HAL open science

Using a Map-Based Encoding to Evolve Plastic Neural Networks

Paul Tonelli, J.-B. Mouret

To cite this version:

Paul Tonelli, J.-B. Mouret. Using a Map-Based Encoding to Evolve Plastic Neural Networks. Evolving and Adaptive Intelligent Systems (EAIS), 2011, Paris, France. pp.9-16, 10.1109/EAIS.2011.5945909 . hal-01300710

HAL Id: hal-01300710

https://hal.science/hal-01300710v1

Submitted on 21 Nov 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Using a Map-Based Encoding to Evolve Plastic Neural Networks

Paul Tonelli
ISIR, CNRS UMR 7222
Univ. Pierre et Marie Curie
F-75005, Paris, France
Email: tonelli@isir.upmc.fr

Jean-Baptiste Mouret
ISIR, CNRS UMR 7222
Univ. Pierre et Marie Curie
F-75005, Paris, France
Email: mouret@isir.upmc.fr

***Abstract*—Many controllers for complex agents have been successfully generated by automatically desiging artificial neural networks with evolutionary algorithms. However, typical evolved neural networks are not able to adapt themselves online, making them unable to perform tasks that require online adaptation. Nature solved this problem on animals with plastic nervous systems. Inpired by neuroscience models of plastic neural-network, the present contribution proposes to use a combination of Hebbian learning, neuro-modulation and a a generative map-based encoding. We applied the proposed approach on a problem from operant conditioning (a Skinner box), in which numerous different association rules can be learned. Results show that the map-based encoding scaled up better than a classic direct encoding on this task. Evolving neural networks using a map-based generative encoding also lead to networks that works with most rule sets even when the evolution is done on a small subset of all the possible cases. Such a generative encoding therefore appears as a key to improve the generalization abilities of evolved adaptive neural networks.**

I. Introduction

Artificial Neural Networks (ANNs) are now ubiquitous in neuroscience, but also in computational intelligence; this technological usefulness of ANNs and their links with their biological neurons, make them natural candidates to design bio-inspired controllers for intelligent agents. Nevertheless, neural networks —and especially recurrent ones— are also complex dynamic systems which are difficult to design with conventional engineering methods. A sensible way to tackle this challenge is to employ evolutionary algorithms, mainly because they only care about the result and do not constrain the inner workings of solutions. Additionally, these algorithms can act both on the structure and the parameters of neuro-controllers.

Evolved ANNs show impressive results as controllers for autonomous robots and artificial agents [1]–[3]. But to be truly efficient in non-stationary problems, ANNs should be not only *adapted* to their task, but also *plastic* by adapting themselves during their lifetime. Following this line of thought, several papers proposed to add Hebbian learning rules to evolved ANNs [4]; this work has then been extended to include neuro-modulatory neurons that can trigger Hebbian learning in their adjacent neurons [5]. Despite promising results, these works required the agent to learn one choice with only two options (e.g. left or right in a maze) and each scenario was tested in the fitness function. This situation is not realistic for unknown environments, in which the agent will have to learn to react to situations for which it has not been selected. Moreover, current successful setups for evolved ANNs involves only a few outputs (less than 3) and a few inputs (less than 8).

The present paper aims at evolving plastic neural networks with better generalization abilities and able to use more input/outputs. Our approach relies on a neuroscience-inspired generative encoding [6], neuro-modulation of heterosynaptic Hebbian rules [5] and, to mitigate deception, an explicit selective pressures for new behaviors [7]–[9]. To evaluate the proposed approach, we used a simulated Skinner box [10], a typical setup in operant conditioning in which an agent must learn association rules between stimuli and actions.

II. Background

A. Computational Neuroscience

Most animals critically rely on their ability to adapt their behavior thanks to modifications of the strength of synapses in their nervous systems, a phenomenon called *synaptic plasticity* [11], [12]. This process has been famously modeled by the Hebbian rule [13], which states that the strength of a connection increases when both pre- and post-synaptic neurons have a strong activity. Further researches extended this learning rule to more general *heterosynaptic rules* that correlates the modifications to more complex combinations of pre and post synaptic activity [14]. A generalized heterosynaptic Hebbian rule can be written as follows [15].

$$\Delta w_{ij} = A \cdot a_i \cdot a_j + B \cdot a_i + C \cdot a_j + D \qquad (1)$$

where i and j are neurons, Δw_{ij} is the modification of synaptic weight w_{ij}, a_i is the activation of neuron i and A, B, C, D are four parameters of the rule.

Synapses changes are also modulated by the concentration of some of the molecules emitted by other neurons [16]. Several of these molecules, like dopamin or serotonin, are emitted by specific neurons of the brain and are involved in the adaptation processes; for example, the activity of dopaminergic neuron has been shown to be a critical for reinforcement learning [17]. To model this phenomena, the equation (1) can easily be extended to use a modulation factor m [5]:

$$\Delta w_{ij} = m \cdot (A \cdot a_i \cdot a_j + B \cdot a_i + C \cdot a_j + D) \qquad (2)$$

Researchers rely on these learning rules to build models that use neurons to describe basic functions of the brain, like action selection [18] or reinforcement learning [19], [20], [20], [21]. Most of the time, these models do not arbitrarily connect individual neurons; instead, artificial neurons are arranged in organized structures, called *neural maps*, that projects to other maps using a few connection schemes such as "one-to-one" and "one-to-all" [6], [18]. In particular, neural maps and neuro-modulation can be combined to model reinforcement learning in the basal ganglia [20], [21]. Considering these neuron-based models of basic brain functions, three main components appears to be useful to define brain-like neural networks: neural maps, heterosynaptic rules and neuro-modulation.

B. Evolution of Plastic Neural Networks

Neural networks have been recognized as a good building block for creating arbitrarily complex functions, both in computational neuroscience and in machine learning [22]. This make them classic candidates to design controllers for autonomous agents. Numerous methods have been proposed to find efficient neural networks for a given task, such as the one proposed by Kasabov [23], which incrementally adds neural networks, used as modular solvers, in a complex, static architecture. Another solution, Evolutionary Algorithms (EA), are appealing because (1) they only require an evaluation of the overall achievement of the task, and not a time-step by time-step error signal (as in supervised learning), (2) they can define both the parameters and the topology of the neural network and (3) they can simultaneously optimize several conflicting objectives. Many methods have been proposed to evolve neural networks, from the direct application of real-valued evolutionary algorithms to the evolution of "construction programs" whose instructions are interpreted to build neural networks (see [12], [22] for reviews).

In most works, synaptic weights are fixed by the EA and can therefore not be changed during the lifetime of the agent, whereas agents that adapt during their lifetime would be even more useful than agents perfectly adapted to their task. In the simplest cases, classic machine learning algorithms (e.g. back-propagation or reinforcement learning) can be used but they cannot be applied to networks of arbitrary topology. An alternative way of thought is to take inspiration from computational neuroscience by importing models of synaptic plasticity into evolved neural networks. Some of the early works relied on simple homeostatic rules [24], while more recent works employed more complex rules from a set of heterosynaptic primitives [4], [15], [25], [26] and introduced neuro-modulation [5], [8].

More precisely, [15] evolved the synaptic weights and the parameters A,B,C and D of heterosynaptic Hebbian rules (equation (1)) in a task which simulates bumblebees foraging for nectar. To add neuro-modulation, [27] and [8] evolved the parameters and the topology of neural networks to solve a T-maze and double T-maze problem in which the reward was alternatively placed in one of the branch. They proposed to distinguish two kinds of neurons: neuro-modulatory neurons and "standard" neurons. Using the Hebbian rule of equation (2), a neuro-modulatory neuron connected to a standard neuron will change the factor m, thus modulating or even de-activating the Hebbian learning rule. Hence, neuro-modulation allow to potentially stop synapse changes when an optimal behavior (according to the reward) is reached.

C. Behavioral diversity

Evolving plastic neural network raises a technical challenge for any evolutionary algorithm: in most situations, there exists a non-plastic (or non-adaptive) neural network that despite being non-optimal, solves a significant part of the task. Unfortunately, it is often impossible for the algorithm to add plastic synapses *a posteriori* without significantly degrading the fitness. From the optimization point of view, this makes most fitness functions that reward learning behaviors very deceptive [7], [8]. A direct consequence is that most fitness employed to evolve plastic neural network have to be precisely crafted to make the adaptive behaviors very attractive, whereas an ideal fitness function should be straightforwardly deduced from the task.

One of the most efficient method to find solutions with this type of fitness is to reward the exploration of the fitness landscape by rewarding novel behaviors, either by using a separate objective, or by removing the fitness objective entirely in favor of a novelty or diversity score. To do so, we need to be able to compute a distance between behaviors. We then use this distance as an objective to maximize. This can be done through the use of diversity where an individual is compared to its nearest neighbors in the population, or using an archive of the previous generations [9], [28], in which case we talk about novelty. These methods have successfully been applied to the problem of evolving adaptive neural networks by Stanley and Risi [29] and Soltoggio [7] using novelty of behavior.

D. Limits of previous papers

Many of the previously presented works use plasticity as a mean to solve tasks that change during the controller's lifetime [25]. However, it has been shown that fixed [30] or recursive [31] networks can often perform better than adaptive networks in these tasks, thanks to the internal dynamics created by recurrent connections. Additionally, the task investigated by these authors usually only requires a switch between two alternatives, both of them being available during evolution. By contrast, for many real tasks, we cannot test all the possible variations during the evolution because this would require too much time to evaluate each individual. Moreover, it can be argued that the long term goal of designing such plastic neural networks is to obtain neural networks which are able to cope with completely unknown situations, and not only with those which have been tried during the evolutionary process.

Other papers present adaptive networks as a solution for improving the robustness of the evolved controller. For example,

in [15], a network is tested in a simulation, and the solution is then implemented on a real robot. Nevertheless, even if some papers present good results [15], recursive neural networks can also perform better than adaptive networks in some of these situations [31], for example when reducing the sensory information available to the controller (reducing contrast).

Finally, many early works focused on evolving networks with a fixed topology [26] or relatively small neural networks, whereas most of the real-world problems require many effectors (e.g. legged robots) and complex sensory inputs (e.g. cameras). Hence, all the successful previous experiments involved from 1 [15], [27] to 3 [30] outputs and at most 8 inputs. Scaling up to more complex problems probably require to evolve modular and/or repetitive networks, which in turn need a complex encoding.

III. Map-based Encoding

A. Inspiration and Goals

An examination of the published neuroscience models [18]–[21], and especially of those that could be employed in artificial intelligence, allows to extract some regularities in the design of models. The most striking feature is the main building block: most models are based on $N \times M$ spatially organized grids of *identical* neurons, called *neural maps*, and not on individual neurons arbitrarily connected. Many models employ only maps of the same size, arbitrary fixed to the dimension of the input, and a few isolated neurons. Connection schemes between maps are of two main kinds, *one to one* connections with constant weights (neuron i of map M_1 is connected to neuron j of map M_2, with a positive weight identical for each connection) and *one to all* connections with constant weights (neuron i of map M_1 is connected to each neuron of map M_2, with identical weights for all connections).

Following the computational neuroscience practices, we opt here for maps as the basic building block to evolve neural networks. Our goal is threefold: (1) obtain neural network similar to those used in neuroscience models to use the latter as references or inspiration; (2) increase the scalability; and (3) improve the generalization properties of evolved plastic neural networks. Maps and regular connections schemes should indeed allow us to handle many similar inputs in the same way, thus making it easy to scale up to many inputs; encouraging preliminary results have already been obtained with a map-based encoding and many similar inputs [6].

Such regular networks should be able to evolve general learning rules because they must use the same modulation scheme for each neuron in a map. This makes map based neural networks better suited for situations that have not yet been encountered during the evolution process.

We chose to use a transfer function which projects to $[0:1]$ instead of the more classic $[-1:1]$ commonly used in machine learning. In this situation, a neuron with a strong negative input will output 0, and therefore have no effect on the post-synaptic neurons. This enables the neuromodulatory mechanisms to selectively enable or disable parts of network.

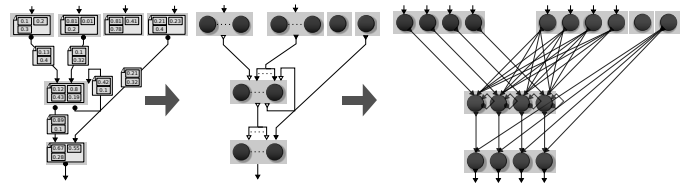


Fig. 1. Example of a controller genotype and phenotype using map encoding for our setup. The view on the left shows the genotype with numerical parameters, the center view shows an intermediate development, where maps are instantiated but not the connections. The right view shows the developed phenotype with maps and connections.

We further modified the links so that the sign of a connection cannot be changed by neuromodulation.

B. Hebbian Synaptic Changes

We opt here for the modulatory rule of equation 2 and used in [27] but we only set the parameter A at 1, while the four other parameters are set to 0; in effect we therefore use a modulated Hebbian rule. This choice was made because it led to the best convergence rate on preliminary experiments. Following [27] and [8], we distinguish two types of neurons: "standard neurons" and modulatory neurons; to use the modulation factor, inputs of each neuron are divided into modulatory inputs I_m (connections from modulatory neurons with activity o_j) and standard I_s inputs (inputs from standard neurons with activity p_j). The output a_i of a neuron i then defined as follows:

$$a_i = \varphi_1\Big(\sum_{j \in I_s} w_{ij} o_j + b_i\Big) \tag{3}$$

where i is the identifier of a neuron, a_i its output, b_i its bias, $\varphi_1(x)$ a sigmoid on $[0,1]$, w_{ij} the synaptic weight between neurons i and j.

Additionally, each synaptic weight w_{ij} is modified with regards to the sum of modulatory inputs and a constant coefficient η:

$$m_i = \varphi_2\Big(\sum_{j \in I_m} w_{ij} p_j\Big) \tag{4}$$

$$\Delta w_{ij} = \eta \cdot m_i \cdot a_i \cdot o_j \tag{5}$$

$$w_{ij}(t+\delta t) = \begin{cases} \min(\max(w_{ij}(t) + \Delta w_{ij}, 0), 30) \\ \text{if } w_{ij}(t) \geq 0 \\ \min(\max(w_{ij}(t) - \Delta w_{ij}, -30), -1e-5) \\ \text{if } w_{ij}(t) < 0 \end{cases} \tag{6}$$

At the beginning of each learning episode, non-modulatory synaptic weights are initialized with random values.

C. Evolving a Network of Maps

A network of neural maps can be efficiently represented with a labeled graph whose labels describe the properties of each map and of each connection (figure 1). Four parameters are associated to each vertex of the graph:

1) neural map / single neuron (Boolean);

2) modulatory neuron(s) / standard neuron(s) (Boolean);
3) bias of the activation function(s) (real value).

Similarly, each edge of the graph is labeled with two parameters:

1) synaptic weight.
2) connection type ("one to one" or "one to all");

The size of maps is not evolved; instead, it is a parameter of the experimental setup; as a result, the same graph can be interpreted for different map sizes.

Such graphs are evolved with a classic direct encoding, broadly inspired by the NEAT encoding [2]. Mutations act directly on the graph and cross-over is not employed; three mutations are possible:

- add/remove a node;
- add/remove/change a connection;
- mutate one or several labels.

No constraint is put to restrict the topology of the networks; in particular, recurrent connections are possible. More details about the map-based encoding can be obtained in [6]. At the first generation, the modulated weights in the network are initialized randomly

D. Control experiments: direct encoding

The same graph can be interpreted while ignoring the first label of nodes, resulting in a classic direct encoding for neural networks, without maps. The same mutations are employed.

IV. Experiments

A. Task Origin

To evaluate our approach, we selected a task based on a setup used to study operant conditioning with animals, commonly called a "Skinner box". In this type of experiment a caged animal, usually a mouse or a rat, is presented with stimuli which can be either sound, light or a combination of the previous elements. When the stimuli are presented, the animal must perform a specific action to obtain a reward. The typical action is the activation (push) of a lever and the reward is usually food or water. If the action is performed without the right stimulus, a punishment is given (electric shock). An example is shown in figure 2.

This task was selected for two main reasons. Firstly, it purposely does not involve delayed reward, a well known problem in reinforcement learning [32] which require more complex mechanisms than basic neuro-modulation [19], [33]. Secondly, this task presents a choice with many possible combinations for the answers. It is therefore different from a switch between only two alternatives, as done by most previous authors [8], [27]. Furthermore, as the number of stimuli and possible actions can be changed, the task combinatorial complexity can be arbitrarily increased or decreased. This feature will allow us to test the generalization abilities of the evolved networks.

B. Formalization

An agent in a skinner box can be formalized as system which associates an input vector of size N_s to a particular action among N_a possible actions. To simplify the use of neural maps, we consider here that $N_s = N_a$. The goal of the agent is to learn a set of association rules that links each possible N_s to an action that maximize its reward. To make the task easier, we restricted ourselves to vectors N_s in which only one input is active at a given time. Such a constraint corresponds to classical reinforcement learning setups where an input is equivalent to a state and in which two inputs cannot be active at the same time. The final network topology is shown in figure 2 (b).

Definitions

- *a rule* is an association between a neuron i from the input map of size N_s and a neuron j from the output map (which is of the same size N_s). To answer correctly to a rule, the neural controller must activate the output j when input i is presented.
- *a rule set* is a group containing N_s rules: one for each possible neuron in the input map. The same rule can be in two different rule sets, but two rule-set cannot be identical, they always have at least one rule different from one to the other.

Depending on N_s, there are $N_s^{N_s}$ possible rule sets. If $N_s = 4$, there are 256 rule-sets, which is different from the bumblebee foragging task with two choices.

To evaluate the performance of agent, the neural network is simulated with inputs which are not subject to random processes or noise other than the weight initialisation. This makes it difficult for the network to have an exploration behavior, whereas we know that this kind of exploration behavior is necessary to test the different solutions. The controller also cannot perform, multiple actions at once (if we take the analogy of the rat in a cage, it cannot push more than one lever at a time). The most elegant solution we found to solve both of these problems is to add a softmax mechanism on the outputs of the network. This softmax mechanism, described in [32] and given in equation 7, has a probability of selecting an output that is linked to the output level of this output compared to the others:

$$P(i) = \frac{\exp(\beta a_i)}{\sum_{i=1}^{n} \exp(\beta a_i)} \tag{7}$$

where $P(i)$ is the probability of selecting output i, a_i is the activity of output i and β is a fixed coefficient. From previous experiments, we also observed that if we use only a max function on the outputs instead of the softmax, the EA finds solutions where the difference between outputs is around $1e{-}2$ for a total range of 1. Using the softmax mechanism, we force the network to differentiate its outputs by a significant value.

The softmax mechanism selects an output based on both the output values and a random process, therefore, if we want the network to adapt, we must add inputs that provides to the network the decision of the softmax and the associated reward. The inputs of the networks can therefore be organised in three

(a) (b)

Fig. 2. (a) Example of a typical skinner box used in operant conditioning. (b) formalization using an evolved neural network setup.

groups. The first group contains the input pattern from the rule. If we use the map encoding, it is a map of size N_s, else, there are N_s separate inputs. The second group consists in the feedback information from the last softmax choice, these inputs are also either a map of size N_s or N_s separate inputs. Finally, there are two additional inputs which represent the positive and negative rewards. Both are single neurons. This setup is shown in figure 2.

The modulation rule we use (section III-B) can only modify a weight efficiently if the postsynaptic neuron activity is different from 0; else, the total modification is 0. This can lead to situations in which an adaptation mechanism relies on both an adaptation rule with specific initial parameters as well as a precise initial weight value. We need to avoid these situations, because they create unstable behaviors. To do so, the connections of the controller which are modulated are randomly initialised before learning each rule set, as shown in 3. The objective is to obtain network which are more robust to various weight values.

In our algorithm, the evaluation procedure of each controller is described on figure 3. Each controller is evaluated against a number of rule sets. Each rule sets consists in a N_s rules. For each rule sets, the modulated weights are randomly initialised. Then, each rule from the set is presented in a pseudo random order to the network. Each presentation is done in two parts. During the first part, the pattern is presented to the network while the other inputs are at O. After k_1 cycles, the output is read from the output and fed to the softmax mechanism, which selects the active output. The score of the network is then updated if it is a late trial. For the second phase, the same pattern is left on the first group of input, while the second group of input is fed the softmax output, which gives information to the network about which output was selected. At the same time, the reward input are set according to whether or not the selected output and the correct output match. After k_2 cycles, another rule is selected.

The neural network required to solve this may seem trivial at first sight. However, the simplest solutions using our simple building blocks require at least one hidden layer of neurons with non-trivial connectivity. The challenge raised by this problem is threefold: (1) identifying and correctly connecting the reward inputs, (2) gating the reward with the softmax choice to modify only the connections corresponding to the chosen action, and (3) applying the resulting reinforcement to a link between the inputs and the output. In classical reinforcement learning implementations, the softmax mechanism integration as well as the delay tuning between an action and the associated reinforcement are not implemented using neurons. This makes the problem simpler to solve.

C. Objectives

The fitness of each individual depends on the process shown in figure 3. For each controller, we record, at the end of each rule set training session, the number of rules the controller was able to answer correctly out of the total number of rules. This gives us the performance of the controller for a set of rule. When we compute the fitness score, we use the same set of rules as the one used during the evolution process. When we test the generalisation abilities of a network, we use either a rule set different from the training set (a rule set cannot be in both sets) or the complete set of possible rule sets (in this last situation, the rule sets used during evolution represent a small fraction of the total).

For the training sessions, we present all the rules in a predefined pseudo-random order. Eeach rule from a set is presented multiple times so that the probability to select at least once all the correct associations is superior to 0.99 for a network with all outputs at a zero value. This random probability is possible thanks to the softmax mechanism previously described.

As explained in II-C, using the behavioral differences to force the EA to explore can improve results with deceptive fitnesses. In the present setup, we describe the behavior of a network with a vector containing the post-learning output values of the network for each rule set. This both partially reduce the softmax mechanism impact on the measure and reduces the importance of the random initialisation of the modulated weights. The distance between behaviors is computed with an Euclidean distance. The diversity score of each individual is its distance from its 15 nearest neighbors. We use the diversity score as a second objective in a multi-objective EA. Objectives are shown in equation 8 where the first objective is the performance with δ being the Kronecker distance between the given (A_g) and correct answer (A_c) while k is the total number of rules tested. The second objective is the diversity, which is the sum of the distance between behaviors with respect to the 15 nearest neighbors.

$$\text{Maximize} \begin{cases} Fit(x) = \frac{1}{k}\sum_{i=1}^{k}\delta(A_g, A_c) \\ Div(x) = \sum_{j=1}^{15} d(B_x, B_j) \end{cases} \tag{8}$$

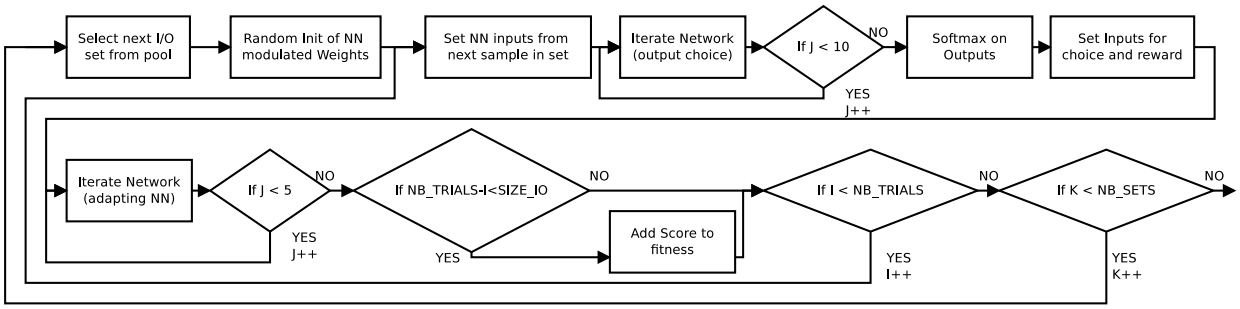


Fig. 3. Flowchart of the evaluation procedure.

D. Parameters

The evolved networks have $2 \cdot N_s + 2$ inputs, where N_s is the size of the map:

- a map of size N_s (N_s inputs with a direct encoding) providing the rule input pattern;
- a map of size N_s (N_s inputs with a direct encoding) providing the softmax choice previously done;
- a single neuron providing the positive reward;
- a single neuron providing the negative rewards.

The output is a map of size N_s (N_s outputs with a direct encoding). A sample network is shown in figure 1. Simple McCulloch and Pitts neurons are used. The modulation values are computed using a sigmoid projecting to $[-1 : 1]$. The sigmoid on the modulating inputs is used to normalize the modification inputs. If the input weights are sufficiently high. The sigmoid functions can be considered as a binary switch. Therefore, we allowed a maximum weight of 30 for our connections. In our problems, each map is a unidimensional list with a constant size. Therefore, the number of stimuli and possible actions is always the same. We used 3 or 4 elements in our experiments. For a map size of 3, we present 40 rules to each controller per rule-set and 90 for a size of 4. This number would allow a network with a constant output to select every correct rule at least once with a probability superior to 99 per cent (a constant output is a regular random when a softmax mechanism is used). The evolutionary algorithm we use is the state of the art NSGA-II [34] with 2 objectives. We used a population size of 400 and 2000 generations. We then use the median, as well as the upper and lower quartiles over 30 runs to have statistically significant results (the distribution of the results is not normal).

E. Setups and Expected Results

The first experiment tests whether or not maps perform better than direct encodings in setups where the number of similar inputs / outputs increases. Our setup was tested both with and without maps with an input / output size of 3 (8 inputs, 3 outputs) and 4 (10 inputs, 4 outputs). With a map size of 4, we further tested the importance of other parameters such as the modulation mechanism and the diversity. If our hypothesis is right, maps should perform better than direct encodings, especially for the bigger size. The second experiment tests if maps create structures which generalize better than direct encodings when the fitness on difficult setups: the evolution fitness is computed on a few examples only. We evolve controllers with a fitness containing 1 to 5 rule tests, while the usual number is 10. The best controllers from each run are then tested against all the possible rule sets. If our hypothesis is correct, maps should have better performance than a direct encoding on the complete sets even when evolved on a few rule sets.

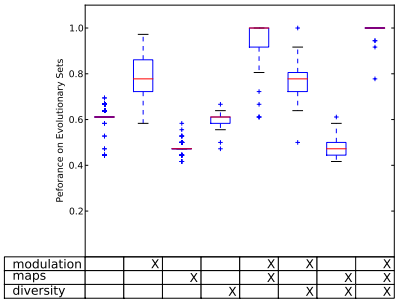


Fig. 5. Test of the impact of different evolutionary parameters on the EA performance on a problem with 4 inputs and 4 outputs. The values are the median (red bar), the box extends from the lower to the upper quartile and the whiskers show the range of the data over 30 runs.

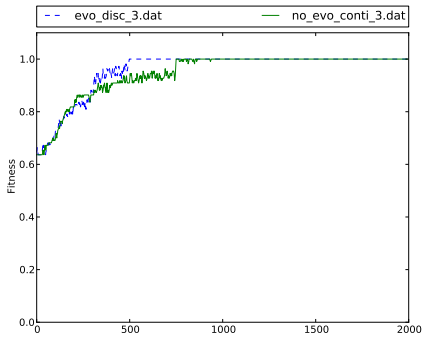
V. Results

A. Increase of fitness when using maps

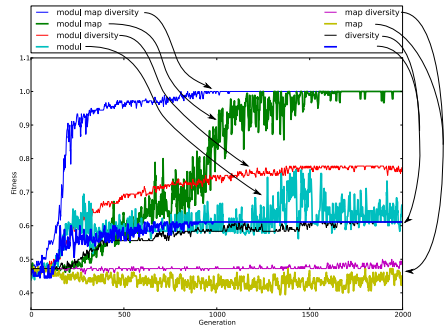
Left part (a) of figure 4 shows that, with a 3 inputs and 3 outputs, both the direct encoding and the map encoding provide equivalent results. The reason may be that while the map encoding simplifies the target network topology, it also adds multiple parameters to the genotype, which in turn increase the search space. This could explain why the results are similar.

By contrast, the results in part (b) of figure 4 show that, in a problem with a 4 inputs and 4 outputs, a map-based encoding is required to obtain the perfect fitness; with a direct encoding, the best individual can only learn 80 per cent of all rules.

As medians do not provide all the possible information, the additional information provided by figure 5 shows that a single run managed to reach the maximum fitness without maps. This proves that our generative encoding scales-up better than a basic direct encoding.



(a)



(b)

Fig. 4. Fitnesses values for the best individuals for each generation on a 3 inputs / 3 outputs (a) or 4 inputs/ 4 outputs (b) problem with and without the map encoding. for a map size of $N_s = 4$, other parameters were tested. The values are medians over 30 runs.

Figure 5 also shows that a diversity objective speeds up the search for solutions. These results are similar to the ones obtained by Risi [29] and Soltoggio [7]. In our experiment, using diversity enabled us to half the number of generation necessary to converge in many variants of these experiments.

Finally, these results show that a modulation rule is necessary to solve this task. We further show that maps without modulation have lower fitnesses (compared to a direct encoding without modulation) in this task where differentiation of the outputs is necessary.

B. Better generalisation when maps are used

Figures 6(a) show that the map-based encoding performs better than the direct encoding when tested on all the rule sets (most of which are not used during the evolutionary process). The better fitness of map encodings can be explained in two ways. The first way is that both encodings manage obtain similar maximum fitnesses during evolution. In this situation, it means that the direct encoding creates overlearning networks, which are performing well only on the training examples, while the maps create networks which are able to generalize and learn rule sets different from the ones used during training. This would confirm our hypothesis. The second solution is that the generative encoding always outperforms the direct encoding. The difference then comes only from the difference in the fitness between generative encoding and direct encoding.

To show that both encodings generate solutions, we considered only the networks which reach a fitness of 1 (all training rule sets perfectly learned). These results are shown on figure 6(b). In this situation, we show that the direct encoding only finds solutions when the evolution tests a few rule sets. In these situations, The map based encoding offer significantly better generalisation results. This confirms our hypothesis that using a map based encoding only generates networks which generalize better.

When we consider only the generative encoding, we see that we cannot differentiate results for two or more training rule sets. Though, when only one rule set is used during evolution, the generalisation score is low. After analysis, some networks were able to generate a fixed pattern with different values for each output of the map. This was done by exploiting the constant pseudo random order in which the patterns are presented which is always the same.

CONCLUSION

This paper investigated the combination of Hebbian learning, neuro-modulation and a neuroscience-inspired map-based encoding to evolve adaptive neural networks. We applied the proposed approach on a toy problem inspired from a classic operant conditioning setup in which numerous different association rules can be learned.

Results show that the map-based encoding scaled up better than a classic direct encoding on this task. With few inputs, our generative encoding led to similar performance as a direct encoding. However, the best networks evolved with the direct encoding have lower fitnesses if we increase the number of inputs / outputs, while the map based encoding performance does not change depending on the number of inputs / outputs. Evolving neural networks using a map based generative encoding also led to networks that can learn most rule sets even when the evolution is done on a small subset of all the possible cases. Such a generative encoding therefore appears as a key to improve the generalization abilities of evolved adaptive neural networks.

While the present setup was a toy example, it can be assimilated to the actor in an *actor-critic* architecture. We hope to be able to evolve such an architecture in future work because it's both powerful in machine learning and has similarities to what can be observed in animals. Nevertheless, such an achievement will most probably require the addition of more advanced mechanisms, for instance based on eligibility traces [35], [36].

ACKNOWLEDGMENT

This project was funded by the ANR EvoNeuro project (ANR-09-EMER-005-01) and a Monaco Ph.D. scholarship.

(a) (b)

Fig. 6. Scores on unknown datasets with (thick lines) and without (thin lines) maps with respect to the number of different examples used during evolution (4 inputs / 4 outputs). (a) Scores for all the runs. (b) Scores for the run that converged. The values are the median (red bar), the box extends from the lower to the upper quartile and the whiskers show the range of the data over 30 runs.

REFERENCES

[1] J. Meyer, S. Doncieux, D. Filliat, and A. Guillot, "Evolutionary approaches to neural control of rolling, walking, swimming and flying animats or robots," *Studies in Fuzziness and Soft Computing*, vol. 109, pp. 1–44, 2003.

[2] K. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," Department of Computer Sciences,University of Texas, Tech. Rep. AI2001-290, 2002.

[3] D. Floreano and C. Mattiussi, *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT Press, 2008.

[4] J. Urzelai and D. Floreano, "Evolution of Adaptive Synapses : Robots with Fast Adaptive Behavior in New Environments," *Evolutionary Computation*, vol. 9, no. 4, pp. 495–524, 2001.

[5] A. Soltoggio, P. Dürr, C. Mattiussi, and D. Floreano, "Evolving neuromodulatory topologies for reinforcement learning-like problems," *Proceedings of the IEEE*, 2007.

[6] J.-B. Mouret, S. Doncieux, and B. Girard, "Importing the computational neuroscience toolbox into neuro-evolution—application to basal ganglia," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation ACM, publisher*, 2010.

[7] A. Soltoggio and B. Jones, "Novelty of behaviour as a basis for the neuro-evolution of operant reward learning," *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, p. 169, 2009.

[8] S. Risi, C. Hughes, and K. Stanley, "Evolving Plastic Neural Networks with Novelty Search," *Adaptive Behavior*, 2010.

[9] J.-B. Mouret and S. Doncieux, "Using Behavioral Exploration Objectives to Solve Deceptive Problems in Neuro-evolution," in *Proceedings of the 11th annual conference on Genetic and Evolutionary Computation*. ACM, 2009, pp. 627–634.

[10] B. Skinner, "The experimental analysis of operant behavior," *Annals of the New York Academy of Sciences*, vol. 291, no. 1, pp. 374–385, 1977.

[11] L. F. Abbott and W. G. Regehr, "Synaptic computation." *Nature*, vol. 431, no. 7010, pp. 796–803, Oct. 2004.

[12] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: From Architectures to Learning," *Evolutionary Intelligence*, pp. 47–62, 2008.

[13] D. Hebb, *The organization of behavior: A neuropsychological theory*. Lawrence Erlbaum, 1949.

[14] J. Rauschecker and W. Singer, "Changes in the circuitry of the kitten visual cortex are gated by postsynaptic activity," 1979.

[15] Y. Niv, D. Joel, I. Meilijson, and E. Ruppin, "Evolution of Reinforcement Learning in Uncertain Environments : A Simple Explanation for Complex Foraging Behaviors," *Adaptive Behavior*, 2002.

[16] C. H. Bailey, M. Giustetto, Y.-y. Huang, R. D. Hawkins, and E. R. Kandel, "Is Heterosynaptic Modulation Essential for Stabilizing Hebbian Plasticity and Memory ?" *Neuroscience*, vol. 1, no. October, pp. 1–10, 2000.

[17] A. Barto, "Adaptive Critics and the Basal Ganglia," *Models of information processing in the basal ganglia*, p. 215, 1994.

[18] B. Girard, N. Tabareau, J.-J. Slotine, and A. Berthoz, "Contracting model of the basal ganglia," in *Modelling Natural Action Selection: Proceedings of an International Workshop*, 2005, pp. 69–76.

[19] G. Baldassarre, "A Modular Neural-Network Model of the Basal Ganglia's Role in Learning and Selecting Motor Behaviours," *Cognitive Systems Research*, vol. 3, no. 1, pp. 5–13, Mar. 2002.

[20] A. Morrison, "A Spiking Neural Network Model of an Actor-Critic," *Neural Computation*, vol. 339, pp. 301–339, 2009.

[21] W. Potjans, A. Morrison, and M. Diesmann, "A Spiking Neural NetworkModel of an Actor-Critic Learning Agent," *Neural Computation*, vol. 339, pp. 301–339, 2009.

[22] X. Yao, "Evolving Artificial Neural Networks," in *Proceedings of the IEEE*, vol. 14, no. 3. IEEE, Nov. 1999, pp. 1423–1447.

[23] N. Kasabov, "ECOS: Evolving connectionist systems and the ECO learning paradigm," *Proc. ICONIP*, no. 1, pp. 1232–1235, 1998.

[24] E. A. Di Paolo, "Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions," in *Proceedings of the 6th International Conference on the Simulation of Adaptive Behavior*, 2000, pp. 440–449.

[25] J. Blynel and D. Floreano, "Levels of dynamics and adaptive behavior in evolutionary neural controllers," *conference on simulation of adaptive behavior on*, no. 1994, 2002.

[26] D. Floreano, "Evolution of plastic neurocontrollers for situated agents," *From animals to animats*, 1996.

[27] A. Soltoggio, J. A. Bullinaria, C. Mattiussi, P. Dürr, and D. Floreano, "Evolutionary Advantages of Neuromodulated Plasticity in Dynamic, Reward-based Scenarios," *Artificial Life XI*, vol. 11, 2008.

[28] J. Lehman and K. Stanley, "Exploiting Open-Endedness to Solve Problems Through the Search for Novelty," in *Proceedings of the 11th annual conference on Aritifical Life*, 2008.

[29] S. Risi, S. D. Vanderbleek, C. E. Hughes, and K. Stanley, "How Novelty Search Escapes the Deceptive Trap of Learning to Learn Categories and Subject Descriptors," *Evolutionary Computation*, 2009.

[30] K. Stanley, B. D. Bryant, and R. Miikkulainen, "Evolving adaptive neural networks with and without adaptive synapses," in *The 2003 Congress on Evolutionary Computation, CEC'03*, 2003, pp. 2557–2564.

[31] E. Tuci and M. Quinn, "Behavioral Plasticity in Autonomous Agents: a Comparison Between Two Types of Controller," *Proceedings of the 2003 international conference on Applications of evolutionary computing*, pp. 661–672, 2003.

[32] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. The MIT press, 1998.

[33] H. Kimura and S. Kobayashi, "An Analysis of Actor-Critic Algorithms Using Eligibility Traces: Reinforcement Learning with Imperfect Value Functions," *-JAPANESE SOCIETY FOR*, 2000.

[34] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[35] W.-X. Pan, R. Schmidt, J. R. Wickens, and B. I. Hyland, "Dopamine cells respond to predicted events during classical conditioning: evidence for eligibility traces in the reward-learning network." *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 25, no. 26, pp. 6235–42, Jun. 2005.

[36] K. Doya, "Reinforcement learning in continuous time and space." *Neural computation*, vol. 12, no. 1, pp. 219–45, Jan. 2000.