# Evolving modular neural-networks through exaptation

Jean-Baptiste Mouret, Stéphane Doncieux

# Evolving modular neural-networks through exaptation

Jean-Baptiste Mouret Stéphane Doncieux

*Abstract*— **Despite their success as optimization methods, evolutionary algorithms face many difficulties to design artifacts with complex structures. According to paleontologists, living organisms evolved by opportunistically co-opting characters adapted to a function to solve new problems, a phenomenon called *exaptation*. In this paper, we draw the hypotheses (1) that exaptation requires the presence of multiple selection pressures, (2) that Pareto-based multi-objective evolutionary algorithms (MOEA) can create such pressures and (3) that the modularity of the genotype is a key to enable exaptation. To explore these hypotheses, we designed an evolutionary process to find the structure and the parameters of neural networks to compute a Boolean function with a modular structure. We then analyzed the role of each component using a Shapley value analysis.**

**Our results show that: (1) the proposed method is efficient to evolve neural networks to solve this task; (2) genotypic modules and multiple selections gradients needed to be aligned to converge faster than the control experiments. This prominent role of multiple selection pressures contradicts the basic assumption that underlies most published modular methods for the evolution of neural networks, in which only the modularity of the genotype is considered.**

## I. Introduction

Finding the parameters and the topology of neural networks is a complex task that has been successfully tackled with evolutionary algorithms many times (e.g. [1]–[5]). Although these methods easily lead to good parameters for many systems, they turn out to be inefficient to evolve artifacts with arbitrary complex structures. In other words, current evolutionary algorithms are widely successful optimization methods but are bad engineers. This result is surprising with regard to the complexity reached by living organisms using "natural" evolution.

These difficulties echo the main criticisms addressed by early – and current – opponents to Darwin. While it is easy to admit that natural selection can maintain and improve a given trait, how do complex characters emerge at first? Put differently, how natural selection could favor the intermediate structures required to solve complex tasks? Darwin wrote a lengthy answer to this challenge in the last edition of *The Origin of Species* [6] in which he highlighted that a structure does not have to be employed for the same purpose during the whole evolution of a species. Consequently, structures might originate from an adaptation to a function and then be opportunistically co-opted to solve a new problem. Darwin termed this concept *preadaptation*, but modern evolutionary biologists use the less connoted word *exaptation* [7]. According to current paleontology, exaptation explains, in particular, crucial steps of the evolution of life such as the appearance of the digits of tetrapods, initially adapted to an aquatic environment [8], or first bones, useful for the storage of inorganic ions.

Despite the prominent role of exaptation in the evolution of complex organisms, this phenomenon seems to be, if not absent, at least rare in artificial evolution. Complex life forms are subject to many selection pressures such as the performance of their vision system, the ability to move faster, the variety of food they can eat, etc. As a consequence, evolution can improve organisms with respect to many selection gradients and the obtained adaptations can later be employed to solve problems raised by another selection gradient, resulting in an exaptation. Classical artificial evolution defines only one selection gradient, the fitness function, thus preventing the evolutionary process to exploit exaptation to lead to complex artifacts.

Recognizing this problem, several researchers (e.g. [9]–[12]) proposed to replace the fitness function by an implicit evaluation scheme in which individuals are situated in an artificial world. Inhabitants of these simulated environments compete for limited resources while trying to meet sexual partners to spread their genotype. These studies provide many insights about the dynamics of evolutionary systems and can lead to complex "artificial life forms"; however, they are not designed to solve concrete problems, such as creating a neuro-controller for a robot to clean a room or finding the optimal structure of a bridge. In this paper, we draw the hypothesis that Pareto-based multiobjective evolutionary algorithms (MOEA, see [13]) can explicitly provide multiple selection gradients, thus enabling exaptation to evolve complex and potentially useful artifacts.

The existence of a selection pressure may not be sufficient to improve a particular trait: if a gene has many pleiotropic effects[1], it may be impossible to improve one character without altering some other ones; consequently, the more independent the genes coding for unrelated traits, the more efficiently a selection gradient can be followed. In other words, the *genotype-phenotype map* – how genotype variations are reflected in the phenotype – should be *modular* (see [14]). This intuition is confirmed by a recent theoretical study [15] that highlights that genotypic modules, phenotypic modules and selection gradients should be "aligned" to fully benefit from the advantages of modularity for the evolvability of complex systems. Moreover, modularity makes easier the repetition of a functional sub-part in an organism. Such a duplicated module can then be employed for a new function without damaging its use in the original context. The impor-

Jean-Baptiste Mouret and Stéphane Doncieux are with the Université Pierre et Marie Curie (UPMC) - Paris 6, Institut des Systèmes Intelligents et de Robotique (ISIR), CNRS UMR 7222, F-75005, Paris, France; e-mail: {mouret,doncieux}@isir.fr

[1]Pleiotropy occurs when a single gene influences multiple phenotypic traits.

tance of modularity to evolve complex systems and especially neural networks has been widely recognized ( [1], [16]–[19]) but, to our knowledge, neural-network modules have never been explicitly linked to selection gradients.

In this article, we propose an evolutionary process designed to exploit exaptation and modularity to synthesize complex systems. Our aim is twofold: (1) investigating the links between evolution, modularity and selection gradients and (2) introducing a scalable method to evolve complex systems. We illustrate our ideas by evolving neural-networks to compute a complex and modular Boolean function, a problem previously employed in another study about modularity [20].

The remainder of this paper is organized into four parts. Section II is an overview of the related previous work. Section III describes our approach, from the definition of a modular genotype-phenotype map to the use of MOEAs to create multiple selection gradients. Section IV details our experiment and analyzes the results using the Shapley value. The last section is a brief discussion.

## II. Previous Work

Only a few papers explicitly deal with exaptation and preadaptation in the evolution of artificial systems [11], [21]–[23]. De Oliveira [11] studies an artificial ecosystem of two-dimensional cellular automata designed to make difficult adaptations by natural selection. Exaptation was possible by exploiting a sequence of non-adaptations caused by genetic drift. Studying the evolution of neural networks, Miglino et al. [22] noticed that some changes of the neural networks were non-adaptive – and consequently only selected by chance – but were later required to increase the fitness. Miglino et al. conclude that this phenomenon was an evidence of artificial preadaptation. While these two papers report observations of preadaptation in an artificial context, Graham and Oppacher [23] proposed a process designed to exploit exaptation to improve existing evolutionary algorithms. Tackling a toy problem, the authors designed four fitness functions of increasing difficulty and associated them to four niches. Individuals of each niche were evaluated using the corresponding fitness but they were allowed to migrate between niches at the end of each generation. At the beginning of their experiments, individuals were viable only in the simplest niche but, after a few hundred generations, they managed to migrate from niches to niches to finally solve the most complex problem. These results demonstrate the power of exaptation to solve difficult tasks by providing different fitness gradients.

Using intermediate steps to evolve solutions for complex problems is often referred to as incremental evolution [4], [24]–[26]. In these works, the task is split into stages of increasing complexity. Candidate solutions are first selected using a first fitness function and, once a convergence criterion is reached, the experimenter changes the fitness to a more complex one. Despite its practical successes, this manual approach has several drawbacks (see [27]) such as the need to find a good switch criterion, the necessity to follow designer's ordering of sub-problems, the inability to explore multiple hypotheses and the absence of explicit selection pressures on the previous sub-tasks.

Mouret et al. [27] show that these drawbacks can be efficiently overcome by defining a multiobjective optimization problem in which each objective corresponds to a sub-task and by then solving it using a Pareto-based MOEA (see [13]). At the beginning of the process, individuals obtain a non-minimal fitness only for the simplest objectives. As they improve with respect to these objectives, they will reach the minimal performance for the sub-tasks required to try the more complex ones. As a consequence of the Pareto sort, individuals that obtain a non-minimal fitness on a previously unreachable sub-task will be non-dominated – and consequently selected – but best individuals for the simpler sub-tasks will be non-dominated too. Thus, the evolutionary process will automatically switch to complex tasks as soon as possible, while maintaining a selection pressure on previous tasks and while not depending on any a priori ordering of sub-tasks. Moreover, a part of the population may dominate with respect to one of the intermediate sub-tasks while another part can improve along another selection gradient; the process can thus simultaneously explore different evolutionary paths. Each selection gradient in this case relies on complete solutions features and has no direct impact on sub-parts of it. This differs from natural evolution, in which particular evolutionary pressures can shape specific organs. This difficulty will be addressed in the present paper by introducing modules in candidate solutions.

The evolution of modular systems and especially modular neural networks has drawn considerable attention in the last few years because of the intuitive hypothesis that they are more evolvable than monolithic ones. Most papers about the evolution of modular neural networks deal with the design of modular encodings with different approaches. For instance, Gruau [1], Hornby and Pollack [28] and Mouret and Doncieux [29] derive techniques from evolutionary programming to exploit the intrinsic modularity of tree-based representations; Doncieux and Meyer [30] and Reisinger et al. [19] use simpler representations based on a list of modules, directly encoded, and a blueprint that specifies how modules are connected. In these papers, the authors implicitly assume that modular systems will be favored by the evolutionary process. Consequently, they do not propose any explicit link between the selection pressures and the modules.

Nevertheless, this hypothesis is questionable. As noticed in [31], monolithic neural networks are often more efficient and easier to obtain than their modular counterpart. The observation that living organisms are highly modular, and that it explains at least a part of their success, raises the question: how modularity did emerge in nature? Wagner et al. [32] review many proposed explanations and conclude that the direct selection for evolvability, i.e. the implicit hypothesis behind modular encodings, is one of the less likely ones. Recent papers in theoretical biology emphasize the role of

the selection pressure to explain the modular organization of organisms. In particular, Lipson et al. [33] and Kashtan and Alon [20] employ numerical models to show that rapid changes of evolutionary pressures induce modular structures. In another theoretical study, Altenberg [15] concludes: "My main proposal is that the evolutionary advantages that have been attributed to modularity do not derive from modularity *per se*. Rather, they require that there be an 'alignment' between the spaces of phenotypic variation, and the selection gradients that are available to the organism."

## III. Method

### A. Modular genotype-phenotype map

Definitions of what a module is are multiple and rather vague. In this work, we chose to call a module a subset of a system containing several entities functionally integrated and largely independent of the entities that constitute the other modules. In the context of neural-networks, the interactions between neurons are defined by the connections and the synaptic weights; so we can consider the network as a directed graph and try to find highly connected clusters. This problem has a long history in graph theory due to its usefulness in sociology – to analyze groups of people – and in biology – notably to analyze gene regulatory networks. Leicht and Newman [34] recently proposed a robust and efficient approach to detect modules by optimizing a measure (simply called "modularity") over the possible divisions of a network. Broadly, Leicht and Newman [34] state that, in a good division, the number of edges between groups is smaller than expected, i.e. in most cases, smaller than the mean number of edges in a random network of the same size. Good divisions regarding this measure can be computed using a spectral analysis method that demands $O(n^2)$ time, where $n$ is the number of vertices in the network.

Such sub-graphs of a neural-network constitute phenotypic modules. Genotypic modules are more straightforward to define since one has only to check that genetic operators only affect sub-parts of the genotype. Using the approach introduced in [34], phenotypic modules can be extracted from any network and many modular genotypes can be designed. Consequently, the important point for the evolvability of complex systems is not only the existence of modules in the genotype and in the phenotype; it is how genotypic modules relate to phenotypic ones.

This relation, called the genotype-phenotype map, is said to be modular if genotypic modules are developed into phenotypic modules[2]. The neural network modules emerge from the topology of the neural network. As a consequence, a small change in the neural network, for instance the addition of a connection induced by a mutation, can considerably modify the optimal division. This makes it difficult to define a modular genotype-phenotype map: depending on the neural network encoding, it can be difficult to guarantee that a genotypic module will lead to a meaningful phenotypic module. In a biological context, Wagner et al. [32] explain that only two processes can lead to a modular genotype-phenotype map: parcellation, which consists in a differential suppression of pleiotropic effects between groups of characters, and integration, which corresponds to the selective acquisition of pleiotropy among characters from the same group. How could these two concepts be transposed to neural networks? Removing pleiotropic effects can be interpreted as making the contour of phenotypic modules more resistant to mutations. Parcellation can consequently be the isolation of the part of the genome coding for a module, extracted using the approach described in [34]. Thus a phenotypic module would remain stable during the evolution. Integration can be seen as the replacement of a module by a copy of another one, thus providing a repetition mechanism.

Trying to implement these operations as mutations in the simplest genotype possible, we start by considering a typical graph-based direct encoding in which two kinds of mutations are possible:

- structural mutation: addition/removal of a neuron or a connection;
- parametric mutation: change of a randomly chosen

[2]In real organisms, many levels of modules are nested and so the modularity of genotype-phenotype map can be difficult to express clearly.

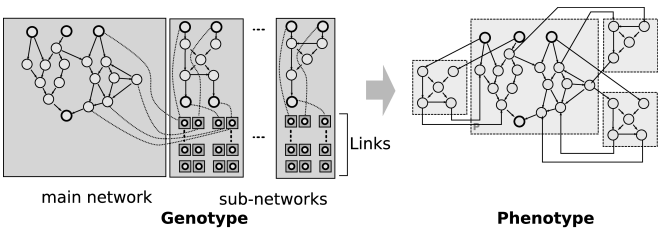


Fig. 1. An individual's genotype is made of a main network and one or several sub-networks. Each sub-network is associated with a list of links that connect the sub-networks to the main network. If several lists of links exist for a same sub-network, the latter is added several times to the main network. The neural network on the right is an example of a phenotype that could have been encoded with the genotype on the left.

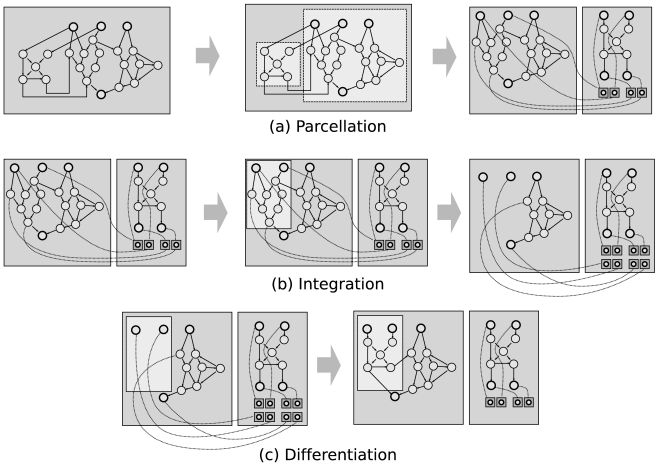


Fig. 2. The three main mutation operators. (a) Parcellation: isolation of a sub-network. (b) Integration: replacing a sub-network by a link to a parcellated one (this allows the repetition of a module) (c) Differentiation: putting a sub-network back in the main network.

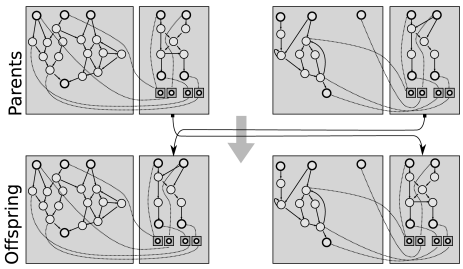


Fig. 3. Principle of the cross-over operator. If both parents have a parcellated module, it is exchanged to create their offspring. Otherwise, the each children is a copy of one of the parent (mutation operators are then applied).

synaptic weight or a neuronal bias; we use here a change in a set of 9 possible values (see appendix).

To easily add integration and parcellation as mutation operators, we then defined our neural network encoding as a main network – encoded using the previously described direct encoding – and a list of sub-networks, each of them associated with links towards the main network (figure 1). Before applying any operator, the main network is divided into modules using [34] and their inputs and outputs are identified. Inputs of a module are either a connection between an extra-module neuron to an intra-module one, or an input of the main network. Outputs are defined similarly. The specification of each modular operator is now straightforward. The parcellation acts in three steps (figure 2(a)):

- removal of a module from the main network;
- addition to the sub-network list;
- addition of the links to the sub-network's links list.

The removed module can be, either randomly selected or deterministically chosen as the best module regarding one of the sub-tasks. In this work, this second approach is used. The integration requires also three steps (figure 2(b)):

- random selection of a module $m_1$ from the sub-network list;
- random selection of a module $m_2$ of the main network with the same number of inputs and outputs as $m_1$;
- removal of the module $m_2$ from the main network;
- addition of the links to the sub-network links list.

To counterbalance the effects of these two operators, we added an operator named "differentiation" that puts back a sub-network into the main network and removes the corresponding links in the list (figure 2(c)). This operator undoes parcellation and integration and it can be used by the evolutionary process to create a variant of a repeated module. Each operator is assigned a mutation rate (between $0.05$ and $0.2$, see appendix).

A simple cross-over operator is implemented by exchanging two parcellated modules (figure 3).

Note that the proposed encoding aims at being simple and abstractly related to biology in order to investigate the links between selection gradients, genotypic modules and selection gradients. Our main goal is therefore not to introduce a novel encoding for a neural network but is to provide a tool to study modularity. Other modular encodings for neural networks (e.g [17], [19], [30]) might be used but their structure makes them harder to link to biological literature.

### B. Selection gradients

Following the work of Altenberg [15], we "align" phenotypic modules and selection gradients, which means that each phenotypic module should be linked to a fitness function. To that aim, we propose to first define fitness functions able to evaluate sub-networks for potentially useful sub-functions. For each individual, the main network is divided into modules using the approach described in [34]. Each such module and each parcellated module are then evaluated according to each objective. We use the best value obtained by a module for each objective as the fitness of the individual with respect to this objective. Each of these fitness functions constitutes an objective of a multiobjective problem. The main fitness, which rewards the ability to solve the problem we are interested in, is added as another objective.

Using a Pareto-based MOEA to optimize this multiobjective problem has several consequences. An individual with an inefficient overall behavior will be considered as Pareto-optimal if it contains at least a good module for a sub-task. If the main task requires such an efficient module to build more complex solutions, it will be gradually improved thanks to the selection pressure inducted by the corresponding objective. This module can be later co-opted at any time. Moreover, the same module can be repeated (it may be useful elsewhere) or propagated in the population by cross-over.

Another important consequence of the use of a MOEA is that, as shown in [27], solving each sub-task is not mandatory. If the evolutionary process finds better solutions without using modules, these individuals will dominate with regard to the main objective and will consequently be selected. Different hypotheses about the usefulness of each sub-task can therefore be investigated at once: some individuals will be good at one sub-task while some other ones will contain useful modules for other sub-tasks; the evolutionary process will opportunistically modify them, without any a priori knowledge about the ordering of sub-tasks or about their utility.

### C. Shapley value

When designing and evaluating an evolutionary process, it is important to understand what makes it efficient. A particular genetic operator may be mandatory to lead to working results; it may also only change the convergence speed or even have no influence on the results. Furthermore, the critical components may be combinations of some particular genetic operators, in which all of them are required to bring improvements but each one is useless alone. In the context of this paper, we would like to be able to estimate the usefulness of each modular operator (parcellation, integration, differentiation and cross-over) and to evaluate what brings the multiobjective approach.

Evaluating the contribution of each such component is equivalent to finding a fair allocation of gains, e.g. the fitness,

between players in a coalitional game, a problem solved in game theory by the Shapley value [35]. The Shapley value is defined from the marginal contribution of a player $i$ to a coalition $S$, where $i \notin S$ and $v(S)$ is a value function (the fitness or any other relevant measure of efficiency):

$$\Delta_i(S) = v(S \cup \{i\}) - v(S)$$

The Shapley value is then defined by the payoff $\gamma_i$ of each player $i \in N$:

$$\gamma_i = \frac{1}{|N|!} \sum_{r \in R} \Delta_i(S_i(r))$$

where $R$ is the set of all $|N|!$ orderings of $N$ and $S_i(r)$ is the set of players preceding $i$ in the ordering $r$.

Once the efficiency of the $2^n$ configurations has been evaluated, the Shapley value can be computed as a summation of $v(s)$ for all the configurations, properly weighted by the number of possible orderings of the elements (see [36]):

$$\gamma_i = \underbrace{\frac{1}{|N|!} \sum_{S \subset N, i \in S} v(s) \cdot (|S|-1)! \cdot (|N|-|S|)!}_{\text{configurations with i}} - \underbrace{\frac{1}{|N|!} \sum_{S \subset N, i \notin S} v(S) \cdot (|S|)! \cdot (|N|-|S|-1)!}_{\text{configurations without i}}$$

## IV. Experiment

### A. Experimental setup: $\big[(a \oplus b) \wedge (c \oplus d)\big]$

We evaluated the proposed approach on the evolution of neural networks to compute the Boolean function $\big[(a \oplus b) \wedge (c \oplus d)\big]$, where $a$, $b$, $c$ and $d$ are Boolean values and $\oplus$ denotes the exclusive "or" operator (xor). This function, previously used in [20] to study the evolution of modular NAND networks, has a clear modular structure: it is made of two "xor" functions, each of them requiring at least one hidden neuron, linked by a simpler logical "and". The truth table of $\big[(a \oplus b) \wedge (c \oplus d)\big]$shows that a simple neural network that returns "false" for any input would have a $75\%$ success rate, a good score at the beginning of the evolutionary process. As a consequence, these degenerated neural networks could easily invade the population whereas they do not constitute a good starting point. This makes the typical single-objective fitness for this function very deceptive. The $\big[(a \oplus b) \wedge (c \oplus d)\big]$function therefore constitutes a simple illustration of the situations in which exaptation could be useful: the typical fitness does not provide useful enough search gradients and we can suggest a helpful sub-function (xor).

We used the sum of errors for each possible set of inputs as the main fitness (expressed in a fitness maximization scheme):

$$F_x = 1 - \frac{1}{16} \sum_{i=1}^{16} |o_i - d_i|$$

where $o_i$ is the output of the neural network for the input set $i$ and $d_i$ the desired output. Each neural network is simulated during $100$ time-steps. Since we do not constrain the structure of the neural networks, nothing prevents them from oscillating. To avoid such behaviors, we attribute an arbitrary low fitness if the output is not constant during the last $10$ time-steps.

We then defined a second objective $F_m$ that rewards the efficiency of a "xor" module in an individual:

$$F_m = \max_{m \in M} F_{xor}(m)$$

where $M$ is the set of all modules. Considering that a module $m$ is compatible if it has 2 inputs and 1 output, $F_{xor}$ is the sum of errors for the module $m$ with respect to the function "xor":

$$F_{xor}(m) = \begin{cases} 1 - \frac{1}{4} \sum_{i=1}^{4} |o_i - d_i| \text{ if m is compatible} \\ -1 \text{ otherwise} \end{cases}$$

The NSGA-II algorithm was employed with a population of $400$ individuals and the parameters described in appendix. We launched five sets of experiments, each of them made of $32$ runs:

- exaptation: parcellation (P), integration (I), differentation (D), cross-over (C) and multiple selection pressures (M);
- M: multiple selection pressures only, i.e. no genotypic modules;
- P+I+D+C: genotypic modules only;
- standard: direct encoding without genotypic modules and without selection pressures;
- NEAT, a popular neuro-evolution approach [3].

We didn't compare our results with a modular encoding for neural network because (1) they are very complex to implement and to tune and (2) our main focus is on the benefits of linking selection gradients to modules.

### B. Results

Figure 4(a) shows the proportion of converged runs ($F_x > 0.9$) as a function of generation, for each of the three investigated approaches. Less than half of the control runs converged within $5000$ generations. This result agrees with [20], in which the authors report that only $72\%$ of their control runs converged in less than $10^5$ generations. NEAT leads to substantially better results since almost all the runs converged in $1500$ generations and more than half of them in only $500$ generations. This confirms the published results in which NEAT outperforms direct encodings (e.g. [3]). Nevertheless, the exaptation-based approach converged faster than NEAT. The $32$ experiments converged in less than $1000$ generations and half of them in about $300$ generations. Figure 4(b) corroborates this observation: on average, $400$ generations are required with the exaptation-based approach while NEAT need $700$ generations[3].

Surprisingly, not only did the approach based only on the genotypic modularity (P+I+D+C) not improve the convergence rate over the control experiments, but also it deteriorated it slightly. The runs that employed only the multiple

[3]This difference is statistically significant ($p < 0.003$).

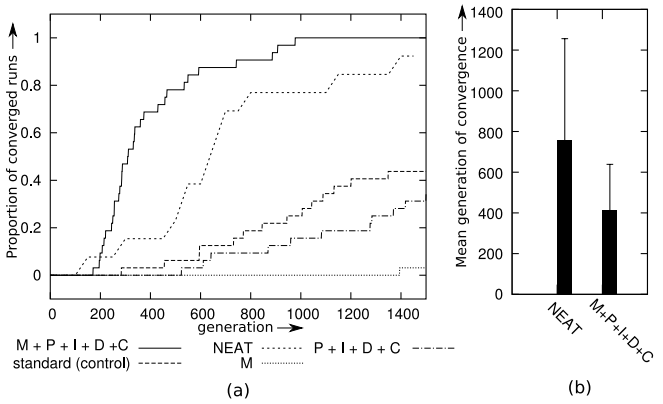


Fig. 4. (a) Proportion of converged runs ($F_x > 0.9$) as a function of generation. "M+P+I+D+C" denotes the full exaptation experiments, "standard" denotes the control experiments, which use a simple direct encoding (with the same parameters as the one used by the exaptation experiment) and one objective; the experiments "M" correspond to the multiobjective approach with the modular genetic operators disabled; in "P+I+D+C", the modular operators are used with the main fitness only. (b) Mean generation of convergence and standard deviation (the other experiments are omitted because only a few runs converged in less than 5000 generations). The differences between these two sets of experiments are statistically significant ($p < 0.003$).

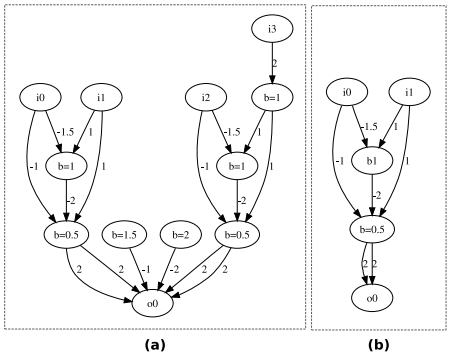


Fig. 5. (a) Typical neural network obtained with the proposed approach. (b) The parcellated module used in (a).

pressures scheme (M) obtained even worse results: only one run converged in less than 1500 generations. Given that our exaptation experiment used the combination of P, I, D, C and M, these results demonstrate that, at least in this experiment, both a modular encoding *and* multiple selection gradients are required to improve the efficiency of the evolutionary process. A simple "waste of resources" may explain the results obtained with the "M" experiment: a part of the population is used to maintain many Pareto-optimal candidate solutions with a low main fitness; if this scheme does not improve the evolutionary process, it is broadly equivalent to reducing the population size, hence possibly deteriorating the convergence rate over the control experiment. Further investigations are needed to fully understand this phenomenon.

A typical neural network obtained with the exaptation approach is drawn on figure 5. It Is clearly made of two repeated modules, each of them computing a "xor" function.

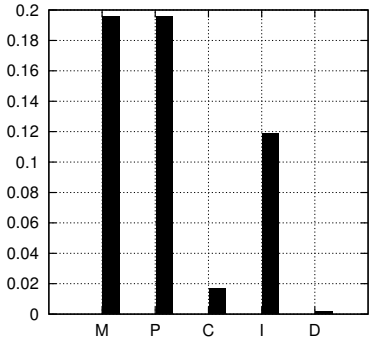


Fig. 6. Shapley values for each component. M: Multiple selection gradients; P: Parcellation; C: Cross-over; I: Integration; D: differentiation.

These modules are two instances of a parcellated module, which obtained an optimal fitness according to $F_{xor}$.

### C. Shapley value analysis

To draw a better picture of what makes the proposed approach efficient, we employed the Shapley value analysis described in section III-C. We investigated the role of the use of two selection gradients (M), of the parcellation (P), of the cross-over (C), of the integration (I) and of the differentiation (D). The efficiency of $2^5 = 32$ variants must therefore be evaluated.

Several choices are available for the value function, which should reflect the efficiency of a variant. We chose to focus our work on the convergence rate because we are primarily interested in getting as much successful experiments as possible. In some configurations, some components of our system can substantially decrease the convergence rate. For instance, we found that using the two selection gradients without parcellation lead to a $3\%$ success rate whereas the control experiment has a $44\%$ rate (section IV-B). If this component is required to obtain good scores in other configurations, it should have a high Shapley value but this value can be substantially lowered by these bad configurations, hiding the interesting analysis. We are interested in understanding which components *improve* over the control experiments and not about the fact that they can lower the scores in some cases. We consequently designed the following value function:

$$v(S) = \max(v_c(\emptyset), v_c(S))$$

where $v_c(S)$ is the convergence rate for the configuration $S$ and $v_c(\emptyset)$ denotes the convergence rate of the control experiment.

We launched 32 runs of each variant for 1500 generations with the same parameters as in section IV-B. The resulting Shapley values are displayed on figure 6.

The two highest values are obtained by the multiple selection gradients (M) and the parcellation operator (P). This means that these two components are *critical* to get the highest convergence rates. The fact that their values are very

close[4] confirms that using only one of them is not sufficient to improve the efficiency of a variant; both of them are required. The parcellation operator links genotypic modules to phenotypic modules and selections gradients are linked to phenotypic modules by the multiobjective approach (M). As a consequence, the obtained Shapley values highlights the need for the alignment suggested by the theoretical work of [15]: genotypic modularity (P) is useless if it is not linked to selection gradients. This conclusion contradicts the underlying hypothesis of current modular encodings for neural networks, which assumes that providing a modular genetic encoding is enough to improve the efficiency of the evolutionary process.

The next highest value is obtained by the integration operator, which is trivially useful for the $\big[(a \oplus b) \wedge (c \oplus d)\big]$function. This Shapley value is substantially lower than the two previous ones, showing that the key-point in the evolution of our modular neural networks is not the repetition mechanism. The cross-over only slightly improves the results and the differentiation has almost no effect.

## V. Discussion

In the artificial evolution paradigm, researchers often try to design a universal encoding which would allow to easily solve any problem by only specifying a simple and high-level fitness function. Hence, they put complex mechanisms in the encoding and try to use as little knowledge as possible in the fitness function. This approach is surprisingly disconnected from evolutionary biology, in which most descriptions of the evolution of living organisms primarily rely on selection pressures. When Darwin introduced his theory, nothing was known about the genome; but this didn't prevent him and his successors from successfully explaining the essence of the origin of most species.

The results presented in this paper indicate that the use of a single fitness function might be an over-simplification of the natural evolutionary process, which could prevent the evolution of complex artifacts. In particular, the Shapley value analysis shows that multiple selection pressures are required to efficiently evolve neural networks for the investigated problem. This result was expected as the presence of multiple selection gradients is the key to enable exaptation, hence the evolution of complex designs. However, a more interesting result of this analysis is that multiple selection gradients are useless by themselves. Both modularity and selections pressures are required: if one of them is omitted, the evolutionary process is less efficient than the control experiment. This leads to a new evolutionary scheme centered on evolutionary pressures and genome modularity.

Since we add more knowledge in the fitness function, this approach might move our research away from a mythical "universal problem solver". However, all the published methods to evolve complex systems rely on biases that could constrain them to specific problems, since we do not know any universally good bias. NEAT, for instance, begins the evolutionary process using only one topology, typically a feed-forward neural network without hidden nodes. This requires from the experimenter the implicit knowledge that such a network is a good starting point. Putting biases in the selections gradients possesses at least the advantage of being explicit, and therefore could allow a better analysis. Compared to incremental evolution methods, the approach presented in this paper puts fewer constraints on the search because intermediate steps are not mandatory. We only suggest potentially useful steps; the process is then free to use or to ignore them.

Having highlighted the need for multiple selection gradients and modular genomes, we can wonder if the selected genome and MOEAs are the best tools for these tasks. The main difference between the proposed encoding and other modular encodings lies in the idea that genotypic modules should develop to phenotypic modules. However, applying a selection pressure directly on the sub-network associated with a genotypic module might be enough to facilitate the emergence of phenotypic modules. Hence, more elaborated modular encodings (e. g. [29]) could be used as long as the inputs and outputs of each module can be determined.

The use of a classical Pareto-based MOEA to introduce multiple selection gradients allows us to rely on a well established set of efficient algorithms. However, recent theoretical [37] and empirical [38] studies demonstrate that such evolutionary processes were not well suited to optimize more than three antagonistic objectives. By employing an objective for each sub-function, the proposed process will easily require more than three objectives and consequently could put Pareto-based MOEA out of their limits. Nevertheless, if our starting hypothesis is valid, the objectives will not be antagonistic: the exaptation process should exploit individuals with a good fitness on one objective to build individuals for the more difficult objectives. Therefore, some individuals could dominate on most objectives. However, the dominance relation puts all objectives at the same level whereas we are mainly interested in the main task. Some experiments with modified domination criteria may therefore reveal themselves more efficient.

## VI. Conclusion

We explored the hypothesis that multiple selection gradients and a modular genotype-phenotype map were two key-points to evolve complex artifacts. To that aim, we defined a bio-inspired modular encoding and employed it with a Pareto-based MOEA in which one objective rewarded the efficiency of a module to complete a sub-function. We further hypothesized that exaptations should occur in this evolutionary framework by co-opting modules evolved for simple sub-functions to solve more complex ones.

We tested these ideas on the evolution of neural networks to compute a modular Boolean function. Our results show that: (1) the proposed method is efficient to solve this task; (2) both modularity *and* multiple selections gradients were required to converge faster than the control experiments. This

[4]Actually, they are equals but proving this equality is out of the scope of this work.

prominent role of multiple selection pressures contradicts the basic assumption that underlies previously published modular methods for the evolution of neural networks.

In further work, we will try to link modules used in other modular encodings to selection gradients in order to understand the set of features required for modular encoding. Then, other methods to create multiple selection gradients should be investigated because Pareto-based MOEA are not well suited to optimize more than three antagonistic objectives. Last, the proposed method should be benchmarked on other tasks to assess its generality.

## REFERENCES


[1] F. Gruau, "Automatic definition of modular neural networks," *Adaptive Behaviour*, vol. 3, no. 2, pp. 151–183, 1995.

[2] S. Nolfi and D. Floreano, *Evolutionary Robotics: Biology, Intelligence and Technology of Self-organizing Machines.* The MIT Press, 2001.

[3] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10(2), no. 99-127, 2002.

[4] J.-B. Mouret, S. Doncieux, and J.-A. Meyer, "Incremental evolution of target-following neuro-controllers for flapping-wing animats," in *From Animals to Animats: Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB)*, 2006, pp. 606–618.

[5] J. J. Gauci and K. O. Stanley, "Generating large-scale neural networks through discovering geometric regularities," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007)*, 2007.

[6] C. Darwin, *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, 1859.

[7] S. Gould and E. Vrba, "Exaptation; a missing term in the science of form," *Paleobiology*, vol. 8, no. 1, pp. 4–15, 1982.

[8] M. Coates and J. Clack, "Polydactyly in the earliest known tetrapod limbs," *Nature*, vol. 347, no. 6288, pp. 66–69, 1990.

[9] T. Ray, "An approach to the synthesis of life," *Artificial Life II*, vol. 11, pp. 371–408, 1991.

[10] D. Ackley and M. Littman, "Interactions between learning and evolution," *Artificial Life II*, vol. 10, pp. 487–507, 1992.

[11] P. P. B. de Oliveira, "Simulation of Exaptive Behaviour," *Lecture notes in computer science*, pp. 354–354, 1994.

[12] S. Elfwing, E. Uchibe, K. Doya, and H. Christensen, "Biologically inspired embodied evolution of survival," *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3, 2005.

[13] K. Deb, *Multi-objectives optimization using evolutionnary algorithms*. Wiley, 2001.

[14] W. Callebaut and D. Rasskin-Gutman, *Modularity: Understanding The Development And Evolution Of Natural Complex Systems*. MIT Press, 2005.

[15] L. Altenberg, "Modularity in Evolution: Some Low-Level Questions," in *Modularity: Understanding the Development and Evolution of Natural Complex Systems*, W. Callebaut and D. Rasskin-Gutman, Eds. MIT Press, 2005.

[16] R. Calabretta, S. Nolfi, D. Parisi, and G. Wagner, "A case study of the evolution of modularity: towards a bridge between evolutionary biology, artificial life, neuro- and cognitive science," in *Proceedings of Artificial Life VI*, C. Adami, R. Belew, H. Kitano, and C. Taylor, Eds. MIT Press, 1998.

[17] G. Hornby, "Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design," *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pp. 1729–1736, 2005.

[18] S. Doncieux and J.-A. Meyer, "Evolution of neurocontrollers for complex systems: alternatives to the incremental approach," in *Proceedings of The International Conference on Artificial Intelligence and Applications (AIA 2004)*, 2004.

[19] J. Reisinger, K. O. Stanley, and R. Miikkulainen, "Evolving reusable neural modules," *Proc. of the Genetic and Evolutionary Computation Conference*, 2004.

[20] N. Kashtan and U. Alon, "Spontaneous evolution of modularity and network motifs," *Proceedings of the National Academy of Sciences*, vol. 102, no. 39, pp. 13 773–13 778, 2005.

[21] D. Stork, B. Jackson, and S. Walker, "Non-optimality via preadaptation in simple neural systems," *Artificial Life II*, vol. 10, pp. 409–429, 1992.

[22] O. Miglino, S. Nolfi, and D. Parisi, "Discontinuity in evolution: How different levels of organization imply pre-adaptation," in *Adaptive Individuals in Evolving Populations: Models and Algorithms*, 1996, pp. 399–415.

[23] L. Graham and F. Oppacher, "A multiple-function toy model of exaptation in a genetic algorithm," *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 4591–4598, Sept. 2007.

[24] I. Harvey, P. Husbands, and D. Cliff, "Seeing the light: artificial evolution; real vision," in *From Animals to Animats 3, Proceedings of the third international conference on Simulation of Adaptive Behavior*, 1994.

[25] J. Kodjabachian and J.-A. Meyer, "Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects," *IEEE Transactions on Neural Networks*, vol. 9, pp. 796–812, 1997.

[26] J. Urzelai and D. Floreano, "Incremental Evolution with Minimal Resources," *Proceedings of IKW99*, 1999.

[27] J.-B. Mouret and S. Doncieux, "Incremental Evolution of Animats Behaviors as a Multi-objective Optimization," in *From Animals to Animats 10*, 2008.

[28] G. Hornby and J. Pollack, "Creating high-level components with a generative representation for body-brain evolution," *Artificial Life*, vol. 8, no. 3, pp. 223–246, 2002.

[29] J.-B. Mouret and S. Doncieux, "Mennag: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars," *Evolutionary Intelligence*, pp. 187–207, 2008.

[30] S. Doncieux and J.-A. Meyer, "Evolving modular neural networks to solve challenging control problems," in *Proceedings of the Fourth International ICSC Symposium on engineering of intelligent systems (EIS 2004)*, 2004.

[31] C. P. Bowers and J. A. Bullinaria, "Embryological modelling of the evolution of neural architecture," in *Modeling Language, Cognition and Action*, 2005, pp. 375–384.

[32] G. P. Wagner, J. Mezey, and R. Calabretta, "Natural selection and the origin of modules," in *Modularity: Understanding the Development and Evolution of Natural Complex Systems*, W. Callebaut and D. Rasskin-Gutman, Eds. MIT Press, 2005.

[33] H. Lipson, J. Pollack, and N. Suh, "On the origin of modular variation," *Evolution*, vol. 56, no. 8, pp. 1549–1556, 2002.

[34] E. A. Leicht and M. E. J. Newman, "Community Structure in Directed Networks," *Physical Review Letters*, vol. 100, no. 11, p. 118703, 2008.

[35] L. S. Shapley, "A value for n-person games," *Contributions to the theory of games*, vol. 2, pp. 307–317, 1953.

[36] A. Keinan, B. Sandbank, C. C. Hilgetag, I. Meilijson, and E. Ruppin, "Fair attribution of functional contribution in artificial and biological networks," *Neural Computuation*, vol. 16, no. 9, pp. 1887–1915, 2004.

[37] O. Teytaud, "On the hardness of offline multi-objective optimization," *Evolutionary Computation*, vol. 15, no. 4, pp. 475–491, 2007.

[38] K. Praditwong and X. Yao, "How well do multi-objective evolutionary algorithms scale to large problems," *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 3959–3966, 2007.


## APPENDIX

- population size: 400
- weights/biases: $\{-2.0, -1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, 2.0\}$
- weight/bias mut. rate: $0.2$;
- min. neurons (random gen.): $10$
- max. neurons (random gen.) : $20$
- min. connections (random gen.) : $20$
- max. connections (random gen.) : $35$
- cross rate : $0.5$
- parcellation rate: $0.25$
- differentation rate: $0.02$
- integration rate: $0.1$
- rate of connection addition: $0.15$
- rate of connection removal: $0.25$
- rate of connection change: $0.1$
- rate of neuron add: $0.025$
- rate of neuron removal: $0.025$
- activation function: $y_i = \tanh\left(-b + 5 \cdot \sum_j w_{ij} x_j\right)$