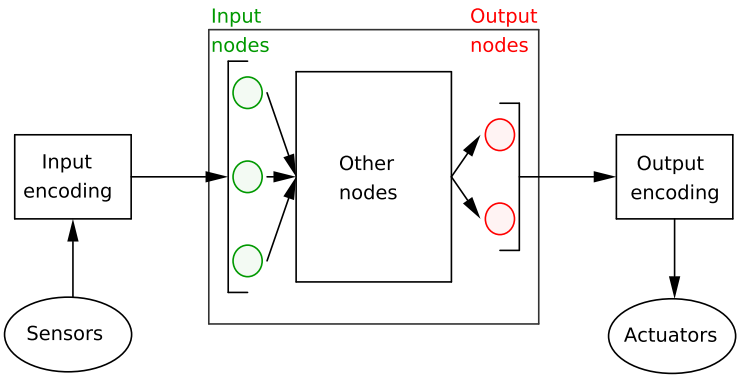


Figure 3.1: Schematic representation of ANN controllers in robotics

Let us emphasize that these two applications are quite different. Indeed, the domain of pattern recognition is part of the supervised learning, as the expected outputs are known for the learning. On its side, a robot controller only receives a signal if the answer is wrong, without information about the way to improve it. It is then part of the reinforcement learning.

Even if learning is often formulated as an optimization problem, it is quite different. In particular, the final system is asked to have good generalization, i.e. good

behavior when faced to a new environment. In a first stage, we present the essential elements for the design of a learning problem. Then, we introduce how we can use evolutionary optimization techniques in this learning process.

3.1.1 Learning

In order to design a well-defined supervised learning problem for a given task, two features have to be clearly identified, namely the training set and the performance measure. As indicated by its name, the training set is the ensemble used to train the system. The choice of this set is crucial and can have a significant impact on the success or the failure of the learning phase. Indeed, it has to represent a good sample of all the possible situations. Otherwise some information can be lacking and the generalization is not guaranteed. The performance measure is the measure used to assess the performance of the final system. In some cases, it is the function optimized during the learning process. In pattern recognition, the training set is composed of a subset of existing patterns and the performance function is generally given by the fraction of correct recognitions.

This description can be extended in the context of reinforcement learning, for which basic ideas are similar but terminology differs. Reinforcement learning is described in terms of interaction between an agent, the system that is learning, and its environment, where it is learning. The environment provide information about the efficiency of the agent strategy for a given task through some reward function. This reward function must be perfectly tuned for the task. Without this, the skill of the final agent can be compromised. This function is not always easy to conceive, especially in the field of robotics.

For the sake of simplicity, we decide to use a shared terminology for our two applications. Through misuse of language, we then keep the terms of supervised learning, i.e. training set and performance function for the application in the domain of robotics.

When a system is trained, its generalization abilities are checked by exposing it to new data forming the so called validation set, that helps to fix parameters such as the time needed for the learning. Indeed, if it is trained for a too long time, the system can have excellent performance on the training set but generalizes poorly to new environments. This phenomenon is called the overfitting or the overtraining and is schematically represented in Figure 3.2. The black, green and red lines represent the accuracy on the training set and two different validation sets. If we observe a behavior similar to the green line, the system has a good generalization ability. With the red line, the system is said to overfit as the accuracy grows for the training set while decreasing for the validation set. A last set, called test set, is used to evaluate the performance function.

The search for overfitting avoidance is one of the main reason why the performance function is not always used as the optimized function during the training. In supervised learning, we often minimize a cost J such as the cross-entropy cost or the squared-error cost (see e.g. Marsland, 2015). In reinforcement learning, a regularization technique (see e.g. Shalev-Shwartz and Ben-David, 2014) is often used to prevent the neural networks from sticking too much to the data of the training set. It consists

in penalizing the performance function by adding a multiple of the L_1 or L_2 norm of the networks weights and thresholds.

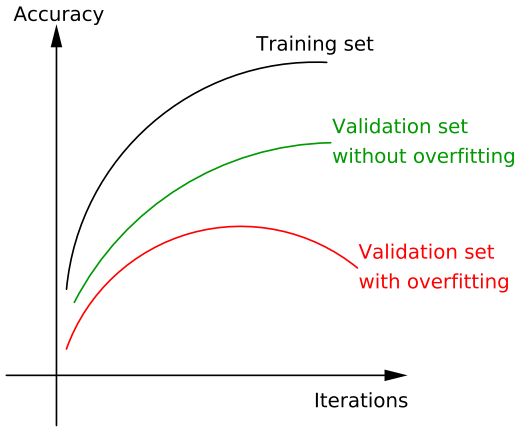


Figure 3.2: Illustration of generalization ability and overfitting

3.1.2 Evolutionary Learning

In evolutionary learning (EL), the classical learning algorithms of ANN are replaced by global optimization techniques based on principles of natural selection. This general process is drawn in Figure 3.3.

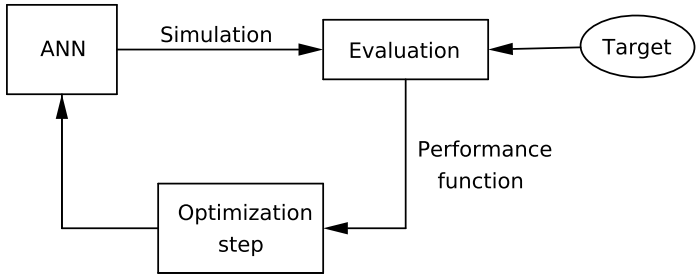


Figure 3.3: General process of evolutionary learning

A population of ANN is created and evaluated thanks to the performance function of the desired target task. Then, this population is modified according to the principle of the chosen optimization algorithm and is again submitted to the performance function evaluation. This process is repeated until the termination criterion is reached.

Contrary to classical learning methods, here the process does not modify the parameters of a given system, but it acts on a population of new instances of the generic system.

These evolutionary algorithms, including GA, display numerous advantages compared to classical learning algorithms of ANN. Indeed, they do not require any knowledge about the derivatives, are able to deal with multimodal functions and prevent ANN from falling in local optima of the fitness landscape. Moreover, EA can train the connections weights while designing the architecture. Evolutionary learning is applied in various areas such as classification, robotics, pattern recognition, VLSI systems and board games players design, and so on (see e.g. Yao, 1999).

3.2 Optimization Techniques

Evolutionary optimization techniques are not the only global optimization methods that can be considered for the training of ANN. Indeed, other techniques such as ant colony, particle swarm, simulated annealing or tabu search have been studied for this role in the literature (see e.g. Blum and Socha, 2005; Boese and Kahng, 1993; Garro et al., 2009; Sexton et al., 1998). We present now two techniques, namely random search (RS) and simulated annealing (SA), used at different steps of our research. Contrary to genetic algorithms, these techniques are trajectory methods, i.e. the search process is characterized by the trajectory of a unique solution in the search space.

3.2.1 Random Search

Random search (see e.g. Andradóttir, 2006) is the simplest optimization method that can be imagined. Indeed, it simply consists in picking randomly a candidate solution in the search space, or in a given neighborood of the current solution, and evaluating the objective function on it. This process is repeated until the termination condition is reached and the candidate with the best fitness is kept as solution. As for GA, the termination condition can be linked to the achievement of a sufficiently good approximate solution, a stagnation of the best fitness or a limited number of iterations. The pseudocode of this method is given in Algorithm 1.

Algorithm 1 Random search

$x \leftarrow x_0$
$T \leftarrow T_0$
while *termination condition(s) not met* **do**
  $x' \leftarrow$ candidate
  $\Delta f \leftarrow f(x') - f(x)$
  if $\Delta f \geq 0$ **then**
    $x \leftarrow x'$
  end
end

This optimization technique is not very efficient as its convergence time grows exponentially with the size of the search space. In this work, it will only be used to evaluate the efficiency of other optimization methods with respect to a bare model.

3.2.2 Simulated Annealing

The simulated annealing method (see e.g. van Laarhoven and Aarts, 1987) is inspired by the annealing of solids, a process in which solids are melted and then cooled down slowly in order to obtain perfect structures. Its fundamental idea is to escape from local optima by accepting at some points solutions with worse fitness than the current solution. The probability of such acceptance is decreased during the search.

The SA process is described in Algorithm 2. It starts by generating a random initial candidate solution x and by initializing a parameter T called the temperature. Then, a candidate solution x', neighbor of x, is randomly sampled and is accepted as new current solution according to the following probability function

$$P_{accept}(T,x,x') = \begin{cases} 1 & \text{if} f(x') \geq f(x) \\ \exp\left(\frac{f(x')-f(x)}{T}\right) & \text{otherwise} \end{cases}$$

which is called the Metropolis condition (Hoos and Stützle, 2004). Finally, the temperature is decreased by replacing T by kT where k is the cooling parameter and is included in $[0,1[$. The same process is repeated until the termination condition is reached. The possible conditions are the same as for the random search.

Algorithm 2 Simulated annealing

```
x ← x_0
T ← T_0
while termination condition(s) not met do
    x' ← neighbor of x
    Δf ← f(x') − f(x)
    if Δf ≥ 0 then
        x ← x'
    else
        p ← random number ∈ ]0,1[
        if p > e^(Δf/T) then
            x ← x'
        end
    end
    T ← kT
end
```

3.3 Comments on Computations

We come to the end of the theoretical part of this thesis as all concepts required for a good understanding of our research work have been described in these three chapters. Before beginning the presentation of our applications and results in evolutionary learning, we would like to comment about some general aspects of our numerical simulations.

All our computations are performed with the MATLAB software and long simulations are run by using the computational resources of the "Plateforme Technologique de Calcul Intensif (PTCI)" located at the University of Namur, Belgium, which is supported by the F.R.S.-FNRS. As the optimization process is stochastic, each experiment is independently replicated ten times and results are reported in terms of means and standard deviations.

The ANN used in this work are based on the model of perceptron and are encoded by their connectivity matrix. In all experiments, self-loops are prohibited, which allow us to use a compact notation where the diagonal of the matrix encoding the connectivity can be used to store the threshold of each node. These ANN are trained to achieve particular tasks, which consists in finding the suitable topology and the appropriate weights and thresholds to obtain networks responsible for good behavior. We resorted to GA to perform this optimization heuristically.

The GA associated with our neural networks is adapted to their representation by their connectivity matrix. The selection is performed by a roulette wheel selection. The crossover operator is a 1-point crossover compatible with the form of our individual, i.e. we randomly select a column of the matrix and we switch the following column between parents to obtain children, as represented in Figure 3.4. Each element of the children matrices has a given probability to be mutated. A deep analysis of the value assigned to crossover and mutation rates have been carried out in our master thesis (Nicolay, 2012). They are slightly different according to the studied application and the used ANN model. The population size is 100 and the replacement step is based on fitness values to ensure the legacy of best individuals. Populations of parents and offspring are thus compared at each generation before keeping the best individuals among both generations. New random individuals (one tenth of the population size) are also introduced at each generation by replacing worst individuals in order to avoid premature convergence.

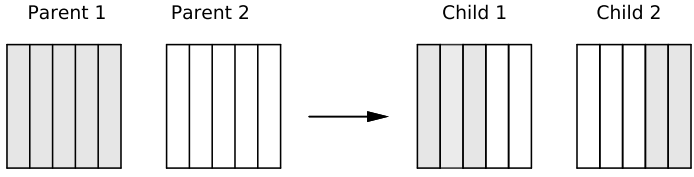


Figure 3.4: Schematic of the 1-point crossover for the connectivity matrices

Concerning the analyses made on our networks, the study of topological modularity has been accomplished with the help of two toolboxes. The first one is called *Matlab Tools for Network Analysis* and has been developed by the Strategic Engineering Research Group (SERG) of the MIT. The second one is the *Brain Connectivity Toolbox* developed at the University of Indiana. The study of functional modules with the DCI has been carried out by our collaborator Andrea Roli on the basis of data series arising from our evolved ANN. For the analyses of fitness, the implementation of the fitness-distance correlation has been made by in-house. As for the autocorrelation, it has also been analyzed thanks to numerical codes provided by Andrea.

Chapter 4

Application in Robotics

The first application we focused on belongs to the field of robotics. In the literature, the exploitation of evolutionary learning techniques to robotics is commonly named evolutionary robotics (ER). As introduced in the previous chapter, it is mainly concerned with the artificial evolution of robots neural controllers (see e.g. Nolfi and Floreano, 2000). ER has contributed with notable results to the domain of artificial cognition. It also provides a viable way for designing robotic systems, such as robot swarms. Nevertheless, in spite of promising initial results, ER has not completely expressed its potential, especially in the design of complex robotic controllers.

A case in point is the issue of training a robot controller such that the latter is able to accomplish more than one task at the same time. The experience in robotic learning already possesses some good practice and specific methodologies, such as robot shaping (see e.g. Dorigo and Colombetti, 1998), which is primarily applied to incremental training of robots. However general results in this context are still lacking. A prominent study is the one by Calabretta et al. (2008), in which the question as to whether a non-modular neural network can be trained so as to achieve the same performance as a modular one in accomplishing multiple classification tasks is addressed. This work is particularly meaningful because it shed light on the mechanisms underpinning the learning sequence that leads to the best performance, which is comparable to the one attained by a modular neural network trained independently on the two tasks. In this work, we address the question in a more general context.

Another point of interest is the study of modularity in robot neural controllers. As already mentioned, there is no consensus concerning the necessary conditions for the appearance of modules in the context of systems design. In this work, we address this question in the framework of evolutionary robotics by considering hypotheses often met in the literature.

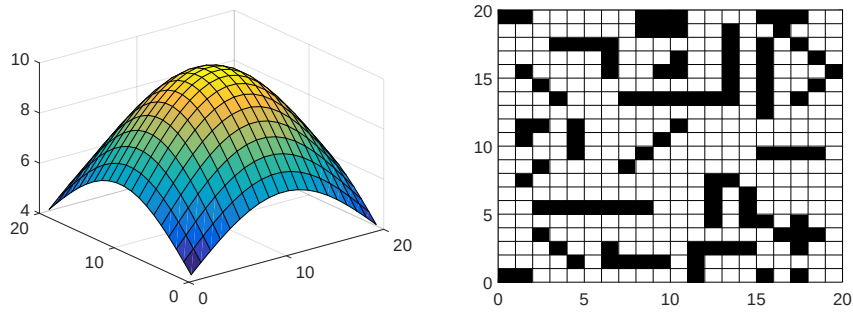
This chapter is based on three published papers entitled “Neural networks learning: Some preliminary results on heuristic methods and applications”, “Learning multiple conflicting tasks with artificial evolution” and “Does training lead to the formation

of modules in threshold networks?". The first section deals with the description of the studied application. Our results are then presented in the two following sections corresponding to the two different networks models considered throughout this work.

4.1 Problem Description

The abstract application we worked on is based on an experimental framework introduced by Beaumont (1993) and can be seen as a toy example of evolutionary robotics. Indeed, it is extremely simple and some hypotheses made such as the grid-world and the perfect sensing and actuation are quite unbearable. Nevertheless, this problem is appropriate in the context of our research. Indeed, we are interested in general questions about the features of evolved networks, regardless the considered tasks. Consequently, this application enables us to address these questions without spending too much time on the experimental settings which can be quite complex in more realistic robotics problems.

In this application, virtual robots are trained to achieve two different tasks in a virtual arena. This arena is a discretized grid of size 20×20 with a two dimensional torus topology on which robots are allowed to move into any neighboring cells at each displacement. Assuming the arena possesses one global maximum, the aim of the first task, named $T1$ in the following, is to reach and to stay on this global maximum, represented with a color code in the left panel of Figure 4.1. The second task, $T2$, consists in moving incessantly by avoiding zones where robots lose energy, called "dangerous zones" and represented by black boxes on the right panel of Figure 4.1.



(a) Area to climb

(b) Area with "Dangerous" zones

Figure 4.1: Arenas where robots are trained

The robots we considered have 17 sensors and 4 motors. Nine sensors check the local slope, that is the heights of the cell on which a robot is located and the cells around it. The eight others are used to detect the presence of "dangerous zones" among cells surrounding robots. The motors control the movements of robots, i.e. moving to the north-south-east and west and their combinations.

ANN used as robot controllers are made of 43 nodes: 17 inputs, 4 outputs and 22 hidden nodes. This number is large enough to let sufficient possibilities of connections to achieve the tasks, but it is still low to use reasonable level of CPU resources. The topology of these networks is completely unconstrained, except for self-loops that are prohibited as explained in Chapter 3. The states of neurons are binary. They can be activated (equal to 1) or inhibited (equal to 0).

Consequently, information acquired by sensors is turned into binary values before being transmitted to the robot controller. The states x_i of the nine first inputs are thus calculated according to the formula

$$x_i = \left\lfloor \frac{1.9(h_i - h_i^{min})}{h_i^{max} - h_i^{min}} \right\rfloor$$

where h_i corresponds to the height observed by sensor i and h_i^{min} and h_i^{max} respectively represent the minimal and maximal heights among the nine cells reachable for the robot. The coefficient 1.9 is used to guarantee that each of these states has an integer part equal to 0 or 1. With regard to the eight other inputs, their state is equal to 1 if the sensor detects a "dangerous" zone and 0 otherwise.

After the ANN computation, the binary controller outputs are turned back into motors movements. More precisely, two motors are devoted to the west-east movements. The two others control the north-south direction. All these movements follow the rule detailed in Table 4.1. By convention, the grid is displayed with the north facing upwards.

Output 1	Output 2	Movement
0	0	south, west
0	1	/
1	0	/
1	1	north, east

Table 4.1: Movement performed by robots according to the ANN outputs

The training set of $T1$ is made up of one arena configuration, whose cells heights are assigned following a normal bivariate function, and ten initial positions on this arena. For each of these positions, the robot carries out 20 steps, and we save the mean height of the last five steps. Then, we compute the value fit which is the mean height on all the displacements. The performance value for $T1$ is finally given by

$$f_1 = \frac{fit - h_{min}}{h_{max} - h_{min}}$$

where h_{min} and h_{max} are respectively the minimal and maximal heights of the grid. Consequently, a robot has a performance equal to 1 if, for all initial positions, it reaches the peak at least at the sixteenth step and stays on it until the end of the simulation.

As for $T2$, the training set is composed of three arena configurations with different "dangerous" zones. Five initial positions are identified per arena. For each of these fifteen initial cells, the robot is assigned a starting energy equal to 100 and performs 100 steps in the arena. If it moves to a "dangerous" cell, its energy is decreased by 5. Moreover, if it moves to a cell that it has already visited, its energy is reduced by 1. This last condition is needed to force the robot to move instead of remaining on a "safe" cell. If the energy becomes negative, we set its value to zero. We save the final energy for each initial position and we compute the mean final energy. The performance of the robot is then computed as

$$f_2 = \frac{\text{final energy}}{\text{maximum energy}}$$

Therefore, a robot has a performance equal to 1 if it moves continuously on the arena without visiting "dangerous" zones and already visited cells.

Let us remark that this kind of performance measures is quite unusual in evolutionary robotics. Indeed, the robot generally receives an indication directly after failing while in our case, it only receives its performance value after its whole walk and it does not know exactly at which step it fails. In a first time, we tried to stop the robot as soon as it walks into a dangerous zone. But this version made the computation of both performance values together harder and longer. Indeed, T_1 can not be correctly assessed if this stop takes place after a small number of steps and has to be evaluated separately. Therefore, we have chosen our uncommon measures to make the dual assessment easier.

When both tasks are trained simultaneously, the training set and the performance functions are slightly adjusted. The training set is composed of the three configurations defined for $T2$ and their initial positions. The robot is assigned a starting energy equal to 20 instead of 100 and performs 20 steps in the arena. The rule for the energy loss is unchanged. as well as the way of computing the mean height on the last five steps for each initial position if the final energy is different from zero. Otherwise, this mean height is set to zero. Similarly, if the value fit is lower than h_{min}, the performance value of the robot is zero. These modifications in performance functions are introduced to prevent robots from having a very good behavior for one task and a very bad one for the other. Let us note that these two tasks are competing each other. Indeed, in the first task, we hope that the robot stops on the peak and in the second one, we ask it to move continuously. So, we look for a tradeoff between the two tasks in the robot behavior. Parameters of training sets and performance functions have been set to the present values after a careful analysis of some generic simulations and by a test and error procedure.

4.2 Discrete Weights Model

The first model used in this work is a discrete model with weight values equal to 1 if the links are excitatory or -1 if they are inhibitory. Given the present state for each

neuron (x_i^t), the updated state is given by the following rule:

$$\forall j \in \{1,\ldots,N\} \quad : \quad x_j^{t+1} = \begin{cases} 1 & \text{if } \sum_{i=1}^{k_j^{in}} w_{ji}^t x_i^t \geq \theta k_j^{in} \\ 0 & \text{otherwise} \end{cases}$$

where k_j^{in} is the number of incoming links in the j–th neuron under scrutiny, θ the threshold of the neuron, a parameter lying in the interval $[-1,1]$, and w_{ji}^t is the weight, at time t, of the synapse linking i to j.

The GA associated with this model is then discrete-valued for the weights and real-valued for the thresholds. The operators used for the steps of selection, crossover, mutation and replacement have been described in detail at the end of Chapter 3. The GA settings are summed up in Table 4.2. As a reminder, each experiment is independently replicated ten times and results are reported in terms of means.

Features	
Population size	100
Number of generations	10000
Crossover rate	0.9
Mutation rate	0.01

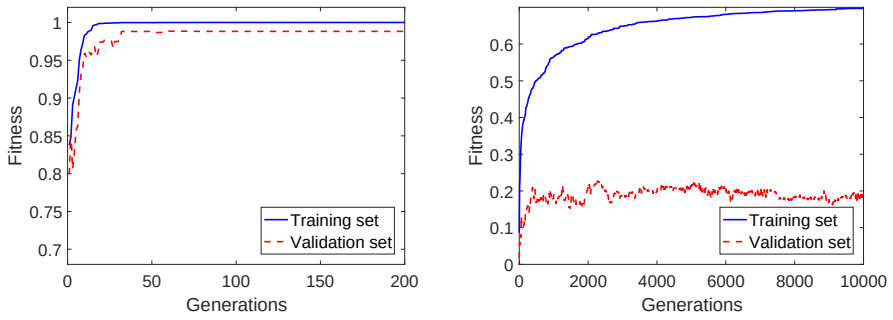
Table 4.2: Summary of the GA settings

We begin by presenting some preliminary results on the learning of each task and a comparison between different GA. We then focused on results concerning our two main questions, namely the study about the best learning sequence and the emergence of modularity.

4.2.1 First Results

First, we studied the learning of each task separately. For this, we analyzed the fitness evolution according to the number of generations on the training set and on validation sets. The latter ones are obtained by modifying the slope of the surface and the position of the peak for $T1$ and by moving zones where robots lose energy for $T2$. Figure 4.2 represents the mean maximum performance for $T1$ on the left panel and for $T2$ on the right panel. We can remark that the second task is the most difficult to achieve. Indeed, we observe that the optimization of $T1$ is perfectly performed in less than 200 generations while $T2$ does not reach its maximum performance at the end of the optimization i.e. after 10000 generations. We can also see that the generalization is easier for $T1$ and that our second task seems to be overfitted. To avoid this problem, we could consider to use a regularization technique (see e.g. Shalev-Shwartz and Ben-David, 2014). As a reminder, the principle is to penalize the performance function to prevent the evolved network from matching too much to the training set. Nevertheless, the aim of this research is not to obtain the best possible networks for achieving the

tasks but to observe the features of evolved networks. Moreover, we can state that the evolved networks obtained so far are better than random networks. Indeed, the mean performance of these networks on validation sets is about 0.2 while the mean performance of the initial population, made of random networks, is inferior to 10^{-2}. Consequently, we decided not to change our experimental settings and to go further in our analyses.



(a) $T1$ (b) $T2$

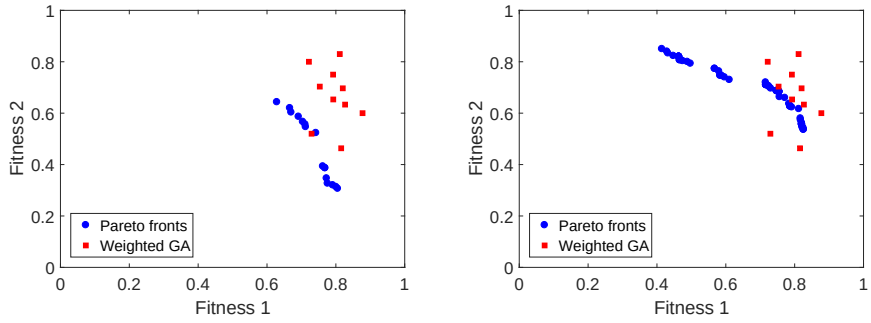
Figure 4.2: Evolution of robot's performance

Then, we used the two kinds of multiobjective genetic algorithms described in Chapter 2 to train both tasks together. The first one is a simple GA whose fitness is the weighted sum of the tasks performance measures. The second one is based on the Pareto optimality theory. The performance values obtained on the training set with these two GA are shown in Figure 4.3. For the weighted GA, we represent each best individual of the ten simulations while for the other one, we draw the best Pareto front among the ten optimizations. Let us remind that the fitness of $T2$ is slightly different from the one used when the task is trained alone, as the robot has to perform 20 steps instead of 100.

In a first time, we used the method of Srinivas and Deb (1994) described in Chapter 2 for the fitness allocation of non-dominated individuals. We can see on the left panel that, in this case, we get better fitness with the weighted GA. Consequently, we tested another allocation method in which we assigned a fitness of 1 to individuals of rank 1 and we decreased the assigned fitness by 20% for each subsequent rank. With this simpler method, we acquired networks with performance measures closer than those got with the weighted GA, as shown on the right panel of Figure 4.3. Moreover, the Pareto fronts appear more spread out, which means that we explore more possibilities during the optimization.

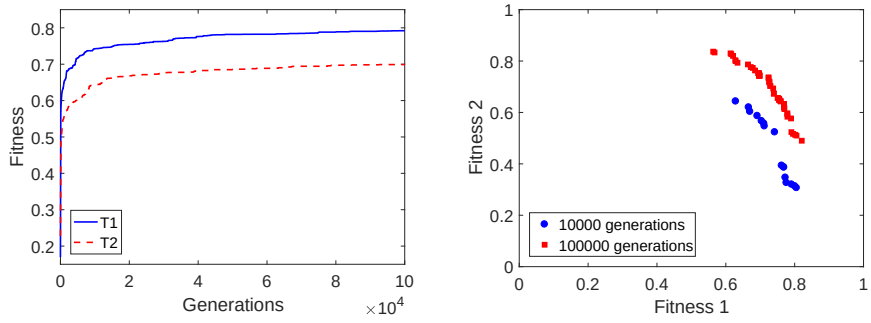
As the GA based on Pareto optimality explores the space of non-dominated individuals, it could require more generations to reach a good level of quality performance. So, we checked this possibility by fixing the number of generations to 100000 and performing the optimization process again. The mean maximum performance values of both tasks at each generation are presented on the left panel of Figure 4.4. We observe

that the biggest part of the learning occurs during the first 10000 generations, even if the Pareto fronts got after 10000 and 100000 generations represented on the right panel are quite different. Let us remark that the CPU time of these simulations are quite long (about 10 days for 100000 generations on a standard PC) due to the sorting and sharing process. We thus have to find a tradeoff between running time and acceptable performance.



(a) Srinivas and Deb'allocation (b) Second allocation

Figure 4.3: Comparison of performance obtained with the weighted GA and the one based on Pareto optimality for different fitness allocation methods



(a) Performance measures (b) Comparison of Pareto fronts

Figure 4.4: Results for the GA based on Pareto optimality with a longer optimization

Although neural networks so far obtained led to the expected robot behaviors, they exhibited many "useless" connections, i.e. links that bring any relevant information and that can be removed. An example of such a link is given by the upper hidden node of Figure 4.5, which does not receive any information from other nodes. Consequently, we envisaged different methods to decrease this number. Firstly, we introduced a penalty on the number of connections as a third objective function in our

genetic algorithms. This function is calculated as

$$f_3 = 1 - \frac{\#\text{ of links}}{\max \#\text{ of links}}$$

For the weighted GA, the choice of the weights assigned to each objective was not trivial. We concluded from preliminary analyses that a good combination is obtained by setting the weights of the two tasks performance to 0.35 and the penalty on the number of connections to 0.3. Nevertheless, these preliminary analyses were not exhaustive and better combinations might exist. Let us note that, to cope with this problem, we considered to implement a meta genetic algorithm to find the optimal weights combination. We achieved this implementation but we never used it because of its demanding running time.

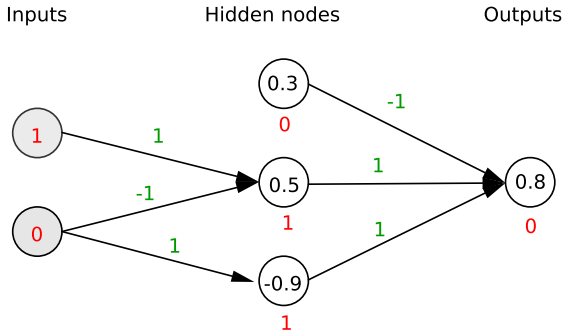
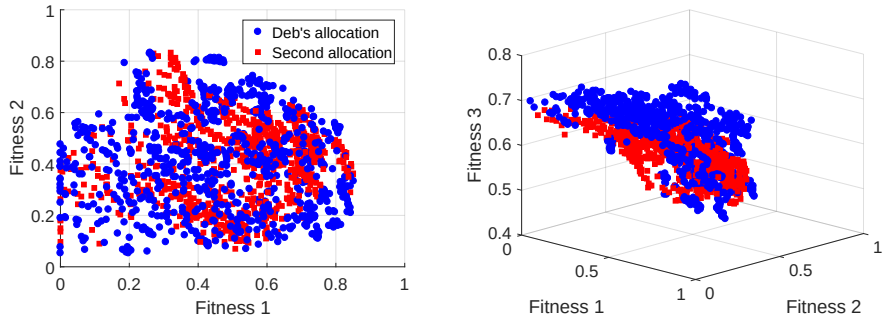


Figure 4.5: Network with a "useless" connection, the node in the middle layer with threshold 0.3 and no entering links

Preliminary results were inconclusive. Indeed, the reduction in the number of connections obtained with the weighted GA was low and slow, and final networks always exhibited many "useless" links. Moreover, for the GA based on Pareto optimality, adding a third objective function prevented best networks from reaching tasks performance they had in the case of the optimization with two objectives. Another possibility to use the penalty was first to train the networks on the tasks, and then to optimize the number of connections by fixing some minimal levels for the two fitness. This method and the previous one brought very similar results.

We also compared our methods of fitness allocation in the case with three objectives. This comparison is displayed in Figure 4.6. On the left panel, we focus on the performance measures of both tasks and we observe a projection on the plane $T1-T2$. The right panel presents a 3D view of the results. We can observe that, contrary to the case with two objectives, the assignment of Srinivas and Deb seems to be more appropriate. Our hypothesis is that, with the second allocation method, the considered solutions are sparse in the space of possibilities. Indeed, as the dimension of the objective space increases, there are more possibilities to be a non-dominated individual and almost all individuals have a fitness equal to 1. On the contrary, the Srinivas

and Deb's assignment takes into account the distance between solutions and favors isolated individuals, which can lead to a bigger diversity among solutions.



(a) Projection on the plane $T1 - T2$

(b) 3D view

Figure 4.6: Comparison between the two allocation methods for the optimization with three objectives

Our second attempt to reduce the number of connections was to replicate our evolved networks by removing "useless" links at random. The principle of this method is very simple. We begin by choosing one link randomly. Then, we remove it and we check if the fitness has decreased. In this case, the link is reinserted in the networks. Otherwise, it is definitely removed. This process is repeated until we get a sequence of 50 randomly chosen links that can not be removed. Next, we check every remaining connections and we remove one among those that do not alter the fitness. We iterate this step until it is no longer possible to keep the same fitness. This method returns limited results. Indeed, it is a stochastic method and, consequently, the final networks strongly depend on the order in which we remove the connections. Moreover, the removal of "useless" links sometimes reduces the fitness. Indeed, the updated state of each neuron is given by a comparison between the sum of stimuli divided by the number of incoming links and the threshold. By removing links, we modify the divisor and we influence this comparison. This limitation is illustrated in Figure 4.7 in which we reproduce the network of Figure 4.5 by detaching the "useless" hidden node. We can observe that the output state has changed because the sum of stimuli is divided by two instead of three.

In order to overcome this limitation, we modified the corresponding threshold proportionally to the importance of the removed connections in the sum of stimuli. Assuming that the sum of stimuli is represented by s, the new threshold is calculated as

$$\begin{aligned} \frac{s}{k_j^{in}} &> \theta \\ s &> k_j^{in}\theta \end{aligned}$$

$$\frac{s}{k_j^{in}-1} > \frac{k_j^{in}\theta}{k_j^{in}-1} = \theta_{new}$$

This change led to networks with lower numbers of connections. We also considered to introduce this approach in GA. We obtained results comparable to those got with previous GA but we noticed that this modification results in a significant decrease (30%) in the number of improvements observed during the optimization process.

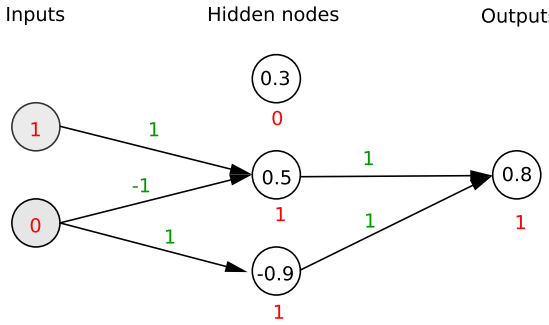


Figure 4.7: Network without the presumed "useless" connection

Once the "useless" links were removed, we tried to improve the fitness of our networks by implementing a GA such that networks keep the same number of connections along the optimization process. The initial population is made of the replicas of a network got after removing "useless" links and mutations of these replicas. The allowed mutations are permutations of links and changes of thresholds. We tested two different GA, namely a GA without crossover and a GA with crossover, with the false preconceived idea that crossover could increase the number of links in the networks. The maximum fitness improvement got with these GA is 0.07 which represents a relative increase inferior to 5%. Consequently, we gave up this attempt.

Then, we tried to obtain results less sensitive to the randomness by developing an algorithm of simulated annealing, as described in Chapter 3. The probability of removing one link is given by

$$P_{\text{remove}} = \begin{cases} 1 & \text{if } fit_{new} \geq fit_{old} \\ \exp(\frac{fit_{new}-fit_{old}}{T}) & \text{otherwise} \end{cases}$$

where T is the temperature and where fit_{old} and fit_{new} are respectively the fitness got by the network with and without the chosen link. We first performed an analysis of the parameters to find appropriate initial temperature (0.1), cooling parameter (0.95) and number of iterations (10000). Then, we compared the results with those got with our random search algorithm. Table 4.3 presents this comparison for our final networks. We observe that the simulated annealing algorithm is more efficient to reduce the number of links. However, the standard deviation and the performance are less

satisfying than for the random search. The use of this algorithm does not solve our problem of variations between final networks as the randomness is always present in the choice of the links to remove.

	Random search	Simulated annealing
Fitness	0.7283	0.6844
Mean number of links	478	421
Std number of links	14	24

Table 4.3: Comparison of methods used to decrease the number of "useless link"

This first phase of analysis have shown that we obtained satisfying results with the two multiobjective genetic algorithms. Nevertheless, we decided to select and to work with the weighted GA for the rest. Indeed, the GA based on Pareto optimality requires more running time due to the sorting and sharing of non-dominated individuals. We have also considered different ways to decrease the number of links in evolved networks. This attempt led to the conclusion that all these ways produce similar and strongly stochastic results.

4.2.2 Best Learning Sequence

Our starting point for this section is the research of Calabretta et al. (2008) in which they address the question as to whether a non-modular neural network can be trained to achieve the same performance as a modular one in accomplishing multiple tasks. In their work, the neural network is trained to perform two simultaneous classification tasks on the same input, one task being harder than the other. The conclusion of this experiment is that the best performance is attained by training first on the hardest task and successively also on the second. This learning scheme is the one enabling the search process to minimize the neural interference, which appears every time neural weights involved in the computation of both tasks are varied considerably, leading to a detriment of the overall network performance. If the hardest task is learned first, then neural interference is reduced as much as possible because the magnitude of weights variations is kept at minimal levels.

In this work, we flesh out the spectrum of experiments on which the results are valid. Indeed, our context of study is very different from the one developed in Calabretta et al. (2008). Firstly, it consists of two dynamical tasks, which are computationally more complex than classification ones. Secondly, both network structure and weights can undergo changes during the learning phase while the structure was fixed in the research of Calabretta. Thirdly, the tasks have separate inputs but common outputs, which is the opposite of Calabretta's experiments. Finally, let us remind that our tasks are conflicting.

Three different learning processes are considered in order to find the best way to train robots for multiple tasks. In the first case, networks learn both tasks simultaneously (named *A* in the following). The fitness is then obtained by averaging the

performance measure of each task using equal weights. In the second one, robots are first trained to achieve the first task and then both tasks simultaneously (named B, $T1$ first) while for the last sequence, it is the second task that is first trained (B, $T2$ first). For these two schemes, the GA is first applied with one fitness function, the one related to the task under scrutiny, and then once again with the weighted sum of both performance measures.

Let us observe that a fourth possible configuration (called *juxtaposition* in the following) could be used. That involves to consider two smaller networks, each having inputs only for a given task and trained for it, and to combine them by adding a small extra network. More precisely, one network composed of 9 inputs, 5 hidden neurons and 4 outputs has only the inputs able to measure the local heights and is trained for $T1$. A second one, formed by 8 inputs, 5 hidden neurons and 4 outputs, is trained to detect "dangerous" zones. As for the extra network, it is made of 8 inputs, corresponding to the outputs of the two small networks, and 4 final outputs, which control robot movements. Only this extra network is then optimized for both tasks with the weighted sum of the performance measures. We do not consider this fourth configuration for the comparison. Indeed, this learning configuration resulted in robots with poor tasks performance. Moreover, we are mainly interested in non-modular networks in this part of the research.

We compared our three different learning schemes. Table 4.4 presents, for each of them, the mean and standard deviation of the final performance on the training set. Given the small number of data available, we used the non-parametric hypothesis test of Kruskal-Wallis (see e.g. Hollander et al., 2013) to evaluate the similarity of samples distributions by comparing their medians, also presented in Table 4.4. As the p-value is equal to 0.1232, the null hypothesis is not rejected at a confidence level of 95%. We can thus reasonably conclude that the medians of the three learning configurations are equal on the training set.

Learning scheme	Mean	Median	Std
A	0.7295	0.7344	0.0588
B, $T1$ first	0.7252	0.7336	0.0407
B, $T2$ first	0.7606	0.7721	0.0606

Table 4.4: Mean and standard deviation of the final maximum performance on the training set

In order to properly assess the performance of controllers obtained at the end of the training process, we then compared their behavior using test sets. For this, we considered different ways for changing the arena. In the first and second test sets, we modified the zones where robots lose energy and the shape of the surface to climb (position of the peak and slope of the surface) respectively. In the last set, we performed both modifications. The mean and standard deviation of the performance got by neural networks trained with the different learning schemes are presented in Table 4.5 for each of these sets. Boxplots of the results are presented on the right panel

of Figure 4.8. As a reminder, the central mark of each box is the median of the data, its edges are the 25th and 75th percentiles, the whiskers extend to the most extreme datapoints considered to be not outliers, and the outliers are plotted individually. Let us remark that, as medians are very close from means and observable in boxplots, they are no longer presented in tables. The third learning sequence, i.e. the training of the second task followed by the one of both tasks (B, $T2$ first), seems to be the best way to train robots for multiple tasks. Indeed, the mean performance of this scheme is the highest for each test set. But we note that the standard deviation is also the highest for this learning sequence.

Arena		Random nets	A	B, $T1$ first	B, $T2$ first
1	Mean	0.0047	0.1073	0.1520	0.2718
	Std	0.0036	0.0707	0.1073	0.1290
2	Mean	0.0120	0.2946	0.3379	0.4318
	Std	0.0060	0.1259	0.1037	0.1593
3	Mean	0.0053	0.1746	0.1687	0.3313
	Std	0.0076	0.1233	0.0933	0.1390

Table 4.5: Mean and standard deviation of the final maximum performance on test sets

To confirm our intuition, we made use of statistical tests whose results are presented on the left panel of Figure 4.8.

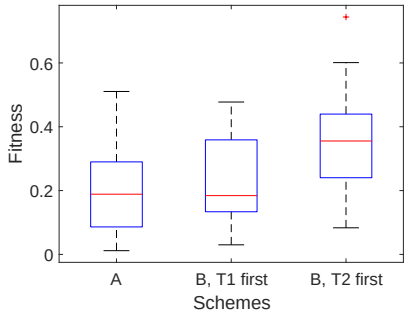
Kruskal-Wallis

p-value	
$3.8281.10^{-4}$	< 0.05

Wilcoxon

Seq. 1	Seq. 2	p-value	
A	B, $T1$ first	0.4161	> 0.05
A	B, $T2$ first	0.0002	< 0.05
B, $T1$ first	B, $T2$ first	0.0033	< 0.05

(a) Hypothesis tests



(b) Fitness boxplots

Figure 4.8: Comparison of final performance obtained with different learning schemes on test sets

The first test that we used is Kruskal-Wallis in order to check the equality of the three medians. As the p-value is smaller than 0.05, we rejected the null hypothesis with a probability of taking a wrong decision equal to $3.8281.10^{-4}$. We thus came

to the conclusion that medians are significantly different. Then, we used the non-parametric Wilcoxon test (see e.g. Hollander et al., 2013) to evaluate medians by pair. With this test, we observed that the median of the scheme B, $T2$ first is significantly different from the two other medians. In conclusion, it seems better to train the hardest task before training both tasks simultaneously.

Tables 4.6 and 4.7 present the performance associated with $T1$ and $T2$ respectively. We can see that the learning scheme B, $T2$ first makes it possible to attain the best results because it helps the training process to improve the fitness function of both tasks.

Arena		A	B, $T1$ first	B, $T2$ first
1	Mean	0.0520	0.1296	0.2672
	Std	0.0879	0.1441	0.1995
2	Mean	0.3322	0.4305	0.4625
	Std	0.1667	0.1736	0.1872
3	Mean	0.2063	0.1910	0.3937
	Std	0.2085	0.1401	0.2240

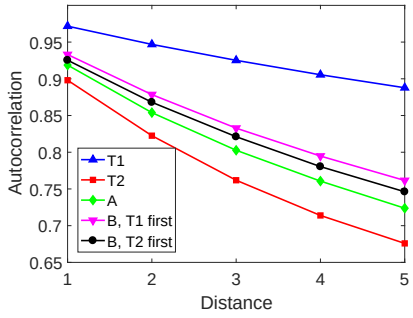
Table 4.6: Mean and standard deviation of $T1$ performance on test sets

Arena		A	B, $T1$ first	B, $T2$ first
1	Mean	0.1627	0.1743	0.2763
	Std	0.0770	0.0771	0.0748
2	Mean	0.2570	0.2453	0.4010
	Std	0.1102	0.0594	0.1554
3	Mean	0.1430	0.1463	0.2630
	Std	0.0589	0.0590	0.0646

Table 4.7: Mean and standard deviation of $T2$ fitness on test sets

Our conclusion coincides with the one drawn by Calabretta et al. in their paper. Nevertheless, since the networks and the learning process are different, the explanation lying on the neural interference is no longer valid and a different argument has to be found. To this aim, we studied the autocorrelation of the performance landscape described in Chapter 2 for each learning sequence. For this, we performed 10 random walks of 10000 steps for each scheme. These walks are executed by networks from the initial population of the GA, i.e. by random networks for the separate and simultaneous learning of tasks ($T1$, $T2$ and A) and by networks respectively trained for $T1$ and $T2$ in the case of successive learning. Figure 4.9 presents the mean of the autocorrelation on the 10 walks for each distance from 1 to 5 between neighboring candidates and for each learning sequence on the left panel. The table on the right displays the mean and standard deviation for autocorrelation of distance 1.

The autocorrelation values could explain the marked differences in the final performance of robots in the extreme cases. Indeed, $T1$ is the easiest task, with the highest autocorrelation values corresponding to a flatter fitness landscape and $T2$ is the hardest, with the lowest values and thus a more rough fitness landscape. The final high performance attained by the learning sequence B, $T2$ first does not seem to find support in the autocorrelation values, which are quite close for the three learning schemes. However, we must observe that the measure is taken on a landscape corresponding to the final phase of evolution on $T1$ (B, $T1$ first) and $T2$ (B, $T2$ first), respectively. Therefore, at the beginning of the search with the composite fitness (accounting for both the tasks), the autocorrelation of the landscape is strongly influenced by the initial training phase with only one active task. Hence, the landscape autocorrelation in scheme B, $T2$ first heavily depends on the autocorrelation of the landscape for learning $T2$, which is the very low. We can conclude that in B, $T2$ first, the addition of the fitness component corresponding to $T1$ is considerably beneficial to the search process, enabling the genetic algorithm to explore a way smoother landscape, whilst the addition of $T2$ fitness has a dramatically detrimental effect, harming search effectiveness.



(a) Distances from 1 to 5

Learning sequence	Mean	Std
$T1$	0.9717	0.0061
$T2$	0.8981	0.0192
A	0.9187	0.0122
B, $T1$ first	0.9331	0.0181
B, $T2$ first	0.9253	0.0184

(b) Direct neighbors

Figure 4.9: Autocorrelation of the performance landscape

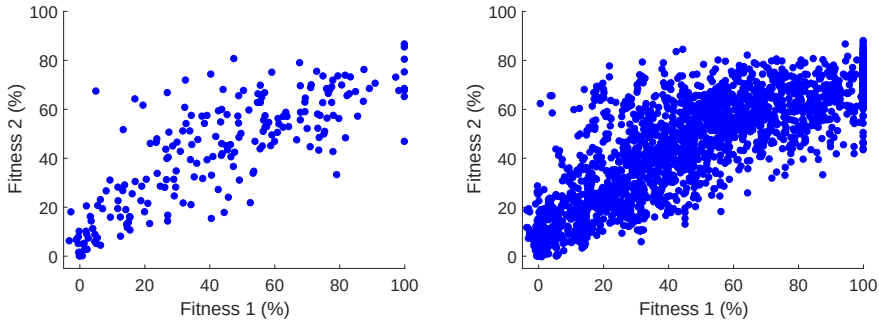
We also investigated the existence of correlation between the tasks. With this aim, we performed the removal of one hidden neuron at once and we analyzed the modifications of the performance measures. These modifications are shown on the left panel of Figure 4.10. We observe a linear relation between the modifications, which is confirmed by the correlation coefficient (0.8138). However, can we conclude that neurons important for $T1$ are also important for $T2$? Or, is this correlation due to the important number of links in the networks, that we modify too strongly by removing one neuron? To obtain a first answer to our questions, we performed the same analysis on networks obtained after removing "useless" links (right panel of Figure 4.10). The correlation is always observable, with a correlation coefficient equal to 0.8099.

To sum up, our two tasks seem to be correlated. To confirm this statement, we observed robot behaviors and mean performance measures obtained by networks trained

with the best learning sequence in particular situations. Table 4.8 presents the fitness if the surface is flat. Let us observe that the fitness of $T1$ can no longer be considered as it is computed by

$$f_1 = \frac{fit - h_{min}}{h_{max} - h_{min}}$$

with h_{min} and h_{max} having the same value. We then concentrate on the fitness of $T2$. We can see that the slope seems to have a big influence on the second task. Indeed, the performance measure is decreased by half. So, we can assume that, when both tasks are trained together, $T1$ helps to learn $T2$ by giving a movement to the robot. Thus, it is not only the fact of being penalized when it stays on place which is important but the fact that the robot has a slope to follow.



(a) Networks trained with scheme *A*

(b) Networks without "useless" links

Figure 4.10: Correlation between the tasks

	Fitness of $T2$
inputs = 0	0.3790
inputs = 1	0.3320

Table 4.8: Mean performance measures of $T2$ reached by networks trained with the best learning scheme in an arena without slope

Table 4.9 shows the performance measures if there are not any "dangerous" zones. In this case, the fitness of $T2$ keeps sense because the robot is always penalized if it stays on the same cell. The results obtained for the first fitness are explained by the fact that the robot reaches the area around the peak but goes on moving to avoid a loss of energy.

	Fitness
$T1$	0.6556
$T2$	0.8907

Table 4.9: Mean performance measures reached with the best learning scheme in an arena without "dangerous zones"

4.2.3 Study of Modularity

To study the topological modularity, we used an implementation of the Louvain method which is adapted to oriented graphs with positive weights. So, the analysis was made on connectivity matrices whose weights are absolute values of initial connections. The mean modularity observed among our final networks is about 0.06, which means that the method does not detect the presence of modules in the networks. We tried to use a non-oriented version of the Louvain method by turning our connectivity matrices into symmetric matrices but results were always very low. Regarding functional modularity, it was analyzed with the DCI on collected multidimensional time series composed of network's variables values during the execution of the task. No clear modules were found in all analyzed cases. Results were similar when we added the penalty on the number of connections and when we considered networks after removing "useless" links, whatever the method used for this removal.

At this stage of our research, we were convinced, supported by the results available in the literature, that we should find modularity in our evolved networks. So, the results obtained were unexpected and disappointing. After some discussions with Franco Bagnoli, from the University of Florence, we came to the conclusion that the absence of modularity could be due to our model. Indeed, the basic statement is that modularity emerges if the evolutionary modifications appear gradually. And so, our binary connections could bring too important modifications. Consequently, we decided to modify our model by considering real weights instead of discrete weights.

4.3 Real Weights Model

The connections weights of the new model are real values lying between -1 and 1. The updates of neurons are then performed following the rule

$$\forall j \in \{1,\dots,N\} \quad : \quad x_j^{t+1} = \begin{cases} 1 & \text{if } \sum_{i=1}^{k_j^{in}} w_{ji}^t x_i^t - \theta \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

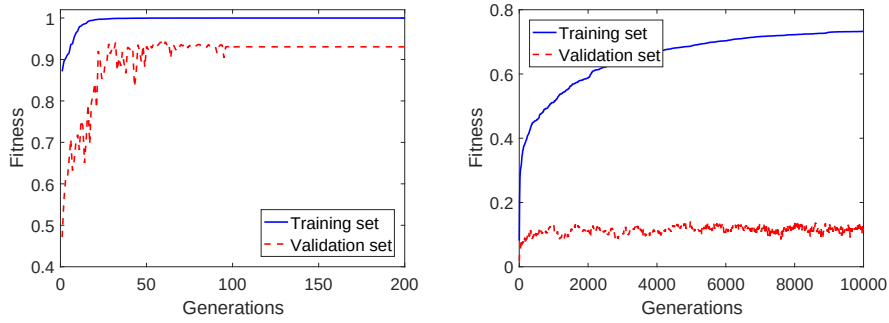
where, as a remainder, k_j^{in} is the number of incoming links in the j–th neuron under scrutiny, θ the threshold of the neuron, a parameter lying in the interval $[-1,1]$, and w_{ji}^t is the weight, at time t, of the synapse linking i to j.

The GA associated with this model is real-valued. The steps of selection, crossover, mutation and replacement are the same as for the previous model. The GA settings are also similar (see Table 4.2), except for the mutation rate which is fixed to 0.005 as the evolutionary changes have to be gradual. Once more, each experiment is independently replicated ten times.

We begin this section by a comparison between our two models. It gives us the opportunity to check if results obtained for the first model are always valid. Then, we present results on the study of modularity.

4.3.1 Comparison with the Discrete Weights Model

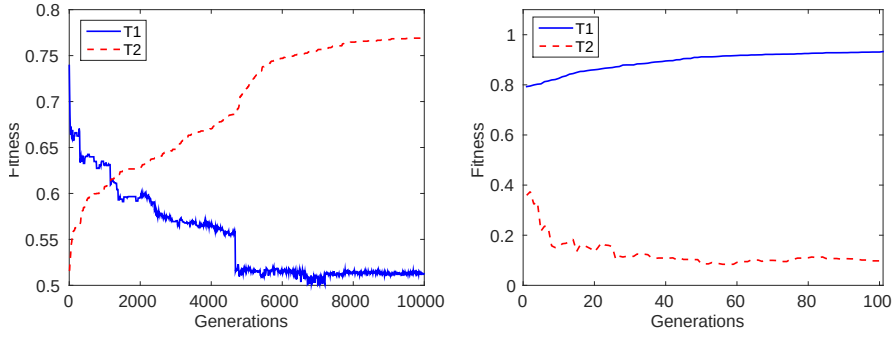
First of all, we observed the fitness evolution of single tasks on the training set and on our validation sets. This evolution is presented on the left panel of Figure 4.11 for $T1$ and on the right panel for $T2$. We can see that $T2$ is overfitted and that its generalization abilities are even worst than for our first model. Indeed, the mean final fitness on validation sets is decreased by half. Nevertheless, results of our evolved networks remains different from those of random networks and are then interesting for our analyses.



(a) $T1$ (b) $T2$

Figure 4.11: Evolution of robot's performance

We also observed the time needed to forget one task. Our initial networks were those optimized with the scheme A. These networks are then trained on $T2$ to observe the forgetting of $T1$ and, in return, trained on $T1$ to observe the forgetting of $T2$. The evolution of fitness associated with $T1$ and $T2$ according to the number of generations is shown in Figure 4.12. We observe that the forgetting of $T2$ is nearly immediate.



(a) $T1$ (b) $T2$

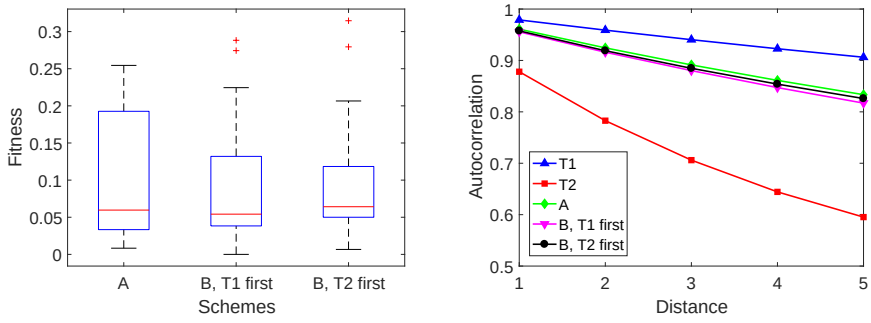
Figure 4.12: Tasks forgetting

Then, we replicated the previous study on the best learning sequence to check if our results were valid whatever the model. Table 4.10 presents the mean performance got by each learning scheme on the training set and on three test sets. As for the model with discrete connections, the medians of the three learning sequences seem at first sight to be similar on the training set. And, contrary to results obtained previously, the medians are also visually similar on the test sets (see boxplots on the left panel of Figure 4.13). This observation is confident with the p-value obtained for the Kruskal-Wallis test which is equal to 0.8784, leading to fail to reject the null hypothesis.

Area	A	B, $T1$ first	B, $T2$ first
Training set	0.7393	0.7544	0.7675
Test set 1	0.0487	0.0595	0.0577
Test set 2	0.1721	0.1668	0.1553
Test set 3	0.0809	0.0628	0.0698

Table 4.10: Mean performance on the training set and on three test sets.

With regard to the autocorrelation of the landscape, we observe on the right panel of Figure 4.13 that $T2$ has always the roughest fitness landscape.



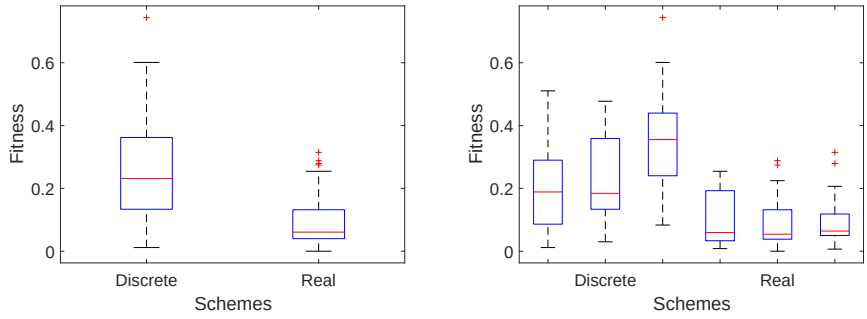
(a) Fitness boxplots

(b) Autocorrelation

Figure 4.13: Comparison of the performance obtained with different learning schemes

By comparing values obtained in Table 4.5 and 4.10, we noticed that the ANN model seems more robust with discrete connections than with real connections. This result can also be observed in the boxplots of Figure 4.14 that present the performance measures obtained for the two models. On the left panel, the different learning schemes are separated while they are grouped together on the right. Our observation is confirmed by the Kruskal-Wallis test with a p-value smaller than 10^{-12} which leads to reject the null hypothesis. These results could be linked to bad generalization abilities of this second model for $T2$. In conclusion, we can see that results are sensitive to the

model. That could also be the case for results on the emergence of modularity.



(a) By learning scheme

(b) By model

Figure 4.14: Comparison of the performance values obtained with the two ANN models

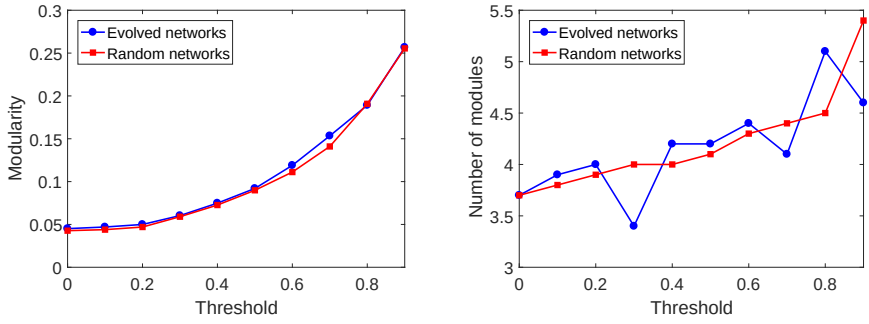
4.3.2 Study of Modularity Emergence

Our starting assumption was that only two conditions are needed for the emergence of modularity. Firstly, at least two tasks must be learned. Secondly, the learning must be incremental, i.e. the modifications in both topology, weights and activation thresholds must be gradual and the structure of the networks can not be too strongly modified in one step.

To study the topological modularity in case *A* and *B*, we analyzed the modularity and the number of modules when we keep large connections, i.e connections whose absolute values of the weights are superior to a fixed threshold value. Figure 4.15 presents a comparison of these results between our evolved networks in case *A* and random networks (used as null hypothesis) with the same features (number of nodes, number of connections and distribution of weights) by considering different threshold values. We can observe that evolved and random networks behave similarly and hence we conclude that the evolved networks are not modular. We obtained analogous results in case *B*.

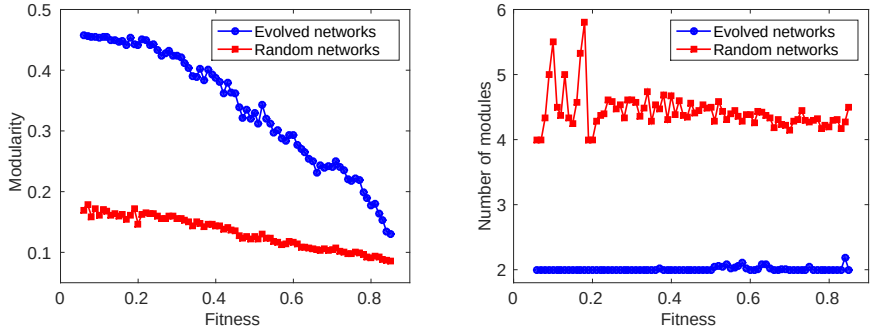
The case *juxtaposition* is analyzed in a different way. Indeed, initial networks are modular as they consist in the combination of two smaller networks, each trained to accomplish one task, and a small extra network. As we saw previously, it is not sufficient to train the extra network to reach good performance measures. So, we enlarged the training to the whole network, which means that connections could be created between the two small networks, between small networks and the extra network, and so on. Thus, we decided to observe the evolution of topological modularity as a function of the robot performance increase. The evolution of trained networks is presented in Figure 4.16 for the modularity (left panel) and for the number of modules (right panel). Results are also compared with those of random networks with the same features. We can see that results obtained for evolved and random networks

are significantly different. We also observe that, in 9 simulations out of 10, the number of modules does not change during the optimization. Each module is made by one initial small network (able to perform a given task), while the nodes of the extra small network are shared between the two main modules. The size of these modules is sometimes slightly modified when one node of the extra network jumps from one module to the other. Although the number of modules is almost constant, we observe a strong decrease of modularity along the learning process.



(a) Modularity (b) Number of modules

Figure 4.15: Topological modularity for the learning scheme *A*



(a) Modularity (b) Number of modules

Figure 4.16: Topological modularity for the learning sequence *juxtaposition*

Following these results, we conclude that none of the training schemes leads to topologically modular networks. In the case *juxtaposition*, it even seems to make disappear the initial modularity, as long as the modularity score is considered. Regarding functional modularity, no clear modules are found in all cases analyzed.

When the robot is trained according to scheme *B*, i.e. sequential learning, naive

modules form in the first phase of training, as only one part of the sensors is stimulated. Nevertheless, these modules disappear when the robot is subsequently trained to accomplish both the tasks. Furthermore, the same results as for the topological analysis are observed in the case *juxtaposition*, showing that the initial modules tend to be blended together in the final training phase. The results returned by the analysis of functional modularity strengthen those on the topology, as they show that not only the networks have no apparent modules, but that they do not even show clusters of nodes which work in coordinated way and corresponding to either of the two tasks.

As the results obtained previously do not support the presence of modules, we consider additional conditions that could be important for their emergence according to the literature. These conditions are the alternation between different goals, the penalty on the number of connections and the decrease in the number of hidden nodes.

Switching between goals

We first followed the suggestion of Lipson et al. (2002) that switching between different goals is important for the emergence of modules. We considered a fourth training scheme D in which the target task is alternated every 100 generations. In preliminary tests, we also considered to alternate every 20 or 50 generations but the simulations with 100 gave us the best robot performance. Even if one particular task is trained in each epoch, all sensors are stimulated. Otherwise, robots can accomplish both tasks but not simultaneously. The results obtained by this fourth training sequence are similar to those of previous schemes. Indeed, the robot performance is also around 0.75 and the analysis of the topological modularity leads us to results similar to those presented in Figure 4.15. The same holds true for functional modularity.

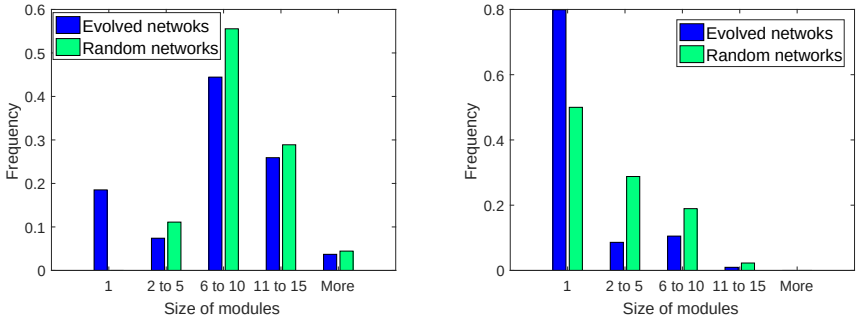
Cost on connections

Bullinaria (2007) as well as Clune et al. (2013) claimed that the penalty on the number of connections is essential for the formation of modules. So, we added this penalty, which is defined as previously by

$$f_3 = 1 - \frac{\#\text{ of links}}{\max \#\text{ of links}},$$

to each of our four training schemes. It contributes to three tenths of the average fitness in each experiment. In case A, the weight of each task is reduced to 0.35 instead of 0.5. For scheme B, robot is trained first on one task with the penalty on the number of connections. Then robot is trained on both tasks with the penalty as described for sequence A. In the case *juxtaposition*, the penalty is only added for the last phase of learning because if it is also used while training the two smaller networks, the resulting fitness is too low (0.35 which is smaller than the half fitness of other experiments). For the case D, the penalty on the number of connections is considered during the training of the two alternated tasks. Let us observe that the fitness described in this paragraph are only used for the training phase. Results are then analyzed using robot performance corresponding to the fitness of both tasks summed using equal weights, in this way we can compare them with the former ones.

When we analyzed topological modularity, we obtained similar results for scheme A, scheme B, $T2$ first and scheme D. Indeed, in these cases, the fitness is nearly the same as without the penalty on the number of connections. Moreover, the modularity is low (~ 0.1) and random networks with the same density of links and the same distribution of weights have comparable values (~ 0.15). The difference appears in the number of modules that is slightly higher in evolved networks as shown in the left panel of Figure 4.17 which shows the distribution of modules according to their size. Indeed, in trained networks, some modules consist of isolated nodes, i.e. nodes without any link with the rest of the network, whereas this never happens in random networks.



(a) Sequence A

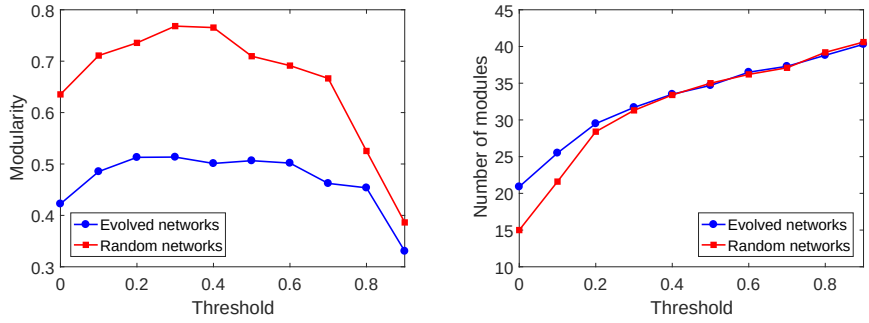
(b) Sequence B, $T1$ first

Figure 4.17: Distribution of modules according to their size

As the cost on the number of connections did not lead to the emergence of modules, we suspected that the penalty that we had fixed was not strong enough to involve modularity. Nevertheless, we got similar results with a larger penalty of 0.5, which is a quite high value representing half of the fitness during the optimization process.

If we consider scheme B, $T1$ first, the value of the fitness decreases slightly with an average value of 0.67. Moreover when we compare the modularity between these networks and random networks with the same features by keeping connections whose absolute values of the weights are larger than a given threshold value (see Figure 4.18), we can observe a significantly different behavior. The value of the topological modularity is lower for evolved networks while their number of modules is higher. If we consider evolved networks without eliminating any connections (threshold of 0), we observe that the mean number of modules is 21, out of which 17 are isolated nodes for evolved networks whereas we got respectively 15 and 7 for random networks. This high frequency of isolated nodes is also clearly apparent in the right panel of Figure 4.17. We can explain such results by the simplicity of $T1$, which requires few connections to be accomplished. When this task is trained alone with the penalty on the number of connections, we get networks with a high level of modularity and a high number of modules, most of which are isolated nodes, comprising non stimulated inputs. As the penalty cost is always active in the second phase of learning, useless

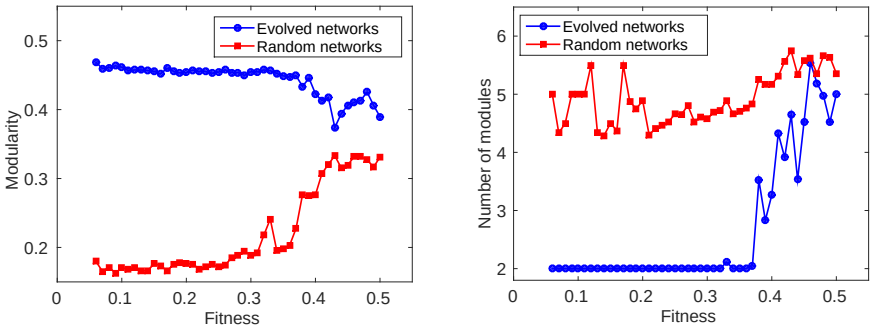
hidden nodes remain isolated, which leads to a higher modularity than for previously analyzed training schemes.



(a) Modularity (b) Number of modules

Figure 4.18: Topological modularity for the learning sequence B, $T1$ first with the penalty on the number of connections

In the case *juxtaposition*, the mean fitness of simulations is 0.44, which is considerably smaller than for other experiments. Figure 4.19 shows the evolution of the modularity according to the increase in robot performance. The cut of modularity seems to be less important than without the penalty but the final fitness is lower. The difference that exists between the modularity of evolved networks and random networks significantly decreases during the learning process.



(a) Modularity (b) Number of modules

Figure 4.19: Topological modularity for the learning sequence *juxtaposition* with the penalty on the number of connections

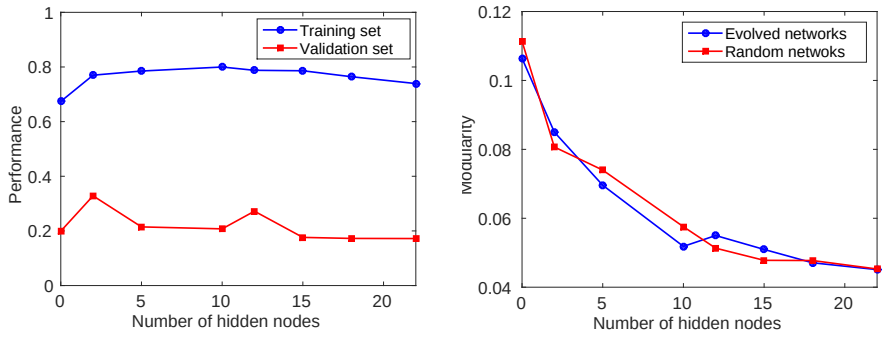
Following this experiment, we conclude that the penalty on the number of links allows to keep a level of modularity close to the initial one. Indeed, the decline in

the number of connections leads to the emergence of isolated nodes that increases the modularity but to the detriment of robot performance. As for functional modularity, significant groups of nodes with coordinated behavior can not be identified.

Number of hidden nodes

Another argument by Bullinaria (2007) is that the number of hidden nodes plays a role in the emergence of modules. Indeed, modularity has more possibilities to appear if the number of hidden nodes is small. Thus, our last experimental settings consisted in decreasing the number of hidden nodes and testing if this can lead to the formation of modules in evolved networks. This case can be considered as a very strong implementation of the previous analysis where, instead of removing one link we remove several links, i.e. all the ones connected to a given node. With this aim, we only considered the first training scheme (A) and networks with 10 and 5 hidden nodes instead of 22. Results are the same as for previous networks with fitness values around 0.79 and modularity values similar to those of Figure 4.15.

Then, we checked the dependence on the number of hidden nodes of robot performance and modularity. One would expect the performance to be very poor for very small number of hidden nodes, i.e. not enough to perform the required computation, then the performance should increase as long as the number of hidden nodes increases up to some number, beyond which no improvement is found. The robot performance is tested on the training set and on the validation sets introduced previously. Results of this analysis are presented in Figure 4.20 and in Figure 4.21 in the case where we added the penalty on the number of connections.



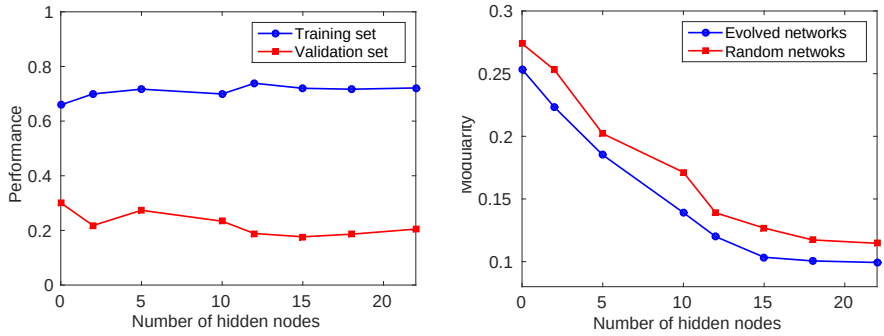
(a) Performance (b) Modularity

Figure 4.20: Importance of hidden nodes

We cannot observe significant decrease of performance when the number of hidden nodes is small. This result might be explained by the fact that memory is not needed to solve the problem and thus hidden nodes, responsible for information storage, are not relevant to accomplish the task. Regarding modularity, it decreases when

the number of hidden nodes increases. Finally, no functional modules have been detected in this case.

In conclusion, also in this experiment no topological modularity is observed. Likewise, the analysis of functional modularity does not support the emergence of modules in these smaller networks.



(a) Performance (b) Modularity

Figure 4.21: Importance of hidden nodes in the case with the penalty

Addition of memory

Hidden nodes do not seem to be strongly required for the tasks learning. This fact suggested us a possible explanation for the absence of modularity and even a reduction of modularity as learning proceeds. In fact, the neural network composed of only input and output nodes can not be modular, otherwise this would imply that some inputs are disconnected from some outputs and this assumption seems hard to satisfy because input signals are not correlated. Therefore, once we initialize the neural network with hidden nodes and links, we add an "unnecessary" structure resulting in some detectable modularity, which will be slowly removed by the learning phase, creating isolated nodes or making all the nodes to work together, and so will finally decrease the network modularity.

As a consequence, we assumed that the studied tasks could be inappropriate to observe modularity in networks. A possible clue to have modular structures emerging from the learning phase of two tasks could be the requirement of memory to be accomplished. Therefore, we modified the first task in order to introduce memory. The new aim of $T1$ is to reach the global maximum in a first time and the global minimum after receiving a signal which is active during an instant. The robot has to memorize the change of target.

The modification of $T1$ leads to the addition of one sensor used to receive the signal giving the goal change. Consequently, our artificial neural networks are made of 44 nodes, namely 18 inputs, 22 hidden nodes and 4 outputs.

The performance function of $T1$ is modified to correspond to the new aim. The

robot carries out 40 steps for each of the 10 initial positions of the training set. During the first 20 steps, its aim is to reach the global maximum and we save the mean height on the last five steps. Then, the robot receives the signal and makes the other 20 steps for which we also keep the mean height on the last five steps. We compute values $fit1$ and $fit2$ that are the mean heights on all the displacements for the first and the second part of the walk. The quality of the robot behavior is finally given by

$$f_1 = 0.5f1_{up} + 0.5f1_{down}$$

where

$$f1_{up} = \frac{fit1 - h_{min}}{h_{max} - h_{min}}$$

and

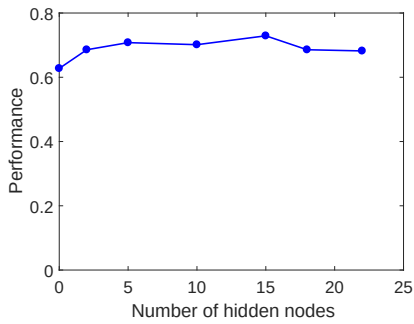
$$f1_{down} = 1 - \frac{fit2 - h_{min}}{h_{max} - h_{min}}$$

The fitness of $T2$, for its part, is unchanged. The ways considered to train robots are the same as previously.

Concerning the topological modularity, results are similar to those obtained for previous experiments. We thus checked if the new version of our experiments requires hidden nodes. Results are shown in Figure 4.22 and, once more, hidden nodes do not seem essential to achieve the tasks. We thought this could be due to our way of designing the problem. We then made two extra attempts by modifying different features.

Hidden nodes	Fitness
22	0.6820
18	0.6857
15	0.7290
10	0.7014
5	0.7078
2	0.6857
0	0.6275

(a) Table



(b) Graph

Figure 4.22: Evolution of the performance according to the number of hidden nodes

Firstly, we considered to change the learning process and to learn $T1$ in two steps. During the first step, we concentrated on the learning of the ascent. Then, the complete task, i.e. the ascent and the descent after the signal, is learned during the second step. Secondly, we slightly modified the first task by considering a variable instant for the signal instead of a fixed one. In this case, robots can receive the signal between the 15th and the 25th step. In both cases, hidden nodes do not seem beneficial to the learning and so, memory has not been introduced in our experiments.

This conclusion could be wrong. Indeed, hidden nodes are not the only way to introduce memory in our problem. It could also be brought by loops in the networks. Consequently, we changed our model in order to prevent loops with the hope to observe differences between networks according to their number of hidden nodes. Results were once more negative.

Our aim was to check if our assumption about the need of memory for modularity emergence was correct. Nevertheless, we failed to design experiments in which hidden nodes are necessary. It could be due to the simplicity of our first task. Therefore, one possible solution would be to add memory in the second task with the hope it would be sufficient to require hidden nodes. Another option would be to work on simpler tasks in order to observe the influence of the conflicting aspect of our tasks on modularity. This is the subject of the following chapter.

Chapter 5

Application to Pattern Recognition

In the previous chapter, we studied the emergence of modularity in neural networks used as robot controllers for abstract robotic tasks. We considered different hypotheses often met in scientific publications such as switching between goals and adding a penalty on the number of connections or on the number of hidden nodes. However, we never observed the emergence of modularity whatever the conditions we contemplated. This lack of modularity is commonly met in systems evolved to attain defined goals by means of evolutionary techniques (see e.g. Eick et al., 2001; Thompson, 2012). Indeed, these networks are usually intricately wired and non-modular. The fundamental reason is that modular structures are generally less optimal than non-modular ones. Typically, there are many possible connections that break modularity and increase fitness. Thus, even an initial modular solution rapidly evolves into one of many possible non-modular solutions, as we have seen in Chapter 4.

Authors of many articles claim that the emergence of modularity does not only depend on hypotheses related to the learning of the tasks, but depends on the tasks themselves. Among them, Kashtan and Alon (2005) suggest that modularity emerges under an environment varying with time in a periodic fashion. That is, they showed that modularity appears when the learning process switch between two tasks with common subgoals. On the contrary, modules are lacking when the second task is random, and, consequently, independent of the first one.

The aim of this chapter is to analyze in detail the importance of this dependency among the tasks. To fulfill this objective, we work on the pattern recognition tasks described by Kashtan and Alon (2005) in their article. Our idea would be to extend their experiments by considering a certain probability of dependency for the two tasks. We would achieve this by imposing that the second task has a probability p to share subgoals with the first one and a probability $1-p$ to be a random task with $p \in]0,1[$. In this way, we would observe the minimum level of dependency, i.e. the smallest

value of p, for the emergence of modularity. By the following, we would like to transpose this experience in other frameworks and particularly to the robotics domain where the tasks are generally more complex.

In a first time, we describe the application in pattern recognition field we worked on and results attained by Kashtan and Alon (2005). Then, we present findings obtained by replicating these experiments. As these results are quite contrasting compared with those from our reference article, we discuss the possible reasons for these differences. Consequently to our puzzling results, no publications have been made on this part of our work.

5.1 Problem Description

The application we focused on is the well studied computational model of pattern recognition using neural networks (see e.g. Bishop, 2007; Ripley, 1996). In this problem, neurons receive inputs from a retina of 4 by 2 pixels. Each pixel is assigned to a binary value (0 or 1). The aim is to recognize objects in the two sides of the retina, namely the left (L) and right (R) 2×2 blocks. An object exists if its four pixels match one pattern of a predefined set (see Figure 5.1), which is chosen at the beginning of each simulation, uniformly random from the available set of patterns, and differs for each simulation. The first task consists in checking the presence of both known left and right objects (L AND R) while the second one checks the presence of known objects on the left or on the right (L OR R).

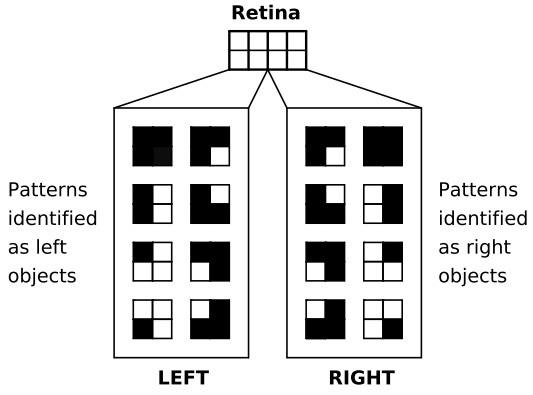


Figure 5.1: Retina and predefined set of known objects

Once more, the application on which we work is very simple and can be seen as a toy example of the pattern recognition domain. Indeed, the input size is very small, as it is made of 8 pixels, contrary to the case of real patterns made of numerous pixels. Moreover, the studied patterns do not imply any noise on the pixels. Nevertheless, our aim is to replicate and to extend results by Kashtan and Alon (2005), who used this

particular application for their research. This simple application is then relevant for our purpose.

Neural networks used to achieve the tasks are feedforward networks made of 23 neurons. These neurons are split up into five layers: eight inputs, three hidden layers with respectively eight, four and two neurons, and one output. Each neuron of the hidden layers has at most three incoming links. For the output, this number is set to two. Connections can be multiple.

Neural networks are evolved by a genetic algorithm. The training set contains 100 different retina patterns among the 256 existing possible configurations. The performance function of each individual is defined by the fraction of correct recognition among the realized tests.

Three different learning configurations are compared throughout this work. The first one, called $C1$ in the following, consists in the learning of the first task. For the second configuration ($C2$), the learning switches between the first and the second task, which have common subgoals while for the last configuration ($C3$), the switch appears between (L AND R) and a random task. This third configuration was not envisaged in the reference experiment. It has been added to check the importance of common subgoals in an environment with varying goals.

The analysis of the networks modularity is performed by using the measure developed by Newman (2006) and detailed in Chapter 1. This measure of modularity is normalized to reduce the effects of the simulations details such as the choice of the neural networks model and the genetic algorithm. The final modularity is computed as a proxy for

$$Q_m = \frac{Q_{real} - Q_{rand}}{Q_{max} - Q_{rand}}$$

where Q_{real} is the modularity of the network, Q_{rand} is the average modularity of randomized networks and Q_{max} is the maximum value the modularity can reach. In both cases, the number of nodes and links is kept the same as our initial network. To compute Q_{real}, the network is transformed into a non-directed network made up of positive weights, which are the absolute values of real weights. On its side, Q_{rand} is calculated as the mean modularity on 1000 random networks. As for Q_{max}, it is estimated thanks to a genetic algorithm by considering the modularity measure Q as the objective function to optimize. The settings of this optimization algorithm are kept identical to those set for the learning of the recognition tasks and Q_{max} is defined as the average Q over 10 simulations.

Results attained in the article of Kashtan and Alon are the following. In the fixed-goal evolution, i.e. for $C1$, nearly perfect solutions (at least 95% recognition) were found after about 21000 generations. The structure of the evolved networks was not modular, with a mean value of Q_m equal to 0.15. For $C2$, the goals were switched every 20 generations. Evolution under these varying goals yielded nearly perfect solutions within about 2800 generations and these networks are able to adapt rapidly after the goal switch. The mean Q_m value of these evolved networks is equal to 0.35. Thus, they have a modular architecture made of two modules, each monitoring a different side of the retina.

5.2 Results

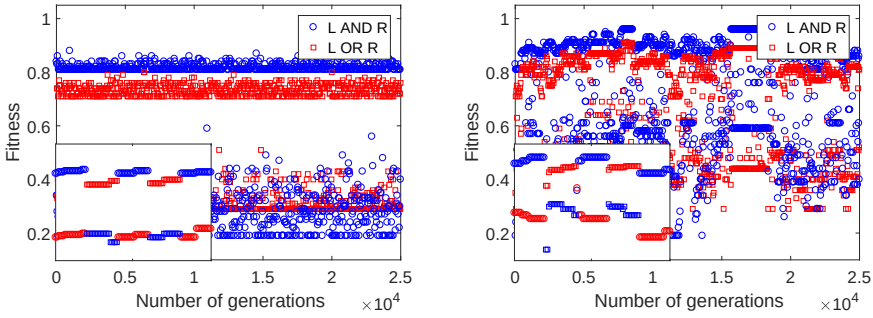
According to Kashtan and Alon, the normalized modularity can be used to decrease the effects of the simulations details. Consequently, we decided to replicate experiments with our own neural networks model and our GA. As results were not those expected, we then used the model of the reference article with the hope to achieve other findings. Results reached with the two models are analyzed and discussed in the following.

5.2.1 First Model

Our first model is the model with discrete weights introduced in the previous chapter. As a reminder, the weights can take values -1 and 1 and the thresholds are real values in the interval $[-1,1]$. Each node sums its weighted inputs and activates its output if this sum divided by the number of incoming links exceeds a given threshold.

The GA associated to this model is the same as in Chapter 4. It has been adapted to keep the feedforward form required in this application. The crossover and mutation rate are fixed to 0.9 and 0.01. The initial number of generations is 25000, as in the reference article. As for the previous application, each experiment is independently replicated ten times.

First, we compared our final fitness with those obtained in the paper. For $C1$, results are similar, with a mean final fitness equal to 0.9650 after 25000 generations. Concerning $C2$, results are more difficult to compare as we only know that "the two tasks are performed nearly perfectly and that the networks adapt rapidly after the goal switch". The maximum fitness reached by each task between two shifts is represented on the left panel of Figure 5.2 for one of our ten simulations.



(a) GA without NI (b) GA with NI

Figure 5.2: Evolution of maximal fitness reached before each switch and zoom of this evolution on 100 generations

We can see that the benefit of one training period seems to disappear after a shift and that this phenomenon occurs all along the learning process. This assumption

is supported by the zoom on 100 generations presented on the left bottom corner of the graph. We assumed this was due to the introduction of new individuals at each generation. Indeed, the mean fitness of random networks for (L AND R) is around 0.7 while it is around 0.3 among networks trained for (L OR R). This fact also holds if we reverse the roles of the two tasks. Thus, new individuals rapidly replace networks trained with the other task and the training is made from scratch after each switch. We then tried to train our networks thanks to a GA in which we do not add new individuals (NI) at each generation. Results are shown on the right panel of Figure 5.2 and seem consistent with our hypothesis. To confirm it, we compute the mean maximum performance values for the two tasks with the two algorithms (see Table 5.1). We can see that fitness are clearly better for the GA without addition of new individuals. The following analyses are then based on this new GA.

	Without NI	With NI
(L AND R)	0.8660	0.9530
(L OR R)	0.8390	0.9360

Table 5.1: Comparison of mean maximum fitness for both GA

After the analysis of the performance function, we moved on to the comparison of modularity between our three learning configurations. In order to use the normalized measure, we have to compute Q_{rand} and Q_{max} by keeping the same number of nodes and links as our trained networks. As the number of links is not fixed, we have to compute these values for each possible number. Figure 5.3 shows the evolution of these values according to the number of links. We observe that modularity logically decreases when the number of links increases.

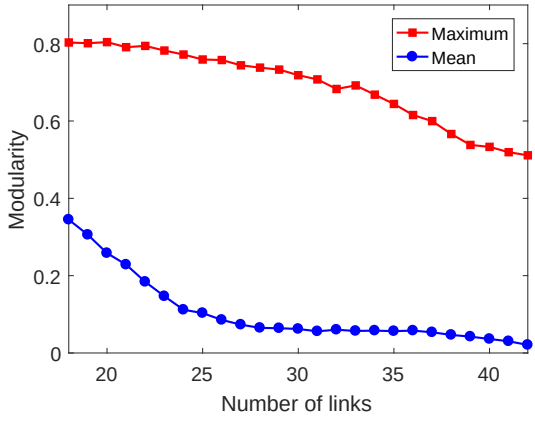
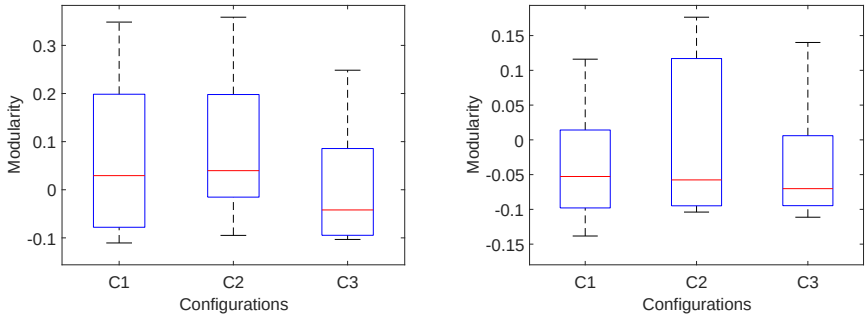


Figure 5.3: Modularity measures according to the number of links in the networks

To compare our three learning configurations, we computed the normalized modularity of the final networks, i.e. the set of ten networks made of the best network of each optimization after 25000 generations. Results are presented by boxplots on the left panel of Figure 5.4. As the p-value of the Kruskal-Wallis hypothesis test is equal to 0.8205, the null hypothesis is not rejected and the medians do not seem significantly different. So, modularity does not appear to emerge from switching between goals. Moreover, values obtained for modularity are very different from those of the article. Indeed, Kashtan and Alon claim that modular networks have modularity values superior to 0.3 while this value is rarely reached by our networks. In addition, 75% of our networks have a modularity value smaller than 0.2 and are then non-modular whatever the learning configuration. We checked if these results can be explained by a premature stop of the learning process. The modularity of final networks obtained after 100000 generations are shown on the right panel of Figure 5.4. We can see that modularity values are even lower and the p-value of the Kruskal-Wallis test is equal to 0.4534, leading to fail to reject the null hypothesis.



(a) After 25000 generations

(b) After 100000 generations

Figure 5.4: Comparison of final modularity for different learning configurations

In the reference article, authors add a penalty on the number of nodes when they compute the performance function of the networks. So, we also tried to add a penalty and checked if it led to the emergence of modularity. Our first idea was to use a penalty on the number of connections as we did in Chapter 4. The weight assigned to its fitness was fixed to 0.3. However, this penalty was not suitable in this case as we obtained final networks without links for the configuration $C1$. This result is consistent as a majority (around 75%) of the retina patterns has an output equal to zero for (L AND R). Such a network without links can thus have a good fitness for this task.

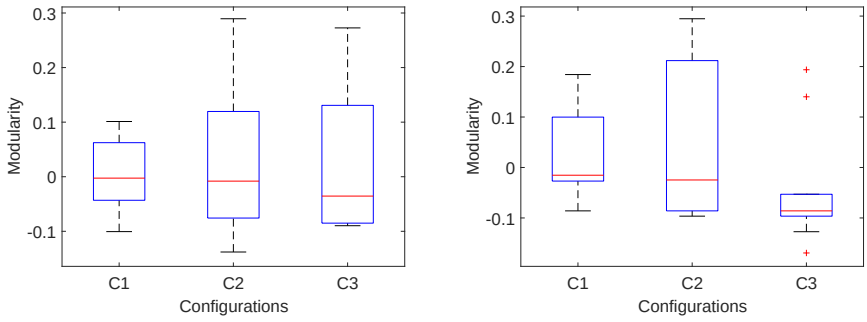
Consequently, we used the same penalty as in the article. This penalty consists in reducing the fitness of 0.01 per additional neuron if more than 12 of the 14 hidden nodes are used. The advantage of this penalty compared to the previous one is that it does not require to assign weights to each fitness. Table 5.2 presents the mean number of links in networks trained without (NP) and with (WP) this penalty. We can observe that this penalty enables to decrease this number. Instead of adding this penalty, we

considered to decrease the number of hidden nodes in our experimental settings. The mean number of links among final networks trained with this reduction of nodes (RN) is also given in Table 5.2. The add of the penalty and the reduction in the number of nodes give similar number of links.

	NP	WP	RN
$C1$	39.4	32.3	33
$C2$	33.8	28.6	29.9
$C3$	34.3	31.4	30.6

Table 5.2: Average number of links in the final networks according to the experimental settings

Figure 5.5 presents the modularity of final networks trained with the penalty on the left and with the initial reduction of nodes on the right. Our results are once more puzzling. Indeed, according to the Kruskal-Wallis tests, whose p-values are respectively equal to 0.946 and 0.088, the medians of the three learning configurations do not seem significantly different.

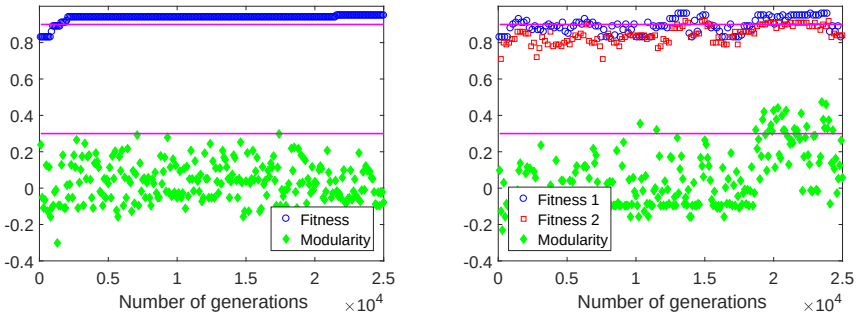


(a) Case WP (b) Case RN

Figure 5.5: Comparison of final modularity for different learning configurations

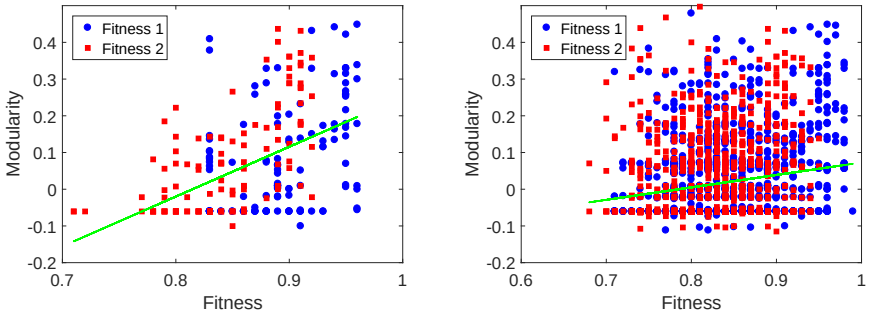
When we analyzed the performance function for $C2$, we remarked that the fitness fluctuates throughout the learning process. Consequently, we decided to observe the evolution of modularity linked to this evolution of fitness. For this, we computed the modularity before each shift for one of our simulations in the case RN. Results are presented in Figure 5.6 on the left panel for the first configuration and on the right panel for the second configuration. We can observe a different behavior of the modularity for the two configurations. Indeed, for $C1$, the modularity almost never overcomes the threshold of 0.3. For $C2$, we can observe an increase of modularity which seems to correspond to a gain in performance. However, this rise is not observable on all simulations. Results are similar for the other cases (NP and WP).

As the increase of modularity seemed to be linked to the one of the fitness, we analyzed in detail the relation between these values. Figure 5.7 shows the relation between performance function and modularity for the simulation represented on the right panel of Figure 5.6 and the linear regression linking these variables. The correlation coefficient is 0.4793 and the link between the two variables is not so evident. It is even worst if we consider the ten simulations to compute this coefficient which is then equal to 0.1675. This situation is represented on the right panel of Figure 5.7.



(a) $C1$ (b) $C2$

Figure 5.6: Evolution of modularity values throughout the learning process



(a) For one simulation (b) For all simulations

Figure 5.7: Modularity values as a function of fitness values

We have observed that modularity fluctuates. So, it could be interesting to consider the comparison of modularity on all generations instead of on final networks. This comparison is presented in Figure 5.8. Even if it is not easily observable, the p-value of the Kruskal-Wallis test is about 10^{-15}, which means that the medians are significantly different. But contrary to the expected behavior, the median of the first configuration seems to be the highest.

In all previous experiments, we have clearly observed that the modularity values are lower than those presented in the article of Kashtan and Alon. In a last test, we compared the performance function of the final networks provided by the article and trained in our experiments. For this, we used a test set made of all possible retina patterns and we focused on the first configuration ($C1$). For the network provided by the article, the performance value is 0.6681 while the mean value is equal to 0.8473 for our final networks. We came to the conclusion that our networks have better generalization abilities but are less modular. Consequently, we got in touch with authors of the article to understand the reasons of these differences. This exchange led to the clarification of some details about the experimental settings and to the study of a second neural networks model.

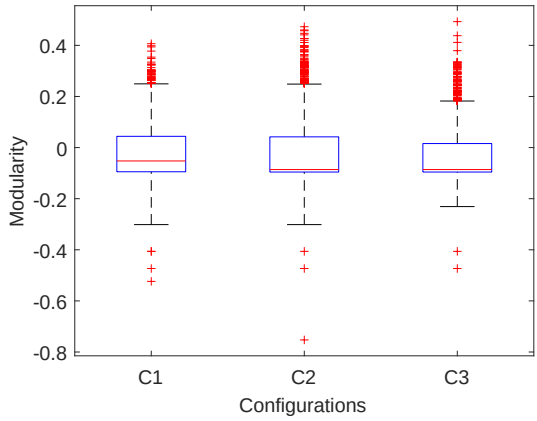


Figure 5.8: Comparison of modularity along the optimization process for different learning configurations

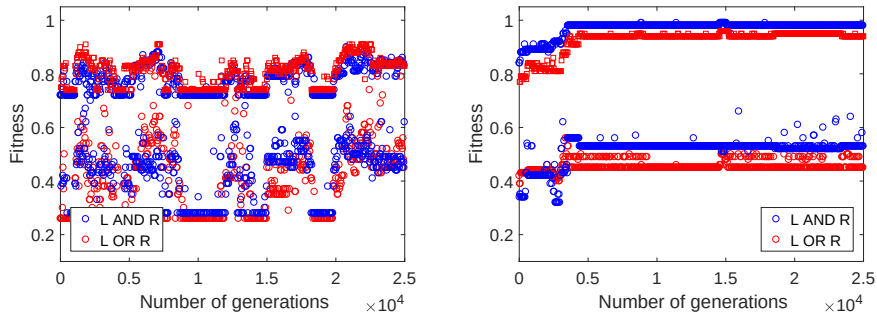
5.2.2 Second Model

The new model is the model used by Kashtan and Alon. Contrary to the previous on, the number of links is fixed as each neuron of the hidden layers has exactly three incoming links. For the output, this number of incoming connections is set to two. Thresholds are now integers instead of real numbers and can take values between -3 and 3. A node activates its output if the sum of its inputs exceeds its threshold (and it is no more divided by the number of incoming links). Neural networks are trained with a performance function taking into account the penalty on the number of hidden nodes. As we could not differentiate the learning configurations up to this point, we neglected $C3$ and we concentrated on $C1$ and $C2$.

With this second model, we considered two GA in order to check if the details of the experimental settings are erased by the normalized modularity. The first one is the GA described for the first model which has been slightly adapted to correspond

to this new model. The second GA is the one used by Kashtan and Alon in their paper. The population size is 600 and the number of generations is fixed to 25000. The selection of parents is made randomly. The crossover and mutation rates are fixed to 0.5. The replacement step is based on the concept of age and elitism is used to avoid the disappearance of best individuals with 150 individuals kept from the parents population.

Concerning the final fitness, results got with these two GA are similar to those of the paper. Indeed, the mean final fitness of $C1$ is superior to 0.9 whatever the GA. For $C2$, we observe in Figure 5.9 that the fitness behavior is different depending on the GA. Indeed, both fitness fluctuate for the first GA (left panel) while they increase and stabilize for the GA from the article (right panel).



(a) First GA (b) Second GA

Figure 5.9: Evolution of the performance values

Before analyzing the modularity, we would like to check if our Q_{rand} and Q_{max} were similar to those of the article. Their computation is made more easily than in the previous model as all networks have the same number of links. Such a comparison is made in Table 5.3.

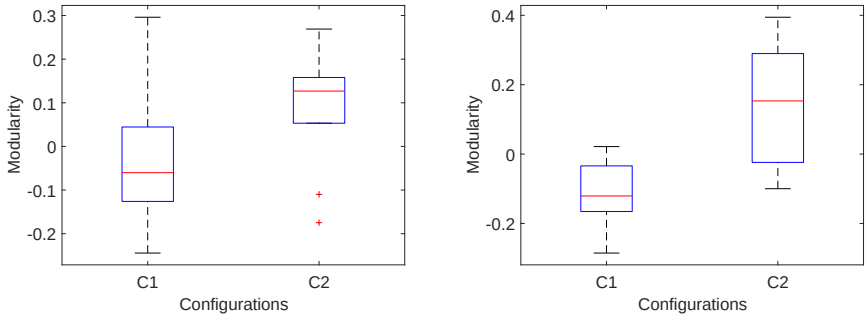
Model	Q_{rand}	Q_{max}
Article	0.36	0.62
Second model	0.40	0.68
Second model WI	0.37	0.62

Table 5.3: Values of Q_{rand} and Q_{max} according to the model

The values of the article are given in the first line. The second and third lines present the values that we obtain by considering networks with and without their inputs (WI). We observe that our values are different, except if we do not consider input nodes when we compute the modularity. Since this choice is not mentioned by the authors, we decided to keep our own values and to compute modularity on our entire

networks. We would like to point out that these values are reached if we consider any number of modules. If we fix this number to two, results are completely different and very far from those of the article.

Figure 5.10 presents the modularity of final networks trained with the first GA on the left and with the second one on the right. We finally observe a difference between the modularity of networks trained with configurations $C1$ and $C2$. The p-values of the Kruskal-Wallis tests are respectively equal to 0.0588 for the first GA and 0.0032 for the second one, wich means that medians are significantly different for the second GA only. Nevertheless, the modularity values are once more far away from those obtained in the reference article.



(a) First GA

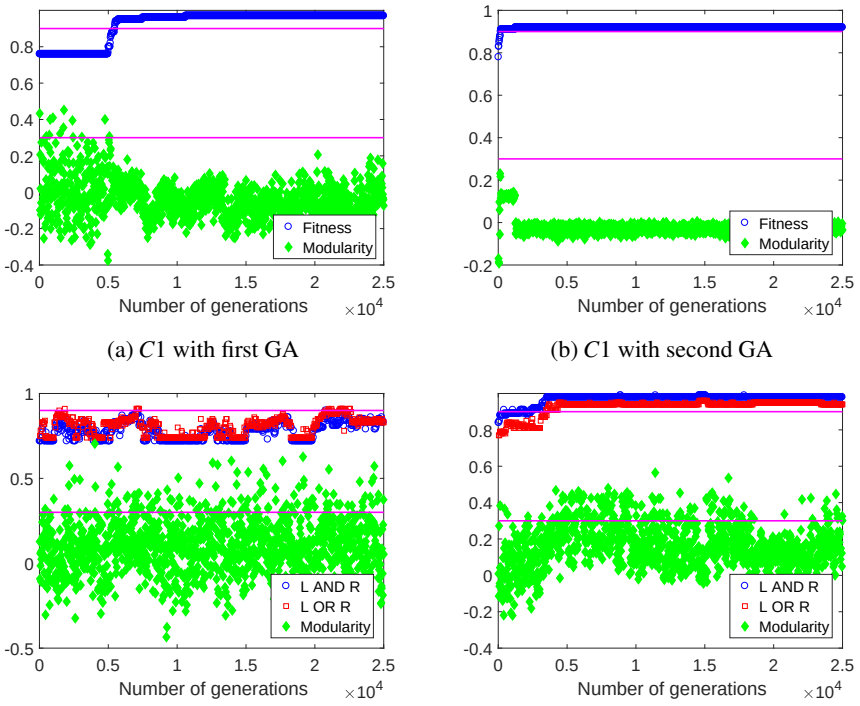
(b) Second GA

Figure 5.10: Comparison of final modularity for different learning configurations

We also analyzed the evolution of modularity during the learning process. Results are shown in Figure 5.11. One simulation of each configuration is represented on the left panel for the first GA and on the right panel for the second one. We can see that modularity varies less strongly when networks are trained with the GA from the article. This fact is consistent with what we observed concerning the fitness values which are also more stable for this GA. However, even if less spread out, variations are alway sizable and the mean modularity given in the article is not reached.

While analyzing our first model, we have shown that a fitness superior to 0.9 did not seem sufficient for the increase of modularity observed in our figures. Consequently, we implemented a simple model of neural networks to test some possible criteria responsible for this appearance. Among this criteria, we can cite some individual features such as the maximal fitness of the trained or untrained task superior to a given threshold. We also thought to features such as the number of generations since the fitness is superior to a given threshold. We considered a network without hidden nodes. The inputs were binary values that indicate if the conditions were fulfilled. The output was also a binary value assigned to 1 if the network is modular. We used a GA to find the right connections for a correct recognition of modular networks. Unfortunately, any learning process could lead to the improvement of the fitness despite the 16 criteria we considered.

Throughout our study, we never succeeded in obtaining values of modularity presented in our reference article and we never observed the emergence of modularity, whatever the considered conditions. Even when the performance functions seem to stabilize, the modularity values oscillate. Moreover, contrary to what expected with the measure of normalized modularity, results seem very sensitive to experimental settings such as the neural networks model and the GA.



(a) $C1$ with first GA

(b) $C1$ with second GA

(c) $C2$ with first GA

(d) $C2$ with second GA

Figure 5.11: Evolution of modularity values along the learning process

Following our research, we have serious doubts about the general statement of Kashtan and Alon according to which modularity emerges from switching between goals with common subgoals. In our opinion, each parameter has been carefully chosen and oriented to observe this appearance of modularity. Indeed, the used GA is quite unusual in the choice of the selection operator and the crossover and mutation rates. Moreover, modularity is studied thanks to a normalized measure and the threshold for observing a modular network seems quite arbitrarily chosen. Finally, networks used for the analyses are selected according to their fitness and this selection is very restrictive. Since the publication of Kashtan and Alon, other researchers have worked on this particular application to pattern recognition. Among them, Clune et al. (2013) have shown that these varying goals are not required for the emergence of modularity,

even if they accentuate the phenomenon. In their paper, modularity has appeared by adding a penalty on the number of links in the networks. Once more, these results have been found with very particular experimental settings. Indeed, Clune et al. used a stochastic multiobjective GA based on Pareto optimality, in which the penalty is only used for 25% of the generations. Our conclusion is that the emergence of modularity is not so easy to observe and is more sensitive to the experimental details than to a given condition.

Conclusion

Synthesis

In this thesis, we have challenged the emergence of modularity in the framework of evolutionary learning in which artificial neural networks are trained by evolutionary techniques to achieve predefined goals. More specifically, we have focused on applications in the fields of evolutionary robotics and pattern recognition. Our main goals were to compare the performance of modular and non-modular networks and to investigate the necessary conditions for the emergence of modularity in evolved networks.

As a first step, we have focused on a toy application in the field of evolutionary robotics. The aim was to train virtual robots to reach the peak of a surface and to avoid zones where they lose energy in a virtual arena. These two tasks were conflicting and shared the same outputs, features rarely met in the literature linked to this domain. In a preliminary phase of analysis, we have shown that the first task was the easiest to achieve. We have also compared different multiobjective genetic algorithms and different fitness allocation methods for GA based on Pareto optimality. This comparison led us to select and work only with the weighted GA in the subsequent part of our work. Moreover, we have shown that the best fitness allocation method seems to be dependent on the number of objectives. We have also considered different ways to decrease the number of links in evolved networks, with the conclusion that all these ways are equivalent and strongly stochastic.

With this first application, we have extended the work of Calabretta et al. (2008) in a more general context. Indeed, in the latter research, authors have shown that a non-modular neural network can be trained so as to achieve the same performance as a modular one in accomplishing multiple classification tasks. Moreover, they have concluded that the best performance is attained by training first on the hardest task and successively also on the second. Although our context of study is very different from the one of this prior article in the considered tasks as well as in the effect of the learning process, the conclusions coincide. Indeed, we have also shown that non-modular networks are as efficient, and even more, as modular ones and the best performance

is attained with the same learning sequence. We have explained these results by the autocorrelation of the performance landscapes. Finally, we have highlighted that this better performance only occurs if the level of generalization is sufficient for the hardest task.

This thesis has been initiated with the preliminary idea that evolved networks should exhibit the same modular organization as in biological and engineered networks. This idea was supported by our way of thinking as well as by reference articles. Consequently, we have studied the emergence of modularity in our robotic tasks. We considered different hypotheses commonly met in the literature such as switching between goals and adding a penalty on the number of connections or hidden nodes. Nevertheless, we have never observed modularity in our evolved networks whatever the added conditions. Moreover, we have shown that even initial modular networks rapidly evolve in non-modular ones.

Consequently to these results, we have tried to replicate the study led by Kashtan and Alon (2005) on an application to pattern recognition. In this work, authors claim that modularity emerges from switching successively between two tasks with common subgoals. The aim of the problem is to recognize predefined objects in the two sides of a retina and the two tasks with common subgoals consist in checking the presence of both known left and right objects and checking the presence of known objects on the left or on the right. The conclusion is achieved after comparing this learning scheme with the learning of both tasks simultaneously and the learning of the first task switched with a random task. Contrary to results presented in that article, our results show that modularity is not observed in networks trained with evolutionary techniques. Moreover, we claim that the presence of modularity obtained in the article is very sensitive to the experimental settings and the particularity of these settings do not allow to draw general conclusions about the necessary conditions for the emergence of modularity. Let us stress that our claim is supported by recent papers where authors also criticized the possibility of modularity to emerge straightforwardly (Nitash et al., 2018; Valverde, 2017).

In conclusion, even if evolutionary techniques mimic the basic process of biological evolution, they do not reproduce the organization of real networks into modules. A possible explanation can be given by the simplification of this process and by the fact that some important aspects are not taken into account. For example, nodes of real networks are separated by physical distances and spatially close nodes are consequently more probable to be linked. In our designed networks, the number of connections is regarded as an essential criterion but this notion of distance is completely missing.

As already mentioned, modularity is a controversial issue. Contrary to authors cited up to now, some researchers have accepted the idea that modularity does not emerge from evolutionary process for a while. As an organization into modules remains interesting and brings substantial advantages, they have developed high-level processes to promote the presence of modularity. Among such techniques, we can cite module libraries and automatically defined functions, that represent subsystems as modules and used them as building blocks (see e.g. Koza et al., 2006). These processes, despite being currently unknown in biology, are shown to improve artificial design.

Perspectives

The immediate perspectives to this work are numerous. Indeed, for lack of time, we have had to make choices and some additional search directions have been put aside for our different applications. The main ones are cited in the following.

Throughout this work, we have implemented different models of artificial neural networks, as well as GA adapted to these models. A first perspective would be to gather our numerical codes together and to provide a catalog for the learning of any given tasks. The same process could be performed for codes enabling the analysis of resulting networks.

In Chapter 4, we have failed to design a task derived from the climb of the peak and requiring hidden nodes to be achieved. We have supposed that this could be due to the simplicity of the initial task. We could try to develop a task with memory, and so a task requiring hidden nodes, starting from the avoidance of zones where robots lose energy. It would represent a way to go further in the analysis of modularity emergence in the complex framework of robotic tasks. It could then allow us to strenghten our results about the absence of modularity in our networks. Another aspect that could be considered in this research for modularity would be to use a genetic algorithm based on Pareto optimality instead of a weighted sum of objectives. It would give us the occasion to compare the modularity attained by the different Pareto solutions.

Concerning the application in the domain of pattern recognition, we have only studied topological modularity, as it was made in the reference paper. It could be interesting to analyze the functional modularity of the evolved networks. We could also use a GA based on Pareto optimality in this second application. The results of this last research point would then be compared with those of Clune et al. (2013), in which authors use this multiobjective GA in a stochastic way. This could help us to reassert that modularity is observed with very specific experimental settings.

Evolutionary techniques do not allow the emergence of modules with simple ANN models. But another point of interest could be to explore other models taking into account more aspects of biological networks, for example networks with various distances between the nodes. Another perspective would be to study the techniques developed to promote modularity. In this sense, we could assume that networks are composed of small modules made up of three of four nodes that can not be separated. Another option would be to initiate the learning process with modular networks and to make some groups of nodes inseparable.

Appendices

Appendix A

Quantum Neural Networks

The research presented in this appendix is linked to the framework of quantum computing. This domain has generated a lively interest for the last two decades, since the discovery of a quantum algorithm able to factorize large integers in polynomial time (Shor, 1999). In fact, the demand for better performance of computers strongly increases and quantum computation could be the answer to overcome the limitations of current supercomputers. Consequently, some multinational technology companies including Google, IBM and Intel are currently racing for building useful quantum systems. In the meantime, some quantum computing companies such as D-Wave have been developed to deliver quantum systems and software.

However, even in the case of relatively simple problems, the search for a quantum algorithm is not trivial. This fact is for instance illustrated by the parceled development of the solution for the well known problems of Deutsch (1985) and Deutsch-Jozsa (1992). Another complication of quantum computing is its physical feasibility. Indeed, quantum computing requires the development of quantum operators working on systems of qubits. Until now, researchers have been able to physically produce operators dealing with small systems composed of one or two qubits. Fortunately, it has been proved that any quantum operator can be built as a combination of these concretely realizable operators (Barenco et al., 1995). But, once more, the development of the right combination is not a trivial problem.

In this work, we study the possibility to design appropriate combinations of quantum operators to achieve defined tasks or computations. In particular, we make use of neural networks endowed with quantum gates linked together to achieve our goal. As the construction and the learning process of these networks are roughly inspired by artificial neural networks, we decide to name them quantum neural networks (QNN).

This appendix is based on a paper entitled “Quantum neural networks achieving quantum algorithms” which has been published in “Advances in Artificial Life and Evolutionary Computation”. The first section is dedicated to the introduction of basic principles of quantum computing. The results of our numerical experimentation are then presented in the second section.

A.1 Background to Quantum Computing

The aim of this section is to present concepts of quantum computing that are required for a good understanding of our work. We first present the basic units of quantum computation and their associated operators. Then we describe the simple problems of Deutsch and Deutsch-Jozsa which are regarded as the trial problems of our study. Let us remark that these presentations are made from an algorithmic point of view and do not broach the physical complexity of quantum computing.

A.1.1 Quantum Bits and their Operators

As the bit is the fundamental unit of classical computation, quantum computation is developed upon the similar concept of quantum bits, commonly called qubits. These quantum bits are modified by particular operators, known as quantum gates, which respect their inherent properties (see e.g. Mermin, 2010; Nielsen and Chuang, 2000).

Quantum bits

Quantum bits have basic states $|0\rangle$ and $|1\rangle$, which correspond to logical states 0 and 1 for classical bits. But, contrary to the latter ones, qubits as any quantum system can also be in a superposition of states

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where α and β are complex numbers constrained by the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. Usually, a qubit is considered as a vector in $\mathbb{C}^2$ and the basic states are then seen as a pair of orthonormal basis vector

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

As qubits are quantum objects, this superposition of states is not observable. Once the qubit is measured, the superposition is lost and the qubit will be found in the state $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. This step is called the decoherence (see e.g. Hornberger, 2009).

In the same way, we can define systems with n-qubit whose basic states are written as

$$|x_n x_{n-1} \ldots x_1\rangle \text{ where } x_i \in \{0,1\} \text{ for } i = 1,\ldots,n.$$

Such states can be written as a tensor product of qubits but quantum computation is much richer. Indeed, thanks to the superposition, a 2-qubit can be in the state

$$\alpha\,|00\rangle + \beta\,|11\rangle$$

which can not be constructed using tensor products of qubits. This property is called the entanglement (see e.g. Mintert et al., 2009) and is proper to quantum systems.

Quantum gates

Quantum gates, working on a qubit or an n-qubit system, are obtained using unitary operators. Hence, they are reversible and they respect the normalization condition in such a way that the norm of the qubit system is preserved. They are the basic building blocks, combined to form quantum circuits. Widely used qubit operators and their matrix representation are presented below.

- Identity operator I:

$$\begin{array}{l} I|0\rangle = |0\rangle \\ I|1\rangle = |1\rangle \end{array} \qquad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- NOT operator X:

$$\begin{array}{l} X|0\rangle = |1\rangle \\ X|1\rangle = |0\rangle \end{array} \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- Operator Y:

$$\begin{array}{l} Y|0\rangle = i|1\rangle \\ Y|1\rangle = -i|0\rangle \end{array} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

- Operator Z:

$$\begin{array}{l} Z|0\rangle = |0\rangle \\ Z|1\rangle = -|1\rangle \end{array} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- Hadamard transformation H:

$$\begin{array}{l} H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{array} \qquad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- Phase operator S:

$$\begin{array}{l} S|0\rangle = |0\rangle \\ S|1\rangle = i|1\rangle \end{array} \qquad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

- $\pi/8$ operator T:

$$\begin{array}{l} T|0\rangle = |0\rangle \\ T|1\rangle = \frac{\sqrt{2}}{2}(1+i)|1\rangle \end{array} \qquad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$$

The most used 2-qubit operator is the controlled-not operator (C_{not}), also called the 2-qubit XOR gate, which is represented by

$$\begin{array}{l} C_{not}|00\rangle = |00\rangle \\ C_{not}|01\rangle = |01\rangle \\ C_{not}|10\rangle = |11\rangle \\ C_{not}|11\rangle = |10\rangle \end{array} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Its effect consists in changing the state of the second qubit if and only if the first one is equal to $|1\rangle$. In the same way, we can define other controlled gates by combining this rule and the qubits presented previously. It has been proved that the controlled-not gate combined with all qubit gates form a universal set for quantum computation (Barenco et al., 1995).

A.1.2 Deutsch and Deutsch-Jozsa Problems

The Deutsch problem consists in deciding if a binary function $f : \{0,1\} \to \{0,1\}$ is constant using only one function evaluation. It is clear that this is not possible in the classical framework, where two function evaluations are needed. To achieve this goal, we have a quantum black box, called oracle, at our disposal. This oracle computes one of the four possible functions, i.e. forming all the possible couples $f(u) = v$ with $u, v \in \{0,1\}$, by applying an unitary operator U_f defined as

$$U_f(|x\rangle |y\rangle) = |x\rangle |y \oplus f(x)\rangle$$

where $|x\rangle$ and $|y\rangle$ are the qubits of the system and $\oplus$ is the addition modulo 2. The quantum circuit representing the solution of this problem is presented in Figure A.1.

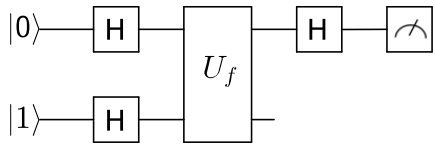


Figure A.1: Quantum circuit for the resolution of the Deutsch problem

The first qubit is initialized to $|0\rangle$ while the second one is set to $|1\rangle$. Then, an Hadamard gate is applied to the two qubits leading to the state

$$\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$$

before calling the oracle. The state is therefore defined as

$$\begin{aligned}
&\frac{1}{2}\left[|0\rangle \left(|f(0) \oplus 0\rangle - |f(0) \oplus 1\rangle\right) + |1\rangle \left(|f(1) \oplus 0\rangle - |f(1) \oplus 1\rangle\right)\right] \\
&= \frac{1}{2}\left[(-1)^{f(0)} |0\rangle \left(|0\rangle - |1\rangle\right) + (-1)^{f(1)} |1\rangle \left(|0\rangle - |1\rangle\right)\right] \\
&= \frac{1}{2}(-1)^{f(0)} \left[|0\rangle + (-1)^{f(0)\oplus f(1)} |1\rangle\right] \left(|0\rangle - |1\rangle\right).
\end{aligned}$$

We therefore find that the state of the first qubit is, up to an irrelevant factor $(-1)^{f(0)}$, given by

$$\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{f(0)\oplus f(1)} |1\rangle).$$

An Hadamard gate is finally applied on it leading to

$$\begin{aligned}
&\frac{1}{2}\left(|0\rangle + |1\rangle + (-1)^{f(0)\oplus f(1)} |0\rangle - (-1)^{f(0)\oplus f(1)} |1\rangle\right) \\
&= \frac{1}{2}\left(\left[(1 + (-1)^{f(0)\oplus f(1)}\right] |0\rangle + \left[1 - (-1)^{f(0)\oplus f(1)}\right] |1\rangle\right)
\end{aligned}$$

$$= \begin{cases} |0\rangle & \text{if } f(0) \oplus f(1) = 0 \text{ i.e. } f(0) = f(1) \\ |1\rangle & \text{if } f(0) \oplus f(1) = 1 \text{ i.e. } f(0) \neq f(1) \end{cases}$$

and this first qubit is measured. If it is found in the state $|0\rangle$ then the function is constant. Otherwise, namely if the measure determines that the qubit is in the state $|1\rangle$, the function is not constant.

The Deutsch-Jozsa problem is a generalization of the Deutsch problem for a binary function $f : \{0,1\}^n \to \{0,1\}$. In this case, the aim is to decide if the function is constant or balanced, which means that we get 0 for half of the function evaluations and 1 for the other half. The resolution is very similar to the previous one and is presented in Figure A.2. Indeed, the qubits are initialized similarly i.e. $|0\rangle$ for the n first qubits and $|1\rangle$ for the last one. Then, an Hadamard gate is applied on all qubits before the oracle intervention. An Hadamard gate operates again on each of the n first qubits. The function is constant if all of them are finally in the state $|0\rangle$.

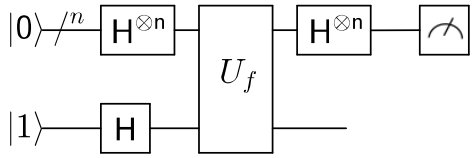


Figure A.2: Quantum circuit for the resolution of the Deutsch-Jozsa problem

Even if these two problems are relatively simple, let us remark that finding their solution is not trivial. Indeed, the algorithm originally proposed by Deutsch (1985) was probabilistic and was successful with a probability of one half. In 1992, Deutsch and Jozsa developed a deterministic algorithm but it required two oracle calls to succeed. The current solution, with only one function evaluation, has been proposed by Cleve et al. (1998). This shows that even in relatively simple cases, there is a need for a general strategy allowing to construct the algorithm associated to the problem at hand.

A.2 Implementation and Results

We can now present our model of quantum neural networks, which is based on a model proposed by Deutsch (1989). The idea is to build a network whose nodes are quantum gates and connections bring quantum information through qubits. The network is feedforward and the number of nodes is constant in every layer. QNN are trained to achieve specific goals by means of evolutionary optimization methods.

Our study is divided in three steps. We first consider to apply our methodology on the trial problems of Deutsch and Deutsch-Jozsa. Then, we compare the performance of different optimization methods. We close our study by identifying a set of universal quantum operators and designing quantum gates with operators from this set.

A.2.1 Deutsch and Deutsch-Jozsa Problems

For the trial problems of Deutsch and Deutsch-Jozsa, we have not considered a set of universal gates. The nodes could only be assigned to one of the three qubit gates I, X and H or to the oracle. Let us remind that this oracle is only used in one layer of the network, but has an effect on all qubits of the layer. Indeed, our $n+1$ qubits, handled separately, have to be turned into a $(n+1)$-qubit system used as a whole by the oracle. This transformation is carried out using the Kronecker tensor product (Neudecker, 1969). The inverse operation is then executed after passing the oracle to recover our $n+1$ qubits.

Quantum neural networks are evolved to solve the target problem by a genetic algorithm. As artificial neural networks, QNN are represented by matrices and the used GA is then similar to those presented for other applications. The selection is performed by a roulette wheel. The genetic operators are the 1-point crossover and the uniform mutation. Their respective rates are 0.9 and 0.01. The population size is 100 and the maximum number of generations is 10000. The replacement step is performed by ranking the unified set of parents and children and selecting the best among them. The space exploration is guaranteed by the introduction of new individuals at each generation. The training environment contains the functions to classify and the fitness of each individual is defined by the fraction of correct classifications. As previously, all experiments have been replicated 10 times. Results presented here are means on these 10 simulations.

First tests on the Deutsch problem have been performed with an initialization of the first qubit to $|0\rangle$ and the second one to $|1\rangle$. All simulations led to the correct solution. The only difference observed among the different solutions relates to the operator applied on the second qubit in the last layer, as it is shown in Figure A.3. This difference is not important since only the first qubit is measured to answer the question. This solution was already found at the first generation of the GA, which could be explained by the small number of possible networks (243).

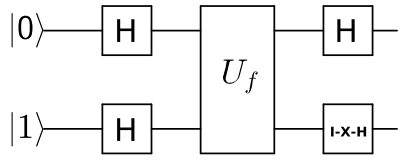
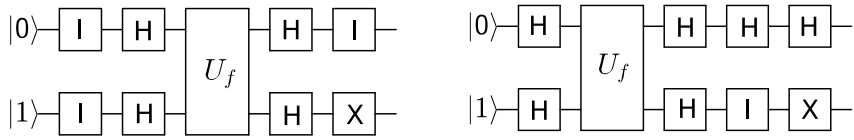


Figure A.3: Solution of the Deutsch problem obtained with our methodology

Then, different parameters have been altered to observe the consequences on the learning and the final algorithm. These parameters are the number of layers in the network, the initialization of the qubits and the measured state to be constant or balanced. When the number of layers is increased, we observe that a solution is always found even if the number of possible networks increase exponentially. Indeed, the number of admissible solutions also increase exponentially according to the number of layers.

For example, if we consider five layers in the network, the two networks presented in Figure A.4 have the same effect on the quantum states.



(a) Circuit 1 (b) Circuit 2

Figure A.4: Example of two equivalent solutions for the Deutsch problem if the network is made of five layers

If we exchange the initialization of the two qubits, we have to consider a network of at least four layers to find a solution. And, most of the time, the solution consists in replacing the network in the previous initialization, which means that a NOT operator is applied to each qubit in the first layer. Results are similar if we alter the initialization by setting both qubits to $|0\rangle$ or $|1\rangle$.

In case we switch the state to measure to have a constant ($|1\rangle$) or balanced ($|0\rangle$) function, we can find a solution whatever we take as initialization of our qubits. The smallest network, given in Figure A.5, is obtained if both qubits are initialized to $|1\rangle$. Even if less frequently met, this scheme has already been presented in the literature (see e.g. Mermin, 2010). For other initializations, the solution is made of four layers. We have also tried to look for a solution by measuring the second qubit instead of the first one but it has not worked whatever the considered initialization and configuration. This result seems consistent as such a solution has never been introduced in the literature.

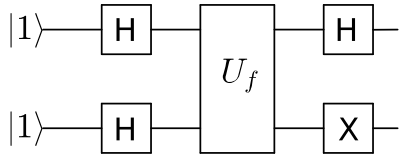


Figure A.5: Solution for the Deutsch problem if a constant function is associated to the state $|1\rangle$

Concerning Deutsch-Jozsa, we have tested different sizes of the problem. Let n be the number of variables of the function, then the number of input states of the function is 2^n and the number of possible balanced functions is given by the number of combinations of 2^{n-1} units taken among 2^n. Figure A.6 presents the number of possible networks as a function of n (left panel) and the mean number of generations to reach the solution with our GA for each of this dimension (right panel). We can see in our two graphs that the increase as a function of n is exponential.

From $n = 3$, we have noticed that our $(n+1)$-qubit systems could not always be split into $n+1$ qubits. This is due to the property of entanglement of quantum states. Indeed, some qubits that are combined with the tensor product are modified by the oracle in such a way that they can no more be separated properly. In this case, we have considered either to keep all functions or to exclude functions that lead to entangled states. In the first case, we could hardly get a fitness of 1. In the second case, we have obtained a fitness of 1 but simulations were longer as a preliminary test was needed to remove this type of functions.

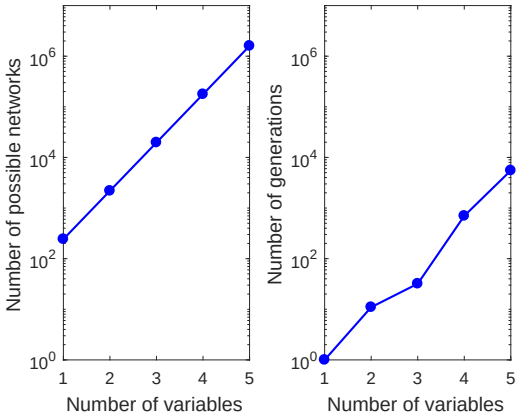


Figure A.6: Number of possible quantum networks and number of generations to reach the solution for the Deutsch-Jozsa problem as a function of the number of variables

A.2.2 Performance of the Optimization Algorithms

Before going further, we have considered the possibility of using optimization methods different from genetic algorithms. In this way, we have implemented a simulated annealing and a random search which are global optimization methods described in Chapter 3. Figure A.7 shows the number of function evaluations required by each method to reach the solution for different sizes of the Deutsch-Jozsa problem. We can observe that these numbers are very similar for the three methods. Therefore, the genetic algorithm and the simulated annealing do not appear more efficient than the random search.

This fact could be explained by our way of coding and modifying our model of quantum neural networks. Indeed, the shift of the oracle from one layer to another because of the application of a mutation for the GA leads to important changes in networks. This remark also holds for SA, as the oracle can be shifted during the exploration of the space of solutions. Because networks are pretty small, these big changes can modify them as strongly as it is made by random search.

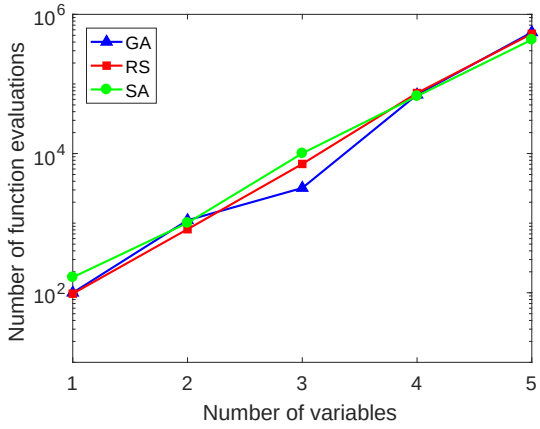
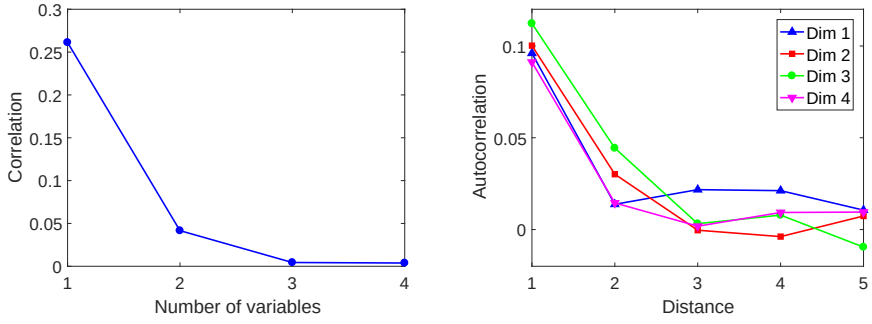


Figure A.7: Comparison of the optimization algorithms

Another explanation could be glimpsed by the analysis of the fitness-distance correlation (FDC) coefficient and the autocorrelation of the function landscape described in Chapter 2. As a reminder, the FDC coefficient measures the correlation between the objective function of a candidate and its distance to the optimal solution. In our case, the distance between any two quantum gates is fixed to one. Moreover, we do not consider the last operator applied on the last qubit since this one does not have any influence on the result. As for the autocorrelation, it measures the correlation between neighboring candidates. For this measure, we consider neighbors at distances from one to five. Results of these two indicators for different sizes of the Deutsch-Jozsa problem are presented in Figure A.8.



(a) Fitness distance correlation

(b) Autocorrelation

Figure A.8: Indicators analysis for different sizes of Deutsch-Jozsa problem

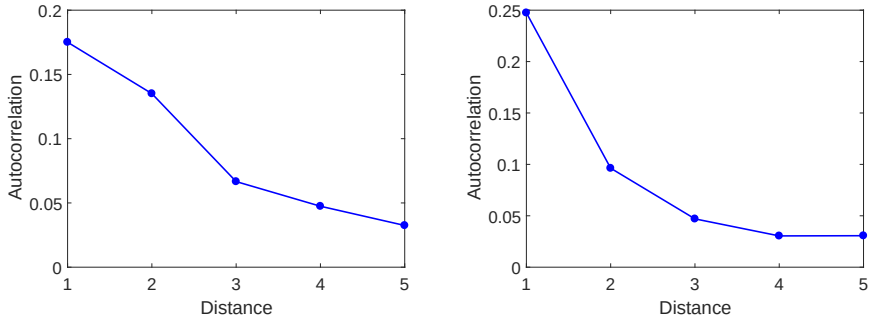
We can see that these two coefficients are quite low, whatever the size of the problem. This observation reinforces our intuition that GA and SA are no more efficient than RS for this application. Indeed, if correlation does not exist between the distance to the solution and the objective function, it can not be assumed that the best individual will be found by crossovers and mutations on good individuals. Similarly, the absence of correlation between neighbors removes any advantage to an optimization method such as SA that travels from one candidate to its neighbors.

A.2.3 Quantum Gates Construction

Our methodology enables us to develop quantum algorithms solving problems of Deutsch and Deutsch-Jozsa without requiring any particular knowledge except for the function to reproduce. Indeed, the appropriate algorithm appears following the learning process applied to a network composed of standard gates. Given the difficulty to develop quantum algorithms and the small number of such algorithms, we think that our results are promising even if the increase according to the number of variables is exponential. Consequently, we have considered to exploit our methodology for the implementation of quantum gates.

Our aim is to identify a set of universal gates and to develop other gates by combining the operators belonging to this set. We follow the statement of Nielsen and Chuang (2000) and work with a set made of six qubit gates to whom the controlled-not gate has been added. The qubit gates are I, H, S, T and their adjoint operators, whose matrices are obtained by conjugating and transposing the matrix of the initial operator. As I and H are self-adjoint, we only have to add S^* and T^*.

Before starting our optimization, we have analyzed the two indicators presented above in order to choose the most appropriate method. For this, we have considered the objective function of the controlled-Z gate and the Toffoli gate, which is a generalization of the controlled-not gate for three qubits. The FDC coefficients for these two problems are respectively equal to 0.1048 and 0.2010. The autocorrelation of the function landscape is represented in Figure A.9.

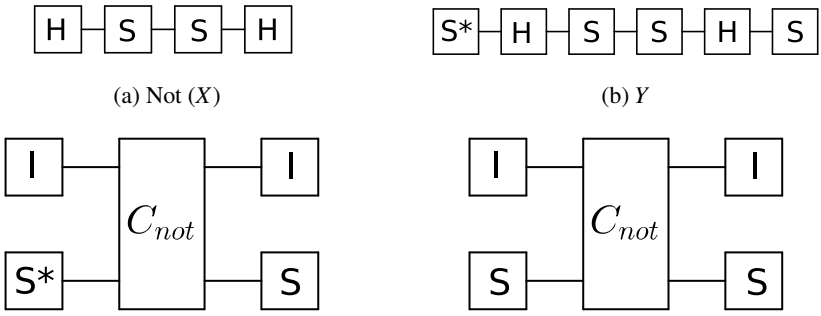


(a) Controlled-Z gate (b) Toffoli gate

Figure A.9: Autocorrelation analysis for two quantum gates

Once more, these quantities are pretty low. Consequently, we have decided to replace our genetic algorithm by a simulated annealing. Indeed, the genetic algorithm requires more cpu time due to crossover and mutation process for analogous results. Our SA has a temperature that is initialized to 1, in such a way that a candidate decreasing the objective function by 0.5 has a probability of $\frac{2}{3}$ to be accepted (Kirkpatrick et al., 1983). The cooling parameter is fixed to 0.99995 for a slow diminution of this probability, with the aim to explore the space of solutions as much as possible.

Firstly, we used our QNN model and our simulated annealing to design the qubit gates that were not part of the defined set, i.e. the X (NOT), Y and Z gates. The Z gate is quite easy to rebuild as it only requires a sequence of two Hadamard gates. On the contrary, X and Y respectively claim four and six layers and are represented in Figure A.10. Then, we have succeeded in recreating the 2-qubit gates controlled-Y and controlled-Z, which are also represented in Figure A.10. Although it has been proved theoretically that all these gates could be rebuilt from a set of universal gates, let us note that we hereby provide their explicit scheme for the first time.



(a) Not (X)

(b) Y

(c) Controlled-Y

(d) Controlled-Z

Figure A.10: Quantum gates designed using our methodology

Nevertheless, we have quickly been confronted to one big limitation of our model, which is the exponential increase of the number of possible networks according to its size. This increase, combined with the absence of correlation given by our indicators for the objective function, makes the resolution impossible in reasonable time for networks with more than about 15 gates.

Despite this limitation, we can envisage to improve the efficiency of our method. Firstly, we can decrease the number of possible networks by fixing the number of controlled-not gates, and stronger, by fixing the number of one qubit gates that differ from the identity. But, even with these constraints, the size of the resolvable networks will be limited. Indeed, we know that a Toffoli gate requires 13 layers of 3 qubits to be designed from our predefined set (Nielsen and Chuang, 2000). With our model, even if we fix the number of needed controlled-not gate, the number of possible networks among which the solution has to be found is superior to 10^{26}.

Another option would be to enlarge our initial set with new quantum operators as soon as we find their structure. Improvements can also be imagined on the learning process. For example, we can consider the addition of a penalty in order to avoid useless sequences of operations.

Appendix B

Applications in Optics

As mentioned in the introduction, these six years of thesis have also given us the opportunity to collaborate with researchers from the department of physics. The aim of this collaboration was to develop a genetic algorithm for the optimization of problems in the framework of optics. Two applications have been studied during this work.

The surface of a LED can be covered by periodic structures. The first application consists in optimizing their geometrical and material parameters to maximize the extraction of light. The parameters to optimize are the period and the height of the structures, as well as their center and their concavity on the left and right sides. The second application deals with the optimization of solar thermal collectors. To make them efficient, it is required to maximize α, which represents the fraction of the solar spectrum that is absorbed by the system, while minimizing the thermal emittance ε, which represents the fraction of the absorber black-body spectrum that escapes the system. This aim can be reached by adjusting the geometrical parameters that characterize the collectors.

In these two applications, the parameters to optimize are real values coded thanks to the Gray code, which is an ordering of the binary numeral system such that two successive values differ in only one bit. The population size (n_{pop}) is 100. At each generation, $n_{pop}/2$ individuals are selected to be the parents. The method of selection is the roulette wheel selection in which the fitness of each individual is based on its rank. The parents are transfered to the next generation and are used to generate the children. The crossover operator is the 1-point crossover with a rate of 0.9 and each bit of the children chromosomes has a probability of 1% to be reversed. The bottom 10% of the population are replaced by random individuals. This implementation does not ensured the legacy of best individuals. Consequently, a test has been added to reinsert best individuals if needed.

Concerning the first application, the optimization of its parameters by the GA led to a light-extraction efficiency of 7%. This corresponds to a relative increase of 89% compared to a flat GaN whose light-extraction efficiency is equal to 3.7%. If

the dielectric constant is considered as an additional adjustable parameter, the light-extraction efficiency achieves 10.8%, which represents a relative increase of 192% in comparison to a flat GaN. As for the second application, previous work attained values equal to 91.2% for α and 1.5% for ε. The optimization with a weighted multiobjective GA considering the fitness as the sum of α and $1-\varepsilon$ led to values equal to 97.8% for α end 4.8% for ε.

The principal interest of exploiting GA in these applications is that each evaluation of the objective function is time consuming. Indeed, it can take up to 50 hours of CPU time for the first application and up to 30 hours for the second. Consequently, given the huge number of possibilities, these problems can not be approached by a systematic scan on parameters. The GA can however address this problem by evaluating a reduced number of possibilities in parallel. Moreover, the individuals evaluated during this exploration of parameters space contributes to a reservoir of information on the fitness. This database is used to avoid multiple evaluations of the same individuals and can be used to guide the exploration through additional heuristics.

More information on methodology and results can be found in the two following articles.

Genetic algorithms used for the optimization of light-emitting diodes and solar thermal collectors

Alexandre Mayer[*a], Annick Bay[a,b], Lucie Gaouyat[a], Delphine Nicolay[c], Timoteo Carletti[c], Olivier Deparis [a]
[a]Research Center in Physics of Matter and Radiation, University of Namur, Rue de Bruxelles 61, 5000 Namur, Belgium; [b]Scripps Institution of Oceanography, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA USA 92093; [c]Department of Mathematics and naXys, Namur Center for Complex Systems, University of Namur, Rempart de la Vierge 8, 5000 Namur, Belgium

ABSTRACT

We present a genetic algorithm (GA) we developed for the optimization of light-emitting diodes (LED) and solar thermal collectors. The surface of a LED can be covered by periodic structures whose geometrical and material parameters must be adjusted in order to maximize the extraction of light. The optimization of these parameters by the GA enabled us to get a light-extraction efficiency η of 11.0% from a GaN LED (for comparison, the flat material has a light-extraction efficiency η of only 3.7%). The solar thermal collector we considered consists of a waffle-shaped Al substrate with $NiCrO_x$ and SnO_2 conformal coatings. We must in this case maximize the solar absorption α while minimizing the thermal emissivity ϵ in the infrared. A multi-objective genetic algorithm has to be implemented in this case in order to determine optimal geometrical parameters. The parameters we obtained using the multi-objective GA enable α~97.8% and ϵ~4.8%, which improves results achieved previously when considering a flat substrate. These two applications demonstrate the interest of genetic algorithms for addressing complex problems in physics.

Keywords: genetic algorithm, optimization, light-emitting diode, solar thermal collector, GaN, Al, cermet

1. INTRODUCTION

Nature has developed its own algorithms for determining optimal solutions. With genetic algorithms (GA), we actually mimic natural selection in order to determine the optimal solutions of complex problems in physics. The idea to implement natural selection to problems that are not related to biology is due to Holland.[1] Other pioneering works on this subject were presented by De Jong,[2] Baker,[3] Goldberg,[4] Harik,[5] etc. While fundamental aspects of these evolutionary algorithms continue to be investigated,[6] their usefulness is proven by a growing number of applications.[7-9]

The idea of genetic algorithms consists in working with a population of individuals, each of them representing a given set of physical parameters and therefore a given value of the objective function we seek at optimizing. The initial population usually consists of individuals with random parameters. The best individuals are selected for the next generation. They are also allowed to breed in order to generate new individuals. Mutations are finally introduced as additional mean of exploration. When applied from generation to generation, this evolutionary strategy makes it possible to get closer and closer to the global optimum of a problem.

These general principles actually leave room for a variety of interpretations regarding the way a genetic algorithm should be implemented. The differences appear in the details when implementing the different steps of the algorithm. The coding of parameters, the definition of an effective fitness to work with when the objective function has several components and the strategy to use for the selection are a few examples. Every developer of a genetic algorithm will finally implement his own tricks to help the genetic algorithm converge more efficiently to the global optimum. For a given implementation of a genetic algorithm, one must take decisions regarding the size of the population, the rate of crossover and the rate of mutation. This is essentially done from experience.

[*] alexandre.mayer@unamur.be; http://perso.fundp.ac.be/~amayer

We present in this article a genetic algorithm we developed for the optimization of light-emitting diodes and solar thermal collectors. The genetic algorithm is presented with details in Sec. II. Sec. III presents the application of the GA to the optimization of a GaN light-emitting diode. Sec. IV presents the optimization of a solar thermal collector using a multi-objective version of the GA. We finally conclude this work in Sec. V.

2. DESCRIPTION OF THE GENETIC ALGORITHM

Let $f = f(\vec{x})$ be an objective function of n physical parameters x_i, where $x_i \in [x_i^{\min}, x_i^{\max}]$ with a specified granularity of Δx_i in the representation of each parameter. A solution in the mathematical sense will consist of a given set of physical parameters. We want to find, amongst this whole set of possibilities for the parameters x_i, the values that maximize globally the objective function f.

Each parameter x_i is represented by a string of n_i bits (0 or 1), also called a "gene". The corresponding value of x_i is given in most applications by

$$x_i = x_i^{\min} + \langle gene \rangle_i \frac{x_i^{\max} - x_i^{\min}}{2^{n_i} - 1} \tag{1}$$

where $\langle gene \rangle_i \in [0, 2^{n_i} - 1]$ stands for the value coded by the gene i in Gray binary coding.[10] The length n_i of each gene is chosen so that $(x_i^{\max} - x_i^{\min})/(2^{n_i} - 1) \leq \Delta x_i$.

If a strict enforcement of the granularity Δx_i is required, one can use instead of Eq. (1) the expression

$$x_i = x_i^{\min} + \langle gene \rangle_i \Delta x_i \tag{2}$$

where $\langle gene \rangle_i \in [0, 2^{n_i} - 1]$ as previously. The genetic algorithm must reject in this case gene values that lead to $x_i > x_i^{\max}$. The advantage of working with a strict representation of the parameters x_i is that the number of possibilities to explore is smaller. The genetic algorithm is therefore likely to converge more rapidly to the solution.

A given set of parameters $\{x_i\}_{i=1}^n$ is finally represented by the juxtaposition of the n genes used for the representation of each parameter. These strings of n genes are also called "DNA". The genetic algorithm actually works on the DNA representation of these parameters when searching for the optimal solution.

We work with a population of n_{pop}=100 individuals. Each individual has its own DNA. It is therefore representative of a given set of parameters. The initial population usually consists of random individuals. These individuals must be evaluated in order to determine their fitness. When the objective function f is a scalar function, the fitness is simply taken as the value of this function. When the objective function f has several components, we work with an effective fitness that depends on the different components f_j of this objective function and on the Pareto-classification of the individuals regarding these different components (see the Appendix). The evaluation of these individuals can be done in parallel on most recent computers since multi-core architectures are today the standard. This makes genetic algorithms especially suited to parallel-programming techniques.

The individuals are then sorted according to their fitness. We select $n_{pop}/2$ individuals ("the parents") by a rank-based "Roulette Wheel Selection". This is a random selection procedure in which the probability for an individual to be selected is proportional to its weight on a "wheel".[10] The individual with the highest fitness receives a weight of n_{pop}, the second-best individual receives a weight of n_{pop}-1, etc. The individual with the smallest fitness receives a weight of 1. A given individual can be selected several times. This enables the best individuals to progressively dominate the population.

The parents are transferred to the next generation. In addition, they determine new individuals ("the children"). For any pair of parents, two children are obtained either (i) by a one-point crossover of the parents' DNA (probability of 90%), or (ii) by a simple replication of the parents' DNA (probability of 10%). The point at which the two parts of the parents' DNA is exchanged is chosen randomly.[10] The transmission of unchanged individuals to the next generation enables the conservation of good solutions. The exploration of new solutions is achieved by the individuals obtained when crossing the parents' DNA. These individuals combine the features of individuals that passed the selection process. These

individuals will from time to time have a higher fitness than their parents, which makes the genetic algorithm progress in its search for optimal solutions.

We finally introduce random mutations on the children's DNA. Each bit of the children's DNA has a probability of 1% to be reversed. This is an essential ingredient for the exploration of parameters. When the rate of mutation is too small, the genetic algorithm may converge too quickly, without finding the global optimum. When the rate of mutation is too high, the exploration of parameters tends to be essentially random and therefore inefficient considering the number of possibilities to explore. The mutation rates to use are typically between 0.1% (mild value for easy convergence) and 5% (aggressive value). We use a value of 1% based on experience with previous problems.

These steps of selection, crossover and mutation must be repeated from generation to generation until convergence is achieved (maximum of 100 generations). By this game of natural selection, the genetic algorithm will progressively converge to the global optimum of the function *f*. We implemented elitism in order to make sure that the best individual is not lost when going from one generation to the next. We also replaced the bottom 10% of the population by random individuals. This incorporation of random individuals at each generation enables the introduction of seeds to the global optimum that may have been missing in the initial population. It also enables the genetic algorithm to consider useful directions in the exploration of parameters. Although this requires additional evaluations of the fitness, experience shows that convergence is actually improved when doing so. Since the evaluation of the fitness was especially time-consuming for the applications presented in this paper, we established a record with all individuals that were evaluated. This record was checked systematically by the genetic algorithm before each evaluation of the fitness in order to avoid unnecessary repetitions of these calculations.

3. OPTIMIZATION OF A LIGHT-EMITTING DIODE

The first application we consider aims at optimizing the light-extraction efficiency η of GaN light-emitting diodes (LED). This application is essentially an extension of the work presented by Bay et al. in Refs 11 and 12. It was demonstrated in this previous work that the light-extraction efficiency of existing GaN light-emitting diodes can be improved by considering a periodic texturation of the surface. This texturation consisted of the periodic repetition of structures made of photoresist with a "factory-roof" geometry.

The main wavelength λ of the GaN LED is 425 nm. The light-extraction efficiency is defined by

$$\eta = \int_0^{2\pi} \int_0^{\pi/2} T(\theta,\phi,\lambda) \sin\theta d\theta d\phi \qquad (3)$$

where $T(\theta,\varphi,\lambda)$ is the transmittance of the system for a radiation of wavelength λ coming from the GaN substrate with (θ,φ) directional angles. The transmittance was calculated by using a Rigorous Coupled-Waves Analysis within the transfer-matrix methodology.[13-14] The dielectric constant of GaN at the wavelength considered is 6.4.[15] The dielectric constant of the current-spreading layer (nickel and gold alloy) was calculated considering ε_{Ni}=-3.7+i 8.1 and ε_{Au}=-1.6+i 6.3.[16] The dielectric constant ε of the photoresist is 2.763 (manufacturer's value for photoresist AZ 9245®).[17]

The parameters that were considered in the work of Bay et al. are the period *P* and the height *H* of factory-roof structures made of photoresist. By scanning on *P* and *H* for values between 1 and 15 µm with a step of 1 µm, a maximum for the light-extraction efficiency η was found for *P*=5 µm and *H*=6 µm. The value of η reported in Ref. 12 is 5.7%, which corresponds to a relative increase of 55% compared to the value achieved without the photoresist (η= 3.7% for the flat GaN).

We extend this previous work by considering periodic structures with a more general shape. We also refine the exploration of parameters. The objective is to achieve higher light-extraction efficiencies by an optimal choice of parameters. The height $h(x)$ of the structures considered for the surface texturation of GaN is given the general expression

$$h(x)=\begin{cases} H\times\left[1-\dfrac{(c.P-x)^{\alpha_{left}}}{(c.P)^{\alpha_{left}}}\right] & when \quad 0\le x\le c.P \\ H\times\left[1-\dfrac{(x-c.P)^{\alpha_{right}}}{(P-c.P)^{\alpha_{right}}}\right] & when \quad c.P\le x\le P \end{cases} \tag{4}$$

where P and H refer as previously to the period and the height of these structures. The parameter $c\in[0,1]$ determines the position of the apex of these structures. For c=0, 0.5 and 1, the apex is respectively on the left, in the middle and on the right of the base period P. The coefficients α_{left} and α_{right} determine the concavity of the left and right edges. Straight edges are achieved when $\alpha_{left}=\alpha_{right}=1$. Values of α_{left} or α_{right} higher than 1 will result in concave edges that extend beyond the reference triangular shape achieved when $\alpha_{left}=\alpha_{right}=1$. Values of α_{left} or α_{right} smaller than 1 will result in convex edges that keep within the reference triangular shape achieved when $\alpha_{left}=\alpha_{right}=1$.

The "factory-roof" structures considered in the work of Bay et al.[12] correspond to c=1, α_{left}=1 and α_{right} =1. We extend this study by considering periodic structures with a period P between 1 and 15 µm (step ≤ 0.1 µm), a height H between 1 and 15 µm (step ≤ 0.1 µm), a relative center position c between 0.5 and 1 (step ≤ 0.01), and α_{left} and α_{right} coefficients between 0.2 and 5 (step ≤ 0.01). The results achieved by the genetic algorithm are summarized in Table 1.

Table 1. Parameters relevant to our model of surface texturation for a GaN LED and resulting light-extraction efficiencies. The four lines correspond to different optimizations of these parameters (the parameters optimized in each study are underlined).

P (µm)	H (µm)	c	α_{left}	α_{right}	ε	η	Source
5	6	1	1	1	2.763	5.7%	Ref. 12
6.98	4.97	0.508	1.074	1.055	2.763	7.1%	GA
3.20	2.13	1	1	1	6.325	7.3%	GA
3.42	2.63	0.603	1.205	0.933	6.340	11.0%	GA

The results presented in Table 1 show that it is possible using the genetic algorithm to obtain parameters that increase substantially the light-extraction efficiency η of GaN light-emitting diodes. The solution found by the GA when considering P, H, c, α_{left} and α_{right} as adjustable parameters enables a light-extraction efficiency η of 7.1%. This represents a relative increase of 92% compared to the η value of 3.7% achieved for a flat GaN. These results were achieved by giving the photoresist a dielectric constant ε of 2.763 (manufacturer's value for photoresist AZ 9245®).[17] We can extend this study and consider the dielectric constant ε as an additional adjustable parameter. This should guide future experimental work by suggesting materials to use for the surface texturation. We considered for this study ε values between 1.2 and 6.35 (step ≤ 0.01). The results achieved when considering P, H and ε as adjustable parameters are presented in the third line of Table 1. We achieve in this case a light-extraction efficiency of 7.3% (relative increase of 97% compared to the flat GaN). By considering finally P, H, c, α_{left}, α_{right} and ε as adjustable parameters, we achieved a light-extraction efficiency of 11.0% (fourth line of Table 1). This corresponds to a relative increase of 197% compared to the flat GaN. The structure associated with this last result is represented in Fig. 1. The solutions found by the GA, when ε is considered as an adjustable parameter, suggest that the material used for the surface texturation should have essentially the same dielectric constant as the GaN. It seems also that the optimal shapes are essentially symmetric with respect to the center of the period.

When searching for optimal values of P, H, c, α_{left}, α_{right} and ε, the number of bits required for the representation of these six parameters was 49 (we used the parameter representation given in Eq. 1 as a strict enforcement of the parameter granularity was not necessary). This corresponds to the length of a DNA. The number of possibilities to explore for the six parameters considered in this optimization was therefore 2^{49}= 562,949,953,421,312. The genetic algorithm managed to find the optimum given in the fourth line of Table 1 after only 1687 evaluations of the fitness.

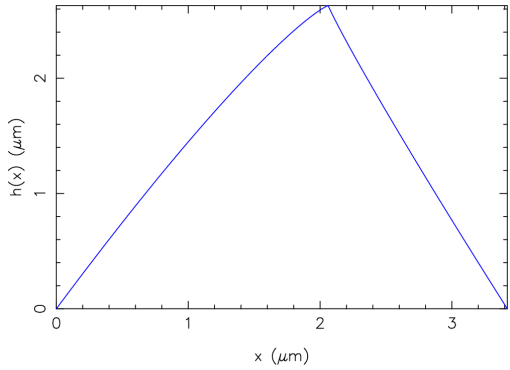


Figure 1. Representation of the structure achieved when searching for optimal values of *P*, *H*, *c*, α_{left}, α_{right} and ε with the objective of maximizing the light-extraction efficiency of GaN light-emitting diodes. The parameters associated with this representation are given in the fourth line of Table 1. The light-extraction efficiency achieved with this structure is 11.0%.

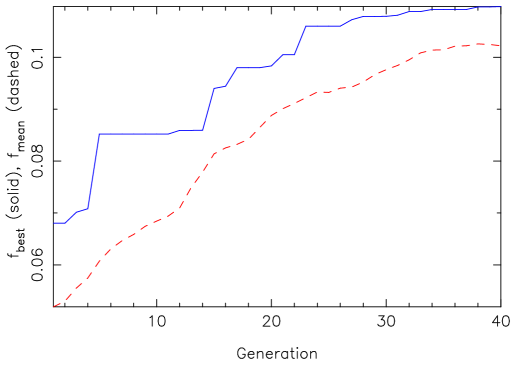


Figure 2. Best fitness (solid) and mean fitness (dashed) in the population when searching for optimal values of *P*, *H*, *c*, α_{left}, α_{right} and ε with the objective of maximizing the light-extraction efficiency of GaN light-emitting diodes.

4. OPTIMIZATION OF A SOLAR THERMAL COLLECTOR

The second application we consider deals with the optimization of a solar thermal collector. This application is essentially an extension of the work presented by Gaouyat et al. in Refs 18 and 19. In this previous work, an aluminium substrate with $NiCrO_x$ and anti-reflection (AR) coatings was studied with the objective of developing high-performance solar thermal collectors. The $NiCrO_x$ ceramic-metal composite (cermet) was chosen because of its high durability and attractive absorption/emission selectivity.[20] The applicability of this material to the development of solar thermal collectors was presented with details in Ref. 19.

In order for a solar thermal collector to be efficient, one must maximize the solar absorption α, while minimizing the thermal emissivity ε in the infrared.[18-19] α represents the fraction of the solar spectrum that is effectively absorbed by the system. ε represents the fraction of the spectrum of a blackbody heated at 373 K that will escape the system (equivalent of thermal losses). These quantities are defined by

$$\alpha = \int_0^\infty [1 - R(\lambda)] B_S(\lambda) d\lambda \Big/ \int_0^\infty B_S(\lambda) d\lambda \tag{5}$$

and

$$\varepsilon = \int_0^\infty [1 - R(\lambda)] B_a(\lambda) d\lambda \Big/ \int_0^\infty B_a(\lambda) d\lambda \tag{6}$$

where $B_S(\lambda)$ is the solar irradiance spectrum (Air Mass 1.5), $B_a(\lambda)$ is the irradiance spectrum of a blackbody heated at 373 K and $R(\lambda)$ is the reflectance of the system for a radiation of wavelength λ having a normal incidence.

Values of α=91.2% and ε=1.5% were achieved in previous work by considering a bi-layer stack of $NiCrO_x$/AR deposited on a flat Al substrate.[19] We seek at improving these results by keeping the same Al/$NiCrO_x$/AR configuration. The Al substrate will be shaped like a "waffle" (see Fig. 3). This idea was inspired by the work of Shimizu on structured W substrates for high-temperature solar absorbers.[21] We use finally SnO_2 as material for the anti-reflection coating. The geometrical parameters that characterize the Al substrate are hence the period *P*, the height *H* of the Al, the ratio *f* between the width of the holes on the front side of the Al and the period, and finally the ratio *r* between the width of the holes on the back side of the Al and that on the front side. Conformal coatings of $NiCrO_x$ (thickness t_1) and SnO_2 (thickness t_2) are then added to this waffle-shaped Al structure.

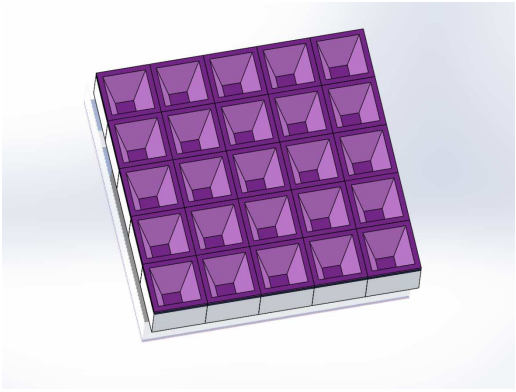


Figure 3. Waffle-shaped Al substrate with conformal coatings of $NiCrO_x$ and SnO_2. This structure is considered for the development of high-performance solar thermal collectors.

The optical properties of this waffle-shaped Al/$NiCrO_x$/SnO_2 system can be simulated by using again a Rigorous Coupled-Waves Analysis within the transfer-matrix methodology for the calculation of R(λ).[13-14] The optical properties of the different materials were taken from the literature and UV-visible and IR ellipsometric measurements.[18,22-23] The parameters considered for this optimization problem are *P*, *H*, *f*, *r*, t_1 and t_2. There are two objective functions to maximize: f_1=α and f_2=1-ε. We must therefore use a multi-objective genetic algorithm.

The general idea of multi-objective genetic algorithms follows the description of Sec. II. We work in this case with an effective fitness that depends on the two components f_1 and f_2 of the objective function and on the *Pareto-classification* of the population with respect to these two components. The definition of this effective fitness is given with details in the Appendix. The multi-objective genetic algorithm seeks in this case at establishing a set of *Pareto-optimal* solutions. These solutions provide (f_1, f_2) values with a distinct advantage compared to the rest of the population. Amongst this set of Pareto-optimal solutions, individuals that are better for f_1 will be weaker for f_2 (but no individual in the whole population is better for both f_1 and f_2). Making finally a choice amongst this set of solutions depends on how we value the two components f_1 and f_2 of the objective function and on their practicability for an experimental device. We implemented elitism and kept records for solutions that were better for either f_1, f_2 or f_1+f_2 (solutions that maximize f_1+f_2 are generally those of interest for a solar collector).

For the optimization of the solar collector, we considered P values between 500 and 1500 nm (step of 5 nm) and H values between 500 and 2500 nm (step of 5 nm). These boundaries enable P and H to keep in the same range as the absorbed wavelengths. We take f values between 0.5 and 0.99 (step of 0.01) and r values between 0 and 0.99 (step of 0.01) in order to explore the full range of pyramidal shapes. Finally t_1 and t_2 are chosen between 50 nm and 100 nm (step of 5 nm) to be representative of layer thicknesses obtained by physical vapor deposition (PVD). The representation of these parameters relied on Eq. 2 in order for H, t_1 and t_2 to be always multiples of the same unit (5 nm). It was indeed easier in this case to achieve good spatial discretization of the system. The number of bits required for the DNA representation of these parameters was 38 and the number of possibilities to explore was 48,763,605,000. The number of evaluations of the fitness was however as small as 2377 for the results presented here.

Solutions providing good values for f_1, f_2 or f_1+f_2 were quickly established by the GA so that a representation of the number of Pareto-optimal solutions is actually more illustrative for the progress achieved by the algorithm. Fig. 4 shows that the genetic algorithm progressively builds up an ensemble of Pareto-optimal solutions for the realization of a solar thermal collector. The solution that provides the maximal value of f_1+f_2 is given in the first line of Table 2. The next three lines provide selected Pareto-optimal solutions established by the GA.

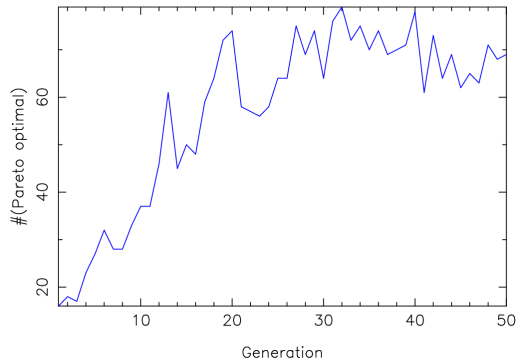


Figure 4. Number of Pareto-optimal solutions when searching for P, H, f, r, t_1 and t_2 with the objective of optimizing the parameters α and ε of a solar thermal collector.

Table 2. Parameters relevant to the optimization of the parameters α and ε of a solar thermal collector. The first line corresponds to the solution that maximizes f_1+f_2, where f_1= α and f_2=1- ε. The next three lines correspond to selected Pareto-optimal solutions established by the GA.

P (nm)	H (nm)	f	r	t_1 (nm)	t_2 (nm)	α	ε	
1345	1960	0.96	0.45	50	50	97.8%	4.8%	f_1+f_2 max
1435	1975	0.99	0.31	55	50	98.4%	5.8%	P-optimal
795	1590	0.90	0.28	50	50	96.1%	4.1%	P-optimal
560	545	0.95	0.28	50	50	95.2%	3.7%	P-optimal

The results presented in Table 2 compare very well with the values of α=91.2% and ε=1.5% achieved in previous work with a flat Al/$NiCrO_x$/AR configuration[19] and with the record values of α=97% and ε=5% obtained on a 3-layers stack.[23] The three Pareto-optimal solutions given in Table 2 illustrate the fact that solutions with better α values have less attractive ε values. These solutions have t_1 and t_2 values around 50 nm. The choice of a particular solution for the

realization of a solar thermal collector will depend at this point on how we want to compromise between α and ε and on other practical considerations.

5. CONCLUSION

These applications prove that genetic algorithms constitute a smart approach to global optimization problems. Genetic algorithms involve indeed a collective exploration of the parameter space. This gives the algorithm the capacity to escape local optima. The algorithm is also more robust against the possible failure of an individual to evaluate the fitness and it can easily account for constrains in the parameter space. The population contributes in this exploration of parameters to a reservoir of information on the fitness. This database may be used to guide the exploration through additional heuristics. Genetic algorithms are finally especially suited to parallel programming techniques. High efficiency on super-calculators was indeed achieved by using a multi-agent programming model in order to parallelize the evaluation of the fitness. The applications presented in this work aimed at optimizing parameters that influence the efficiency of GaN light-emitting diodes and solar thermal collectors. The solutions found by the GA turned out to improve the efficiencies achieved in previous work and to be competitive with record values found in the literature. The multi-objective version of the genetic algorithm actually provides a whole set of solutions with distinct advantages. Making a choice amongst this set of solutions depends on how we value the different components of objective function and on practical considerations. Addressing these optimization problems by scanning on the different parameters considered would have been intractable because the number of possibilities to consider grows exponentially with the number of parameters (each evaluation of the fitness required up to 50 hours of CPU time for the optimization of the LED and up to 30 hours for the optimization of the solar collector). The genetic algorithm could however address these optimization problems by evaluating in parallel only a reduced number of possibilities. These results prove that genetic algorithms are a great value for addressing complex problems in physics.

ACKNOWLEDGMENTS

A.M. is funded as Research Associate by the Fund for Scientific Research (F.R.S.-FNRS) of Belgium. L.G. is supported by FNRS-FRIA. D.N. and T.C. acknowledge the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme and initiated by the Belgian Science Policy Office. This research used resources of the "Plateforme Technologique de Calcul Intensif (PTCI)" (http://www.ptci.unamur.be) located at the University of Namur, Belgium, which is supported by the F.R.S.-FNRS under the convention No. 2.4520.11. The PTCI is member of the "Consortium des Equipements de Calcul Intensif (CECI)" (http://www.ceci-hpc.be). Erwin Volon is acknowledged for his Solid Works representations of the Al/NiCrO$_x$/SnO$_2$ structure. Frédéric Wauthelet is finally acknowledged for the script that enabled the development of multi-agent versions of the genetic algorithm.

APPENDIX: EFFECTIVE FITNESS FOR A MULTI-OBJECTIVE GENETIC ALGORITHM

We define here the effective fitness that was used by the multi-objective genetic algorithm in Sec. IV. This effective fitness is used to establish a classification of the population. This is indeed required in the selection step of the algorithm. The effective fitness presented here is that used by Nicolay in previous work.[24] The idea is due originally to Deb.[25]

Let us consider a population of n_{pop} individuals. We refer by n to the number of parameters x_i and by f_j to the different components of the objective function.

A solution $\vec{x}_1$ is dominated by the solution $\vec{x}_2$ if $f_j(\vec{x}_2) \geq f_j(\vec{x}_1) \forall j$ and $\exists j : f_j(\vec{x}_2) > f_j(\vec{x}_1)$.

Pareto-optimal solutions are solutions that are not dominated. These solutions will receive a rank of 1. Solutions will receive a rank of 2 if they are not dominated when discarding solutions of rank 1. Solutions of rank 3 are solutions that are not dominated if we discard solutions of rank 1 and 2. We can proceed in this way and attribute a rank to every individual in the population.

The effective fitness will be higher for individuals that have lower ranks. The genetic algorithm will tend in this way to develop solutions that are Pareto-optimal instead of solutions that improve a specific combination of the components f_j of the objective function. We seek also at establishing a set of Pareto-optimal solutions that presents a good dispersion. This prevents indeed early convergence of the GA to a given individual. We proceed therefore in the following way to define the effective fitness.

All individuals of rank 1 receive an effective fitness of n_{pop}. We then define a sharing function in order to reduce, amongst individuals of the same rank, the effective fitness of individuals that are too close. For individuals of the same rank, we define a distance matrix whose components $d_{k,l}$ are defined by

$$d_{k,l} = \sqrt{\sum_{i=1}^{n} \left(\frac{x_i[k] - x_i[l]}{x_i^{\max} - x_i^{\min}} \right)^2} \tag{7}$$

where x_i[k] refers to the parameter x_i of an individual k, $x_i^{\max} = \max_{k \in [1, n_{pop}]} x_i[k]$ and $x_i^{\min} = \min_{k \in [1, n_{pop}]} x_i[k]$.

The sharing function between two individuals is then defined by $S_{k,l} = 1 - (d_{k,l} / \sigma_{share})^2$ if $d_{k,l} \leq \sigma_{share}$ and 0 otherwise. Following Ref. 24-25, we take $\sigma_{share} = 0.5 / \sqrt[n]{10}$. We then define the niche count of a given individual by $m_k = \sum_l S_{k,l}$ where the sum is restricted to individuals of the same rank. The effective fitness of each individual is then divided by its niche count. We penalize in this way individuals that are too close to other individuals within the same rank.

The effective fitness of all individuals of rank 2 is then initialized with a value of $0.99 \times f_{effect,min}$[rank 1], where $f_{effect,min}$[rank 1] refers to the minimal value of the effective fitness for the individuals of rank 1. We proceed then by computing the distance matrix $d_{k,l}$, the sharing function $S_{k,l}$ and finally the niche count m_k for all individuals of rank 2. The effective fitness of these individuals is then divided by their niche count.

We continue in this way until all individuals in the population have been attributed an effective fitness. Other definitions are possible for this effective fitness. They may be considered in future work.

REFERENCES

[1] Holland, J. H., [Adaptation in Natural and Artificial Systems], University of Michigan Press, Ann Arbor, 1-183 (1975).

[2] De Jong, K. A., [An analysis of the behaviors of genetic adaptative systems], PhD thesis, University of Michigan, Ann Arbor, 1-256 (1975).

[3] Baker J. E., "Adaptative selection methods for genetic algorithms," Proceedings of the 1st International Conference on Genetic Algorithms and their Applications, 101-111 (1985).

[4] Goldberg, D. E., "Sizing populations for serial and parallel genetic algorithms," Proceedings of the Third International Conference on Genetic Algorithms, 70-79 (1989).

[5] Harik, G., Cantu-Paz, E., Goldberg, D. E. and Miller, B. L., "The gambler's ruin problem, genetic algorithms, and the sizing of populations," Evol. Comput. 7(3), 231-253 (1999).

[6] Ma, Z. H., "Chaotic populations in genetic algorithms," Appl. Soft Comput. 12, 2409-2424 (2012).

[7] Goldberg, D. E., [Genetic Algorithms in Search, Optimization and Machine Learning], Addison-Weley, Reading, 1-432 (1989).

[8] Preblea, S., Lipson, M. and Lipson, H., "Two-dimensional photonic crystals designed by evolutionary algorithms," Appl. Phys. Lett. 86(6), 061111 (2005).

[9] Lin, A. and Philips, J., "Optimization of random diffraction gratings in thin-film solar cells using genetic algorithms," Sol. Energ. Mat. Sol. C. 92, 1689-1696 (2008).

[10] Judson, R., "Genetic Algorithms and their use in Chemistry," Reviews in Computational Chemistry 10, 1-73 (1997).

[11] Bay, A., Sarrazin, M., and Vigneron, J.-P., "Search for an optimal light-extracting surface derived from the morphology of a firefly lantern," Opt. Eng. 52(2), 028001-1-028001-7 (2013).
[12] Bay, A., André, N., Sarrazin, M., Belarouci, A., Aimez, V., Francis, L. A. and Vigneron, J.-P., "Optimal overlayer inspired by Photuris firefly improves light-extraction efficiency of existing light-emitting diodes," Optics Express 21 (S1), A179-A189 (2013).
[13] Vigneron, J.-P., Forati, F., André, D., Castiaux, A., Derycke, I. and Dereux, A., "Theory of electromagnetic energy transfer in three-dimensional structures," Ultramicroscopy 61, 21-27 (1995).
[14] Vigneron, J.-P. and Lousse, V., "Variation of a photonic crystal color with the miller indices of the exposed surface," Proc. SPIE 6128, 61281G (2006).
[15] Eliseev, P., Smolyakov, G. and Osinski, M., "Ghost modes and resonant effects in AlGaN-InGaN-GaN lasers," IEEE J. Sel. Top. Quantum Electron. 5, 771-779 (1999).
[16] Lynch, D., Hunter, W. and Palik, E., [Handbook of Optical Constants of Solids II], Academic Press, Inc., 1991.
[17] "Microchemicals, optical parameters of photoresists," (2010). www.microchemicals.eu/technical_information.
[18] Gaouyat, L., Mirabella, F. and Deparis, O., "Critical tuning of magnetron sputtering process parameters for optimized solar selective absorption of $NiCrO_x$ cermet coatings on aluminium substrate," Appl. Surf. Sci. 271, 113-117 (2013).
[19] Gaouyat, L., He, Z., Colomer, J.-F., Lambin, Ph., Mirabella, F., Schryvers, D. and Deparis, O., "Revealing the innermost nanostructure of sputtered $NiCrO_x$ solar absorber cermets," Sol. Energ. Mat. Sol. C. 122, 303-308 (2014).
[20] Kennedy, C. E., "Review of Mid- to High-Temperature Solar Selective Absorber Materials," NREL/TP-520-31267, National Renewable Energy Laboratory, Colorado, July 2002.
[21] Shimizu, M., Takeuchi, K., Sai, H., Iguchi, F., Sata, N. and Yugami, H., "High-temperature solar selective absorber material using surface microcavity structures," Proc. ASME 5, 783-787 (2011).
[22] Stenzel, O., [The Physics of Thin Film Optical Spectra: An Introduction], Springer, Heidelberg (2005).
[23] Zhao, S. and Wäckelgård, E., "Optimization of solar absorbing three-layer coatings," Sol. Energ. Mat. Sol. C. 90(3), 243-261 (2006).
[24] Nicolay, D., [Modélisation et apprentissage des réseaux de neurones artificiels], Master thesis, University of Namur, Namur, 48-57 (2012).
[25] Deb, K., "Multi-objective evolutionary algorithms: Introducing bias among Pareto-optimal solutions," [Advances in Evolutionary Computing], Springer-Verlag, Berlin, 263-292 (2003).

A multi-objective genetic algorithm for the optimization of a flat-plate solar thermal collector

Alexandre Mayer,[1,*] Lucie Gaouyat,[1] Delphine Nicolay,[2] Timoteo Carletti[2] and Olivier Deparis[1]

[1] *Laboratoire de Physique du Solide, University of Namur, Rue de Bruxelles 61, 5000 Namur, Belgium*

[2] *Department of Mathematics, University of Namur, Rempart de la Vierge 8, 5000 Namur, Belgium*

[*] *alexandre.mayer@unamur.be*

Abstract: We present a multi-objective genetic algorithm we developed for the optimization of a flat-plate solar thermal collector. This collector consists of a waffle-shaped Al substrate with $NiCrO_x$ cermet and SnO_2 anti-reflection conformal coatings. Optimal geometrical parameters are determined in order to (i) maximize the solar absorptance α and (ii) minimize the thermal emittance ε. The multi-objective genetic algorithm eventually provides a whole set of Pareto-optimal solutions for the optimization of α and ε, which turn out to be competitive with record values found in the literature. In particular, a solution that enables $\alpha = 97.8\%$ and $\varepsilon = 4.8\%$ was found.

OCIS codes: (350.6050) Solar energy; (310.6845) Thin film devices and applications; (350.4600) Optical engineering; (000.3860) Mathematical methods in physics.

References and links

1. E. Rephaeli and S. Fan, "Absorber and emitter for solar thermophotovoltaic systems to achieve efficiency exceeding the Shockley-Queisser limit," Opt. Express **17**(17), 15145–15159 (2009).
2. N.P. Sergeant, O. Pincon, M. Agrawal, and P. Peumans, "Design of wide-angle solar-selective absorbers using aperiodic metal-dielectric stacks," Opt. Express **17**(25), 22800–22812 (2009).
3. H.L. Zhang, J. Baeyens, J. Degrève, and G. Cacères, "Concentrated solar power plants: Review and design methodology," Renewable and Sustainable Energy Reviews **22**, 466-481 (2013).
4. E. Wäckelgård, G.A. Niklasson, and C.G. Granqvist, "Selectively solar-absorbing coatings," in *Solar Energy: The state of the art*, J. Gordon, ed. (Science, 2001), chap. 3.
5. X.-F. Li, Y.-R. Chen, J. Miao, P. Zhou, Y.-X. Zheng, L.-Y. Chen, and Y.-P. Lee, "High solar absorption of a multilayered thin film structure," Opt. Express **15**(4), 1907–1912 (2007).
6. F. Cao, K. McEnaney, G. Chen, and Z. Ren, "A review of cermet-based spectrally selective solar absorbers," Energy Environ. Sci. **7**, 1615–1627 (2014).
7. C.E. Kennedy, "Review of Mid- to High-Temperature Solar Selective Absorber Materials," NREL/TP-520-31267, National Renewable Energy Laboratory, Colorado, July 2002.
8. R.A. Buhrman, "Physics of solar selective surfaces," in *Physics of solar selective surfaces*, K.W. Böer, ed. (Springer, 1986, vol. 3), chap. 4.
9. M. Langlais, J.-P. Hugonin, M. Besbes, and Ph. Ben-Abdallah, "Cooperative electromagnetic interactions between nanoparticles for solar energy harvesting," Opt. Express **22**(S3), A577–A588 (2014).
10. T. Boström, E. Wäckelgård, and G. Westin, "Solution chemical route to nickel-alumina coatings for thermal solar absorbers," Sol. Energy **74**, 497–503 (2003).
11. V. Teixeira, E. Sousa, M.F. Costa, C. Nunes, L. Rosa, and M.J. Carvalho, "Spectrally selective composite coatings of Cr-Cr_2O_3 and Mo-Al_2O_3 for solar energy applications," Thin Solid Films **392**, 320–326 (2001).

12. S. Zhao and E. Wäckelgård, "The optical properties of sputtered composite of Al-AlN," Sol. Energ. Mat. Sol. C. **90**(13), 1861–1874 (2006).
13. S. Zhao and E. Wäckelgård, "Optimization of solar absorbing three-layer coatings," Sol. Energ. Mat. Sol. C. **90**(3), 243–261 (2006).
14. L. Gaouyat, F. Mirabella, and O. Deparis, "Critical tuning of magnetron sputtering process parameters for optimized solar selective absorption of NiCrOx cermet coatings on aluminium substrate," Appl. Surf. Sci. **271**, 113–117 (2013).
15. L. Gaouyat, Z. He, J.-F. Colomer, Ph. Lambin, F. Mirabella, D. Schryvers, and O. Deparis, "Revealing the innermost nanostructure of sputtered NiCrOx solar absorber cermets," Sol. Energ. Mat. Sol. C. **122**, 303–308 (2014).
16. D. Chester, P. Bermel, J.D. Joannopoulos, M. Soljacic, I. Celanovic, "Design and global optimization of high-efficiency solar thermal systems with tungsten cermets," Opt. Express **19**(53), A245–A257 (2011).
17. A. Lasagni, M. Nejati, R. Clasen, and F. Mücklich, "Periodic surface structuring of metals by laser interference metallurgy as a new fabrication method of textured solar selective absorbers," Adv. Eng. Mater. **8**(6), 580–584 (2006).
18. J.J. Cuomo, J.F. Ziegler, and J.M. Woodall, "A new concept for solar energy thermal conversion," Appl. Phys. Lett. **26**(10), 557–559 (1975).
19. M. Shimizu, K. Takeuchi, H. Sai, F. Iguchi, N. Sata, and H. Yugami, "High-temperature solar selective absorber material using surface microcavity structures," Proc. ASME **5**, 783–787 (2011).
20. J.H. Holland, *Adaptation in Natural and Artificial Systems* (University of Michigan, 1975).
21. K.A. De Jong, *An analysis of the behaviors of genetic adaptative systems*, PhD thesis (University of Michigan, 1975).
22. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Weley, 1989).
23. R. Judson, "Genetic Algorithms and their use in Chemistry," in *Reviews in Computational Chemistry*, K.B. Lipkowitz and D.B. Boyd, ed. (VCH, 1997, vol. 10), chap. 1.
24. R.L. Haupt and D.H. Werner, *Genetic Algorithms in Electromagnetics* (Wiley&Sons, 2007).
25. K. Deb, "Multi-objective evolutionary algorithms: Introducing bias among Pareto-optimal solutions," in *Advances in Evolutionary Computing* (Springer-Verlag, 2003), pp. 263–292.
26. D. Nicolay, *Modélisation et apprentissage des réseaux de neurones artificiels*, Master thesis (University of Namur, 2012).
27. J.-P. Vigneron, F. Forati, D. André, A. Castiaux, I. Derycke, and A. Dereux, "Theory of electromagnetic energy transfer in three-dimensional structures," Ultramicroscopy **61**, 21–27 (1995).
28. J.-P. Vigneron and V. Lousse, "Variation of a photonic crystal color with the miller indices of the exposed surface," Proc. SPIE 6128, 61281G (2006).
29. O. Stenzel, *The Physics of Thin Film Optical Spectra: An Introduction* (Springer, 2005).

1. Introduction

Flate-plate solar thermal collectors, amongst other solar-energy conversion devices like thermophotovoltaic (TPV) [1,2] or concentrating solar power (CSP) systems [3], are a nice example of the use of a renewable energy. Without the need of additional electric energy consumption they allow to heat water for domestic use (temperature lower than 100°C) [4–6]. Amongst all possibilities for producing solar absorbers for low to mid-temperature (100°C-300°C) applications (e.g., intrinsic absorbers, multilayers or other systems based on cooperative effects) [7], cermets, in the form of thin films, are today the only industrial alternative. Cermets are nanostructured composites in which metallic nanoparticles are embedded in a ceramic matrix. This structure is especially adapted for strong absorption in the UV-visible region, due to plasmonic absorption in the particles, coupling between the particles, and interband electronic transitions in the matrix [8, 9]. As the cermet coating is deposited on an IR-reflective substrate, its IR-transparency actually allows the solar collector to have a low emittance, therefore reducing thermal losses. Many cermet materials such as Ni-Al_2O_3, Cr-Cr_2O_3, Al-AlN and Ni-NiO are known to be good candidates [10–13]. Thanks to a previous work by Gaouyat *et al.* [14, 15], Ni-$NiCrO_x$ was found to be an ideal candidate for solar absorber applications because of its various absorption mechanisms. In order to reach higher performances, cermets are always coupled with an anti-reflection layer in a tandem absorber system [16]. Tin oxide was chosen for its ability to be produced by sputtering in addition to its anti-reflective property. It was proven in a previous work that a structuration of the substrate can lead to a further increase of

the absorption [17]. Indeed the use of a structured substrate, in synergy with a tandem bilayer, could relax the constraints of fine tuning of layer deposition parameters [14]. Therefore the effect of the structure has to be explored. Amongst all the patterns already studied, such as 1-D sinusoidal corrugation [17], lamellar profiles [17], pyramids [1] or dendrites [18], we decided to exploit the idea of Shimizu *et al.* [19] and we considered waffle-like patterns consisting of the periodic repetition of truncated inverted pyramids. The confrontation of tandem solar absorbers with and without substrate structuration will be considered in this study in order to understand the role of this additional feature. The focus of this work is on the optimization of flat-plate solar thermal collectors for domestic heating of water.

The optothermal properties of solar absorbers are characterized by two quantities: the solar absorptance α and the thermal emittance ε. They describe respectively the absorber ability to harvest the sun radiation and to avoid thermal losses calculated from the black-body spectrum of the absorber. In order to maximize the efficiency of the collector, the solar absorptance α should be maximized and the thermal emittance ε should be minimized. We hence need to conjointly optimize α and ε in order to achieve an efficient collector. This optimization is often done empirically since a complete investigation of all possible combinations of parameters would be untractable. The usual parameters are the thicknesses of the layers or the metal content of cermet layers [16]. In the present work, the parameters to be optimized are not only the thicknesses of the tandem absorber layers ($NiCrO_x$ and SnO_2), but also the geometrical parameters of the waffle-like structure.

Nature has developed its own algorithms for determining optimal solutions. With genetic algorithms (GA), we actually mimic natural selection in order to determine the optimal parameters of complex problems in physics [20–22]. The idea consists in working with a population of individuals, each of them representing a given set of physical parameters. The initial population usually consists of random individuals. The best individuals are then selected. They generate new individuals for the next generation. Random mutations in the coding of parameters are finally introduced. When applied from generation to generation, this evolutionary strategy makes it possible to determine the global optimum of a problem. These general principles actually leave room for a variety of interpretations regarding the way a genetic algorithm should be implemented [23, 24]. There are indeed different ways to assign a fitness to each individual, different strategies for the selection, different methods for the crossing and mutation of parameters. Every developer of a genetic algorithm will finally implement his own tricks to converge more efficiently to the solution. For a given implementation of a genetic algorithm, a decision must be taken for the size of the population, the rate of crossover and the rate of mutation. This is essentially done from experience.

We present in this work a multi-objective genetic algorithm we developed for the optimization of a solar thermal collector that consists of a waffle-shaped Al semi-infinite substrate with $NiCrO_x$ and SnO_2 conformal coatings. The geometrical parameters of this system must be adjusted in order to achieve two objectives: (i) to maximize the absorptance α and (ii) to minimize the emittance ε. The details of this algorithm are presented in Sec. II. Sec. III then presents the results achieved with the solar thermal collector. Sec. IV finally concludes this work.

2. Multi-objective genetic algorithm

Let $\vec{f} = \vec{f}(\vec{x})$ be an objective function of m components $f_1(\vec{x}), \ldots, f_m(\vec{x})$. Each component $f_j(\vec{x})$ depends on n physical parameters x_i, where $x_i \in [x_i^{\min}, x_i^{\max}]$ with a specified granularity of Δx_i in the representation of each parameter. We want to find, amongst this whole set of possibilities for the parameters x_i, the values that maximize globally the different components of the objective function.

Each parameter x_i is actually represented by a string of n_i bits (0 or 1), also called a "gene".

n_i is chosen so that $(x_i^{\max} - x_i^{\min})/(2^{n_i} - 1) \leq \Delta x_i$. The value of the physical parameter x_i is then given by $x_i = x_i^{\min} + \langle \text{gene } i \rangle \times \Delta x_i$, where $\langle \text{gene } i \rangle \in [0, 2^{n_i} - 1]$ stands for the value coded by the gene i in Gray binary coding [23]. The genetic algorithm must reject gene values that lead to $x_i > x_i^{\max}$ in order to achieve a strict enforcement of our parameter specifications. A given set of parameters $\{x_i\}_{i=1}^n$ is finally represented by the juxtaposition of the n genes used for the representation of each parameter. These strings of n genes are also called "DNA". The genetic algorithm actually works on the DNA representation of parameters when searching for optimal solutions.

We work with a population of n_{pop}=100 individuals. Each individual has its own DNA. It is therefore representative of a given set of parameters $\{x_i\}_{i=1}^n$. The initial population usually consists of random individuals. These individuals must be evaluated in order to determine the corresponding values of the objective function $\vec{f}$. We must also define an effective fitness f_{eff} for the classification of these individuals. Working with $f_{\text{eff}} = \sum_{j=1}^m w_j f_j$, where w_j are arbitrary weighting factors, would lead the GA to optimize a specific linear combination of the components f_j of the objective function, without taking into account how individuals actually compare for each f_j. We will work instead with an effective fitness f_{eff} that depends on the *Pareto-classification* of these individuals [24–26]. This classification is based on the concept of dominance: a solution $\vec{x}_1$ is dominated by the solution $\vec{x}_2$ if $f_j(\vec{x}_2) \geq f_j(\vec{x}_1) \forall j$ and $\exists j : f_j(\vec{x}_2) > f_j(\vec{x}_1)$. *Pareto-optimal* solutions are solutions that are not dominated. The effective fitness f_{eff}, which is given with details in the Appendix, will be higher for individuals that are not dominated. This will force the GA to establish a whole set of Pareto-optimal solutions, instead of just focussing on a specific linear combination of the f_j.

The individuals are then sorted according to this effective fitness. $n_{\text{pop}}/2$ individuals ("the parents") are selected by a rank-based "Roulette Wheel Selection" [23, 24]. This is a random selection procedure in which the probability for an individual to be selected is proportional to its weight on a "wheel". The individual with the highest effective fitness receives a weight equal to n_{pop}, the second-best individual receives a weight equal to $n_{\text{pop}} - 1$, etc. The last individual receives a weight equal to 1. Individuals with a higher effective fitness have thus more chance to be selected. A given individual can be selected several times. This enables the best individuals to progressively dominate the population.

The parents are transferred to the next generation. In addition, they generate new individuals ("the children"). For any pair of parents, two children are obtained either (i) by a one-point crossover of the parents' DNA (probability of 90%), or (ii) by a simple replication of the parents' DNA (probability of 10%). The position in the chain of bits at which the two parts of the parents' DNA is exchanged is chosen randomly [23, 24]. The transmission of unchanged individuals to the next generation enables the conservation of good solutions. The exploration of new solutions is achieved by the individuals obtained when crossing the parents' DNA. We finally introduce random mutations: each bit of the children's DNA has a probability of 1% to be reversed. This is an essential ingredient for the exploration of parameters. It enables indeed a final refinement of the parameters.

These steps of selection, crossover and mutation must be repeated from generation to generation until convergence is achieved (maximum of 100 generations). By this game of natural selection, the genetic algorithm will progressively determine optimal solutions for the problem considered. We implemented elitism in order to make sure that the individuals that provide the best values for $f_1, \ldots, f_m$ and $\sum_{j=1}^m f_j$ are not lost when going from one generation to the next. We also replaced the bottom 10% of the population by random individuals. This enables the introduction of seeds to optimal solutions that may have been missing in the initial population.

3. Optimization of a solar thermal collector

We can apply now the multi-objective genetic algorithm to the optimization of a solar thermal collector. In a previous work by Gaouyat *et al.* [14, 15], a flat aluminium substrate with $NiCrO_x$ and anti-reflection (AR) coatings was studied with the objective of developing high-performance solar thermal collectors. The $NiCrO_x$ ceramic-metal (cermet) composite was chosen because of its high durability and attractive absorption/emission selectivity [7]. It was shown that $NiCrO_x$ is an ideal candidate for the development of efficient solar thermal collectors.

In order to build an efficient solar thermal collector, we need to (i) maximize the solar absorptance α and (ii) minimize the thermal emittance ε [14, 15]. These quantities are defined by $\alpha = \int_0^\infty [1-R(\lambda)]B_S(\lambda)d\lambda / \int_0^\infty B_S(\lambda)d\lambda$ and $\varepsilon = \int_0^\infty [1-R(\lambda)]B_a(\lambda)d\lambda / \int_0^\infty B_a(\lambda)d\lambda$, where $B_S(\lambda)$ is the solar irradiance spectrum (Air Mass 1.5), $B_a(\lambda)$ is the black-body spectrum of the absorber and $R(\lambda)$ is the total (hemispherical) reflectance of the system for a radiation of wavelength λ at normal incidence. Since we assumed semi-infinite Al substrate, the transmittance vanishes and therefore the absorption is equal to $A(\lambda) = 1 - R(\lambda)$. α represents the fraction of the solar irradiance spectrum (B_S) that is effectively absorbed by the system. ε represents the fraction of the absorber black-body spectrum (B_a) that will escape the system (equivalent of thermal losses). In a Rigorous Coupled-Wave Analysis (RCWA), the calculated reflectance is the sum of the specular reflection component and all diffracted-order components. Since $R(\lambda)$ is calculated over the whole hemisphere, it fully accounts for scattering of the periodic structure in a 2π solid angle. Our calculation therefore takes into account both energy harvesting from the sun (α) and heat emission (ε) in all directions. Regarding the choice of the absorber operating temperature, since water heating for domestic use is considered here, the emittance ε was calculated at a fixed temperature (373 K), which is the maximal operating temperature of the device.

Values of $\alpha = 91.2\%$ and $\varepsilon = 1.5\%$ were achieved in a previous work by considering a bi-layer stack of $NiCrO_x$/AR deposited on a flat Al substrate [15]. We seek at improving this result by considering a waffle-shaped structuration of the substrate (see Fig. 1). We take SnO_2 as material for the anti-reflection coating. The geometrical parameters that characterize the corrugated surface of the semi-infinite Al substrate are the period P, the height H of the holes, the ratio f between the width L of the holes on the upper side and the period ($f = L/P$), and finally the ratio r between the width l of the holes on the bottom side and the width L of the holes on the upper side ($r = l/L$). Conformal coatings of $NiCrO_x$ (thickness t_1) and SnO_2 (thickness t_2) are then added to this structure.

The optical properties of this waffle-shaped Al/$NiCrO_x$/SnO_2 system were simulated by the RCWA method for the calculation of the total hemispherical reflectance $R(\lambda)$ [27,28]. Unpolarized light at normal incidence was assumed. We used 19×19 plane waves (diffraction orders) in the RCWA calculations. The optical properties of the different materials were taken from the literature and UV-visible and IR ellipsometric measurements [13, 14, 29]. We then used the multi-objective genetic algorithm to determine optimal geometrical parameters. The objective function had two components: $f_1 = \alpha$ and $f_2 = 1 - \varepsilon$ (α must be maximized; ε must be minimized). There were six parameters to determine: P, H, f, r, t_1 and t_2. We considered P values between 500 and 1500 nm (step of 5 nm) and H values between 500 and 2500 nm (step of 5 nm). These boundaries left P and H in the same range as the incident wavelengths. We took f between 0.5 and 0.99 (step of 0.01) and r between 0 and 0.99 (step of 0.01) in order to explore the full range of inverted pyramidal shapes. We took finally t_1 and t_2 between 50 nm and 100 nm (step of 5 nm) in order to be representative of layer thicknesses obtained by physical vapor deposition (PVD). These parameter specifications left us with 48,763,605,000 possibilities to explore. Only 2377 evaluations of the fitness were however required by the GA.

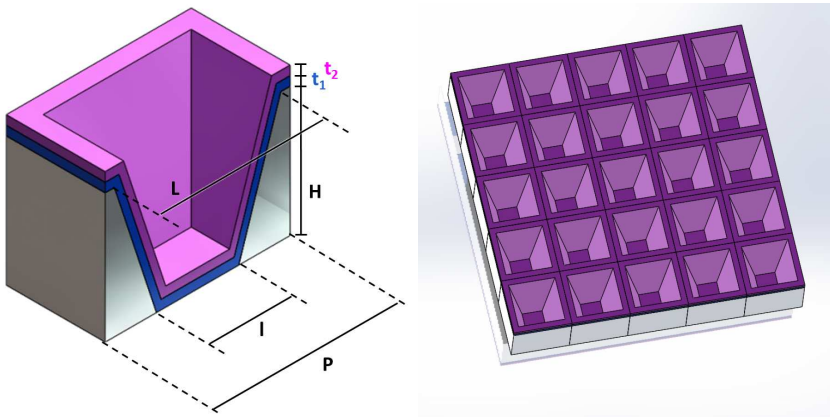


Fig. 1. Waffle-shaped Al structure with $NiCrO_x$ and SnO_2 conformal coatings. This corrugated structure sits on a semi-infinite Al substrate. This structure is considered for the development of high-performance flat-plate solar thermal collectors.

Figure 2 shows that the genetic algorithm progressively established a whole set of *Pareto-optimal* solutions. The number of these solutions increases indeed progressively to 70 on average after 30 generations. These solutions all provide (f_1, f_2) values with a distinct advantage compared to the rest of the population. No individual in the whole population provides indeed better values for both f_1 and f_2. Amongst this set of Pareto-optimal solutions, individuals that are better for f_1 are necessarily weaker for f_2. This is illustrated in Table 1, where a selection of Pareto-optimal solutions is provided. Table 1 also provides the solution that maximizes $f_1 + f_2$.

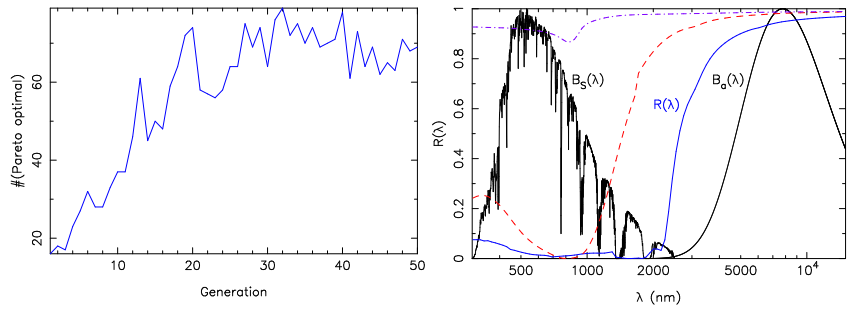


Fig. 2. Left: number of Pareto-optimal solutions when searching for P, H, f, r, t_1 and t_2 with the objective of optimizing the parameters α and ε of a solar thermal collector; Right: reflectance spectrum of the waffle-shaped $Al/NiCrO_x/SnO_2$ structure that provides $\alpha = 97.8\%$ and $\varepsilon = 4.8\%$ (solid), a flat $Al/NiCrO_x/SnO_2$ structure with $t_1 = t_2 = 50$ nm (dashed) and a flat uncoated Al (dot-dashed). The figure includes the normalized solar irradiance spectrum $B_S(\lambda)$ and the normalized black-body spectrum $B_a(\lambda)$ of the absorber at 373 K.

These results compare very well with the values of $\alpha = 91.2\%$ and $\varepsilon = 1.5\%$ achieved in previous work with a flat $Al/NiCrO_x$/AR configuration [15] and with the record values of $\alpha = 97\%$ and $\varepsilon = 5\%$ obtained on a 3-layers stack [13]. The calculation accounts not only for the enhancement of α, but also for the combined optimization of both α and ε. The higher values

Table 1. Parameters relevant to the optimization of α and ε of a solar thermal collector. The first line corresponds to the solution that maximizes $f_1 + f_2$. The next three lines correspond to selected Pareto-optimal solutions.

P (nm)	H (nm)	f	r	t_1 (nm)	t_2 (nm)	α	ε	
1345	1960	0.96	0.45	50	50	97.8%	4.8%	f_1+f_2 max
1435	1975	0.99	0.31	55	50	98.4%	5.8%	P-optimal
795	1590	0.90	0.28	50	50	96.1%	4.1%	P-optimal
560	545	0.95	0.28	50	50	95.2%	3.7%	P-optimal

achieved for the absorptance α are coupled with an increase of the emittance ε. The interpretation of the results is facilitated by the definition of a cut-off wavelength λ_c describing the transition of the reflectance from zero to one (more specifically, λ_c is defined as the wavelength at which $R(\lambda_c) = 50\%$). A shift of λ_c to higher wavelengths leads to an increase of both α and ε (see the definitions of α and ε and the representations of $B_S(\lambda)$ and $B_a(\lambda)$ in Fig. 2). This increase of the emittance ε to values that stay below 5.8% for the solutions presented in Table 1 is however low enough to maintain strong performances.

The solution that provides the maximal value for $f_1 + f_2$ gives absorptance and emittance values of $\alpha = 97.8\%$ and $\varepsilon = 4.8\%$. The reflectance $R(\lambda)$ associated with this solution is shown in Fig. 2. The figure includes for comparison the reflectance spectrum of a flat uncoated Al as well as the reflectance spectrum of a flat Al/$NiCrO_x$/SnO_2 stack with $t_1 = t_2 = 50$ nm. These results confirm that the cermet and anti-reflection coatings play their role in reducing significantly the reflectance $R(\lambda)$ in the main part of the solar spectrum $B_S(\lambda)$. This explains the high values of α. $R(\lambda)$ then increases rapidly to values that are close to 1 for the main part of the absorber black-body spectrum $B_a(\lambda)$. This explains the small values of ε.

As the solar and black-body spectra only slightly overlap, a conjoint optimization of both the solar absorptance α and the thermal emittance ε was indeed possible. The transition in the reflectance must however be sharp and located at a wisely chosen cut-off wavelength (λ_c). This cut-off wavelength depends on the black-body temperature because of optimization considerations [6]. The ideal reflectance curve of a solar absorber is represented in Fig. 4 of Cao *et al.* [6]. It indicates a cut-off wavelength λ_c at around 2.5 μm, for a 373 K black-body temperature. With the addition of the underlying structure, the cut-off wavelength observed in Fig. 2 with our Al/$NiCrO_x$/SnO_2 configuration has shifted from 1.5 μm to 2.5 μm indeed. The shift of λ_c to longer wavelengths leads to a strong increase of the solar absorptance α. It also leads to a slight increase of the thermal emittance ε. Emittance values of the order of 5% are however tolerable as they do not spoil performances.

The comparison between the flat and waffle-shaped Al/$NiCrO_x$/SnO_2 configurations proves that the pattering of the Al substrate has a significant impact on the reflectance spectrum and therefore on the absorptance α and the emittance ε. Values of $\alpha = 84.9\%$ and $\varepsilon = 1.7\%$ are indeed obtained with the flat Al/$NiCrO_x$/SnO_2 configuration (taking $t_1 = t_2 = 50$ nm), while values of $\alpha = 97.8\%$ and $\varepsilon = 4.8\%$ are obtained with the waffle-shaped configuration. We checked that relative variations of 5% on optimal parameters lead to average relative deviations of $\Delta\alpha/\alpha = 0.2\%$ on α and $\Delta\varepsilon/\varepsilon = 4.0\%$ on ε. This robustness is due to the fact that the transition in $R(\lambda)$ from almost zero to one arises in a wavelength region where the $B_S(\lambda)$ and $B_a(\lambda)$ spectra have small intensities.

The optimization of parameters was performed by assuming normal incidence of the sun light, according to the standard practice in the field. In domestic use, flat-plate solar thermal collectors are of course not all the time exposed at normal incidence and the performances will therefore be degraded. This point should be considered in practical applications. The solutions

listed in Table 1 represent different alternatives for the realization of a high-performance solar thermal collector. The optimization significantly enhanced the solar absorptance α with a reasonably moderate increase of the thermal emittance ε, hence reaching the expected record performances. Making a choice between these different solutions will depend on the trade-off we want to have between α and ε and on other practical issues.

4. Conclusion

We applied a multi-objective genetic algorithm to the optimization of a solar thermal collector that consists of a waffle-shaped Al substrate with $NiCrO_x$ and SnO_2 conformal coatings. This problem involved the determination of optimal geometrical parameters in order to (i) maximize the solar absorptance α and (ii) minimize the thermal emittance ε. By using a multi-objective genetic algorithm, we actually obtained a whole set of Pareto-optimal solutions. These solutions represent different alternatives for the realization of a collector, the choice of a particular solution depending on a trade-off between α and ε. The values of $\alpha = 97.8\%$ and $\varepsilon = 4.8\%$ achieved in this work turn out to be competitive with record values found in the literature. Approaching this problem by a systematic scan on parameters would have been untractable considering the huge number of possibilities (48,763,605,000) and the time required for each evaluation of the fitness (up to 30 hours on a supercalculator). The genetic algorithm could however address this problem by evaluating in parallel only a reduced number of possibilities. This proves the interest of multi-objective genetic algorithms for addressing complex optimization problems in physics.

Appendix: Effective fitness based on a Pareto-classification of the population

We define in this Appendix the effective fitness f_{eff} that was used with the multi-objective genetic algorithm [26]. We refer as previously by n_{pop} to the size of the population, by n to the number of parameters x_i and by m to the number of components f_j of the objective function. Pareto-optimal solutions were defined as solutions that are not dominated. They receive a rank of 1. Solutions that are only dominated by solutions of rank 1 receive a rank of 2. Solutions of rank 3 are only dominated by solutions of rank 1 and 2. We can proceed in this way and attribute a rank to the whole population. The effective fitness f_{eff} must be higher for individuals of lower rank if we want the GA to search for Pareto-optimal solutions.

We proceed therefore in the following way to define the effective fitness: all individuals of rank 1 receive an effective fitness of n_{pop}. We then define a sharing function in order to reduce, amongst individuals of the same rank, the effective fitness of individuals that are too close from each other. This will indeed avoid early convergence of the GA to a given individual. For individuals of the same rank, we define a distance matrix whose components are defined by $d_{k,l} = \sqrt{\sum_{l=1}^{n}(x_i[k] - x_i[l])^2/(x_i^{\max} - x_i^{\min})^2}$, where $x_i[k]$ refers to the parameter x_i of an individual k, $x_i^{\max} = \max_{k\in[1,n_{\text{pop}}]} x_i[k]$ and $x_i^{\min} = \min_{k\in[1,n_{\text{pop}}]} x_i[k]$. The sharing function between two individuals is then defined by $S_{k,l} = 1 - (d_{k,l}/\sigma_{\text{share}})^2$ if $d_{k,l} \leq \sigma_{\text{share}}$ and 0 otherwise. Following previous work [25, 26], we take $\sigma_{\text{share}} = 0.5/\sqrt[n]{10}$. We then define the niche count of a given individual by $m_k = \sum_l S_{k,l}$, where the sum is restricted to individuals of the same rank. The effective fitness of each individual is finally divided by its niche count. The effective fitness of all individuals of rank 2 is then initialized with a value of $0.99 \times f_{\text{eff,min}}[\text{rank 1}]$, where $f_{\text{eff,min}}[\text{rank 1}]$ refers to the minimal value of the effective fitness for the individuals of rank 1. We proceed by computing the distance matrix $d_{k,l}$, the sharing function $S_{k,l}$ and the niche count m_k for all individuals of rank 2. Their effective fitness is then divided by their niche count. We continue in this way until the whole population has been attributed an effective fitness.

Acknowledgments

A.M. is funded by the Fund for Scientific Research (F.R.S.-FNRS) of Belgium. L.G. is supported by FNRS-FRIA. D.N. and T.C. acknowledge the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme and initiated by the Belgian Science Policy Office. A.M, D.N. and T.C. are members of naXys, Namur Center for Complex Systems. This research used resources of the "Plateforme Technologique de Calcul Intensif (PTCI)" (http://www.ptci.unamur.be) located at the University of Namur, Belgium, which is supported by the F.R.S.-FNRS under the convention No. 2.4520.11. The PTCI is member of the "Consortium des Equipements de Calcul Intensif (CECI)" (http://www.ceci-hpc.be). Erwin Volon is acknowledged for his Solid Works representations of the Al/$NiCrO_x$/SnO_2 structure.

Bibliography

R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2010.

S. Andradóttir. Chapter 20: An overview of simulation optimization via random search. In S. G. Henderson and B. L. Nelson, editors, *Simulation*, volume 13 of *Handbooks in Operations Research and Management Science*, pages 617–631. Elsevier, 2006.

E. Angel and V. Zissimopoulos. On the classification of np-complete problems in terms of their correlation coefficient. *Discrete Applied Mathematics*, 99(1):261–277, 2000.

A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, 1995.

I. A. Basheer and M. Hajmeer. Artificial neural networks: Fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43:3–31, 2000.

D. Beasley, D. R. Bull, and R. R. Martin. An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):56–69, 1993a.

D. Beasley, D. R. Bull, and R. R. Martin. An overview of genetic algorithms: Part 2, research topics. *University Computing*, 15(4):170–181, 1993b.

M. A. Beaumont. Evolution of optimal behaviour in networks of boolean automata. *Journal of Theoretical Biology*, 165(4):455–476, 1993.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New-York, 2007.

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

C. Blum and K. Socha. Training feed-forward neural networks with ant colony optimization: an application to pattern classification. In *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, pages 6 pp.–. IEEE, 2005.

K. D. Boese and A. B. Kahng. Simulated annealing of neural networks: The 'cooling' strategy reconsidered. In *1993 IEEE International Symposium on Circuits and Systems*, volume 4, pages 2572–2575. IEEE, 1993.

J. A. Bullinaria. Understanding the emergence of modularity in neural systems. *Cognitive Science*, 31(4):673–695, 2007.

E. Bullmore and O. Sporns. Complex brain networks: Graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198, 2009.

R. Calabretta, A. Di Ferdinando, D. Parisi, and F. C. Keil. How to learn multiple tasks. *Biological Theory*, 3(1):30–41, 2008.

R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998.

J. Clune, J.-B. Mouret, and H. Lipson. The evolutionary origins of modularity. *Proceedings of the Royal Society of London B: Biological Sciences*, 280(1755): 20122863, 2013.

K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50. Morgan Kaufmann Publishers, 1989.

D. Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 400(1818):97–117, 1985.

D. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 425(1868):73–90, 1989.

D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 439(1907):553–558, 1992.

M. Dorigo and M. Colombetti. *Robot Shaping: An Experiment in Behavior Engineering*. MIT Press, 1998.

S. N. Dorogovtsev and J. F. F. Mendes. *Evolution of Networks: From Biological Nets to the Internet and WWW*. Oxford University Press, 2013.

A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, Heidelberg, 2015.

S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus. Does code decay? assessing the evidence from change management data. *IEEE Transactions on Software Engineering*, 27(1):1–12, 2001.

D. Floreano and L. Keller. Evolution of adaptive behaviour in robots by means of darwinian selection. *PLOS Biology*, 8(1):1–8, 2010.

C. M. Fonseca and P. J. Fleming. Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization. In *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 416–423. Morgan Kaufmann Publishers, 1993.

C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

B. A. Garro, H. Sossa, and R. A. Vazquez. Design of artificial neural networks using a modified particle swarm optimization algorithm. In *2009 International Joint Conference on Neural Networks*, pages 938–945. IEEE, 2009.

C. Gershenson. Classification of random boolean networks. In *Proceedings of the 8th International Conference on Artificial Life*, pages 1–8. MIT Press, 2002.

W. Gerstner and W. M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Boston, 1989.

L. H. Hartwell, J. J. Hopfield, S. Leibler, and M. A. W. From molecular to modular cell biology. *Nature*, 402:47–52, 1999.

R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms*. John Wiley & Sons, Hoboken, New Jersey, 2004.

M. Hollander, D. A. Wolfe, and E. Chicken. *Nonparametric Statistical Methods*, volume 751. John Wiley & Sons, Hoboken, New Jersey, 2013.

H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Elsevier, Amsterdam, 2004.

K. Hornberger. Introduction to decoherence theory. In A. Buchleitner, C. Viviescas, and M. Tiersch, editors, *Entanglement and Decoherence: Foundations and Modern Trends*, volume 768, pages 221–276. Springer, Berlin, Heidelberg, 2009.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann Publishers, 1995.

N. Kashtan and U. Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences*, 102(39):13773–13778, 2005.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-competitive Machine Intelligence*, volume 5. Springer Science & Business Media, 2006.

F. Kursawe. A variant of evolution strategies for vector optimization. In *Parallel Problem Solving from Nature*, pages 193–197. Springer, 1991.

H. Lipson, J. B. Pollack, and N. P. Suh. On the origin of modular variation. *Evolution; international journal of organic evolution*, 56(8):1549—–1556, 2002.

D. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

S. Marsland. *Machine Learning: an Algorithmic Perspective*. Chapman and Hall/CRC, New-York, 2015.

D. Mermin. *Calculs et Algorithmes Quantiques: Méthodes et Exemples*. EDP sciences, Les Ulis, 2010.

M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

F. Mintert, C. Viviescas, and A. Buchleitner. Basic concepts of entangled states. In A. Buchleitner, C. Viviescas, and M. Tiersch, editors, *Entanglement and Decoherence: Foundations and Modern Trends*, volume 768, pages 61–86. Springer, Berlin, Heidelberg, 2009.

T. M. Mitchell. *Machine learning*. McGraw-Hill Series in Computer Science, Boston, 1997.

H. Neudecker. A note on kronecker matrix products and matrix equation systems. *SIAM Journal on Applied Mathematics*, 17(3):603–606, 1969.

M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

D. Nicolay. Modélisation et apprentissage des réseaux de neurones artificiels. Master's thesis, University of Namur, 2012.

M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

C. G. Nitash, B. Lundrigan, L. Smale, and A. Hintze. The effect of periodic changes in the fitness landscape on brain structure and function. *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, pages 469–476, 2018.

S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-organizing Machines*. MIT press, 2000.

P. Peretto. *An Introduction to the Modeling of Neural Networks*. Cambridge University Press, 1992.

R. Pfeifer and C. Scheier. *Understanding Intelligence*. MIT Press, 2001.

A. Prieto, B. Prieto, E. Martinez Ortigosa, E. Ros, F. Pelayo, J. Ortega, and I. Rojas. Neural networks: An overview of early research, current frameworks and new challenges. *Neurocomputing*, 214:242 – 268, 2016.

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

R. Rojas. *Neural Networks: A Systematic Introduction*. Springer, Berlin, 1996.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958.

F. Rosenblatt. Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithm. In *Proceedings of the First International Conference on Genetic Algorithms and Their Application*, pages 93–100. Lawrence Erlbaum Associates, 1985.

R. S. Sexton, B. Alidaee, R. E. Dorsey, and J. J. D. Global optimization for artificial neural networks: A tabu search application. *European Journal of Operational Research*, 106(2):570–584, 1998.

S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

P. F. Stadler. Towards a theory of landscapes. In R. López-Peña, R. Capovilla, R. García-Pelayo, H. Waelbroeck, and F. Zertuche, editors, *Complex Systems and Binary Networks*, pages 78–163. Springer, 1995.

A. Thompson. *Hardware Evolution: Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution*. Springer Science & Business Media, 2012.

G. Tononi, A. R. McIntosh, D. P. Russell, and G. M. Edelman. Functional clustering: Identifying strongly interactive brain regions in neuroimaging data. *NeuroImage*, 7 (2):133–149, 1998.

S. Valverde. Breakdown of modularity in complex networks. *Frontiers in Physiology*, 8:497, 2017.

P. J. M. van Laarhoven and E. H. L. Aarts. Simulated annealing. In *Simulated Annealing: Theory and Applications*, volume 37, pages 7–15. Springer Netherlands, Dordrecht, 1987.

M. Villani, A. Filisetti, S. Benedettini, A. Roli, D. Lane, and R. Serra. The detection of intermediate-level emergent structures and patterns. In *European Conference on Artificial Life*, volume 12, pages 372–378. MIT Press, 2013.

M. Villani, A. Roli, M. Filisetti, M. Fiorucci, I. Poli, and R. Serra. The search for candidate relevant subsets of variables in complex systems. *Artificial Life*, 21(4): 412–431, 2015.

G. P. Wagner and L. Altenberg. Perspective: Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976, 1996.

G. P. Wagner, M. Pavlicev, and J. M. Cheverud. The road to modularity. *Nature Reviews Genetics*, 8(12):921–931, 2007.

E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63(5):325–336, 1990.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.