AUTOMATIC SYNTHESIS OF WORKING MEMORY NEURAL NETWORKS WITH NEUROEVOLUTION METHODS

Tony Pinville
ISIR, CNRS UMR 7222
Université Pierre et Marie Curie-Paris 6
4 place Jussieu, F-75252
Paris Cedex 05, France
email: pinville@isir.upmc.fr

Stéphane Doncieux
ISIR, CNRS UMR 7222
Université Pierre et Marie Curie-Paris 6
4 place Jussieu, F-75252
Paris Cedex 05, France
email: doncieux@isir.upmc.fr

ABSTRACT

Evolutionary Robotics is a research field focused on autonomous design of robots based on evolutionary algorithms. In this field, neuroevolution methods aim in particular at designing both structure and parameters of neural networks that make a robot exhibit a desired behavior. While such methods have shown their efficiency to generate reactive behaviors, they hardly scale to more cognitive behaviors. One of the reasons of such a limitation might be in the properties of the encoding, i.e. the neural network representation explored by the genetic operators. This work considers *EvoNeuro* encoding, an encoding directly inspired from computational neuroscience [1] and tests its efficiency on a working memory task, namely the AX-CPT task. Neural networks able to solve this task are generated and compared to neural networks evolved with a simpler direct encoding. The task is solved in both cases, but while direct encodings tend to generate networks whose structure is adapted to a particular instance of AX-CPT, networks generated with EvoNeuro encoding are more versatile and can adapt to the new task through a simple parameter optimization. Such versatile neural network encoding might facilitate the evolution of robot controllers for tasks requiring a working memory.

KEY WORDS
evolutionary algorithms; neural networks; computational neuroscience; working memory; neuroevolution.

1 Introduction

Evolutionary algorithms are stochastic algorithms based on natural evolution. In the same way as nature uses the principle of the "survival of the fittest" to improve the overall quality of the individuals in a population over a long time, evolutionary algorithms explore a search space and give solutions with the better fitness a higher probability to survive and generate siblings [2]. While such algorithms are frequently used to optimize parameters, neuroevolution methods use them to synthetize artificial neural networks that achieve a task described by a high level fitness function (fitness is the name of the function to optimize). It is used in particular in Evolutionary Robotics to make real or simulated robots exhibit a desired behavior [3].

Most evolutionary algorithms optimize a fixed size genotype, whereas neuroevolution methods aim at exploring in both parameter and structure space, in the search for neural networks able to achieve a given task. Exploring in the structure space requires to define an encoding, i.e. a representation of a neural network with its dedicated search operators. Typically, the mutation operator can individually add or delete neurons or connections [4], but this leads to networks with no particular regularity in their structure.

While Evolutionary Robotics up to now mainly deals with reactive behaviors, in this work we use this method to obtain working memory neural networks, that can be considered as a prerequisite to more cognitive behaviors. To our knowledge, despite the fact that working memory is a critical brain function, few works have used neuroevolution method to build a working memory model and implement this mechanism in a robotic agent [5].

Defined as "the ability to transiently hold and manipulate goal-related information to guide forthcoming actions" [6], working memory modeling is a central area of research in computational neuroscience. Lots of models have been built at different levels of abstraction, from low level [7] to highly abstract connectionist models [8, 9]. Multiple tasks have been defined to test working memory abilities: Delayed-Response tasks (DR) [10], Delayed Matching-to-Sample Tasks (DMS) [11], Ocular Delayed-Response Tasks (ODR) [12], Vibrotactile Discrimination Task [7], Stroop task [13] or AX-CPT [8, 14, 9]. This last task has particularly caught our attention, because it provides a relatively specific probe of goal representation, maintenance and updating [15] and has been modeled with highly abstract connectionist models [8, 9].

Despite computational neuroscience and neuroevolution both focus on neural networks, evolved neural networks present few similarities with models produced in computational neuroscience. The main difference is based on the fact that evolutionary methods mostly use individual neurons, sometimes organized in modular fashion, whereas neuroscience models rely on much more structured networks.

Noticing that biological systems are often based on the repetition and combination of hierarchically organized modules, several researchers proposed to define encodings with some of these abilities [16, 17, 18]. The EvoNeuro encoding [1], used in this work, directly draws inspiration from computational neuroscience, and includes structure primitives like neural maps, for instance. This encoding has been tested to automatically generate neural networks exhibiting the action-selection behavior of basal ganglia [1]. Results have shown that this encoding easily achieves this task, while a basic encoding never solves it. These encouraging results lead us to consider other basic abilities of the brain such as working memory.

Two main points are argued here:

- Neuroevolution can automatically generate neural networks with a working memory functionality;

- EvoNeuro encoding generates more versatile neural networks than a simpler direct encoding.

2 AX-CPT task

AX-CPT task is a modified version of the classic Continuous Performance Test (CPT) [19]. Introduced by Braver [8], this paradigm has become a standard benchmark to study syndroms thought to involve prefrontal cortex dysfunction such as schizophrenia [14] or to evaluate aging effect on performance [9].

The task is the following: during each AX-CPT trial, participants are presented with a sequence of stimuli containing a context cue (stimulus A or B) and a probe (X or Y) on the computer screen. They have to respond to a target probe (X) with a manual response on the keyboard, the target response key, but only when the target probe is immediately preceded by a specific context cue (A). In every other case, for example in AY, BX or BY sequences, they have to respond to a probe with a nontarget response key. AX trials occur very frequently during the experiment to induce a strong tendency to make a target response to the X-probe. A key aspect of the task is that in some trial conditions (termed BX), the contextual information must be used to inhibit a dominant response tendency, whereas in other trials (termed AY) context serves an attentional biasing function. This task requires a relatively simple form of working memory, where the prior stimulus must be maintained over a delay until the next stimulus appears, so that the subject can discriminate the target from non-target sequences.

Several high level computational models [8, 14, 9] have been created to make novel and testable predictions regarding the behavioral performance of the subjects.

The first model [8] is a simple model of the prefrontal cortex based on two information processing roles for the PFC: short-term active memory and inhibition. Following models [14, 9] are trying to define a model of cognitive control which simulates system interactions between PFC and dopamine (DA).

On the other hand, Frank and O'Reilly [20, 21] propose a more biologically plausible model, the PBWM (for Prefrontal-cortex, Basal-Ganglia Working memory Model). It is based on the postulate that the basal ganglia provides a selective dynamic gating mechanism for information maintained via sustained activation in the PFC. A wide variety of working memory tasks have been tested on this model like the Stroop effect, the AX-CPT, the 1-2-AX or the Wisconsin card sort task [20].

3 EvoNeuro encoding

The simplest encoding in neuroevolution is direct encoding, in this case the genotype is the same as the phenotype. Here we evolves a labeled graph which can be modified structurally (add/remove a connection or a node) and parametrically (change of a label) with an evolutionary algorithm. The graph is represented as a classic adjacency list where cross-over is not used and mutation operators can: (1) add a node on an existent connection, with random labels; the connection is split in two and the two parts keep the same labels; (2) remove a random node and its associated connections; (3) add/remove a connection between two random nodes. Nodes and connections can be labeled by a list of real parameters that represent weights, threshold, neuron type... These parameters are mutated using polynomial mutation [22]. Each node describes a neuron and the labels define then neuron parameters (time constant, threshold, inhibitory status). The connections are labeled with a single real number interpreted as the synaptic weight.

The Evoneuro encoding (figure 1) uses the same principle and adds two building blocks taken from computational neuroscience models: (1) map of neurons, (2) connection schemes between neural maps. Maps are defined as spatially organized grids of *identical* neurons (same time constant, same threshold, same inhibitory status). Connection schemes between maps are restricted to three cases: (1) one to one connection 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), (2) 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) and (3) one to all connections with weights following a Gaussian distribution. As in [1], we use a lPDS-based (locally Projected Dynamic System) neuron model [23] which is a variant of the classic leaky integrator with similar dynamics but which verifies the dynamic property of contraction [23]. See [1] for a detailled description of EvoNeuro encoding.

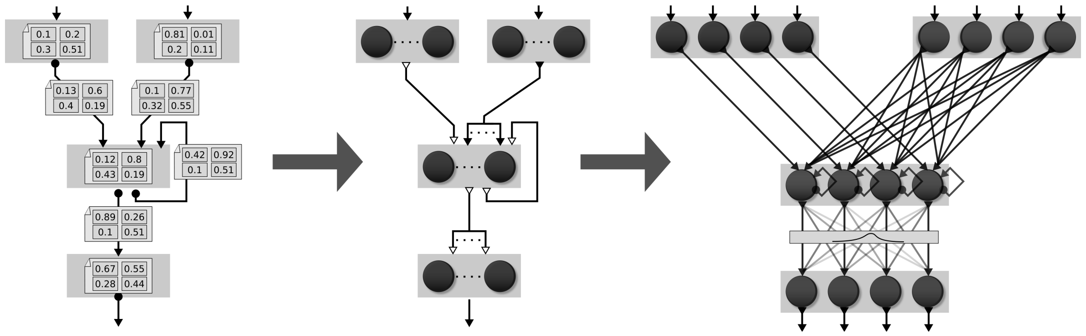

Figure 1. Overview of the development process. From left to right: (1) the genotype is a labeled graph with evolvable labels; (2) the labels are interpreted to a neuroscience-inspired description of the neural network; (3) for a given size of maps, this neural network can be fully developed into a neural network (for instance to evaluate its fitness).

4 Experiments

4.1 AX-CPT

Here we have used the rules of classic AX-CPT task [8] which consists of randomly cue/probe presentations with the following constraints: (1) Target trials: A followed by X occurs 70 % of the time (to probe the inhibitory function of PFC); (2) A cue followed by a non-target probe letter (A-Y) 10% of the time; (3) A non-cue followed by a target probe letter (B-X) 10% of the time; (4) A non-cue followed by a non-target probe letter (A-Y) 10% of the time;

A letter's presentation corresponds to a number randomly chosen in $]0.9, 1[$ for the corresponding input and zero on the other ones. The neural network has thus four inputs, one per letter, and two outputs. The response of the network is considered to be "non-target" if the first output is greater than the second and "target" otherwise.

4.2 Fitness

In the remaining text, the following notations are used:

- x: a developed individual (a neural network);
- k: number of inputs; ($k = 4$, one for each letter);
- $\mathbf{v}$: vector of input letters ($\mathbf{v} \in [0, 1]^k$);
- T: the maximum simulation duration (T = 1000);
- T_c: the end of the simulation;
- $\gamma(x, \mathbf{v}, t)_i$: activation level of the output neuron i ($i \in \{1, 2\}$) at time t ($t \in [0, T]$);
- $\gamma_c(x, \mathbf{v})_i$: activation level of the output neuron i ($i \in \{1, 2\}$) at the end of the simulation (i.e. $t = T_c(x, \mathbf{v})$).

For each individual, a sequence of N letters are presented, each letter's presentation is simulated until its output converges to a constant vector or until it reaches the maximum number of time-steps ($t = T$). From a practical viewpoint, a neural network is considered to have converged when S (with S randomly chosen in $\{10, 20, \cdots, 100\}$) successive outputs have a difference of less than ε (in these experiments, $\varepsilon = 10^{-6}$). S is variable, because preliminary results have shown us that with a constant S (i.e. $S = 10$), feed-forward networks tuned with a high accuracy are generated most of the time. They relied on neuron dynamics rather than recurrent connections to fulfill the task, thus exhibiting a non-robust memory behavior specialized for a particular value of S.

To compute T_c, we first define the "convergence function" $K(x, t, \mathbf{v})$:

$$K(x, t, \mathbf{v}) = \begin{cases} 0 \text{ if } \left|\gamma(x, \mathbf{v}, t)_i - \gamma(x, \mathbf{v}, t-n)_i\right| < \varepsilon, \\ \quad \forall n \in \{1, \cdots, S\}, \forall i \in \{1, 2\} \\ 1 \text{ otherwise} \end{cases} \tag{1}$$

T_c can now be defined as $T_c(x, \mathbf{v}) = t$ with $K(x, t, \mathbf{v}) = 0$ and $K(x, t', \mathbf{v}) = 1$ for $t' < t$.

The main objective function (fitness) aims at checking that the network answers the correct response for any given $\mathbf{v}$. Furthermore, we are interested to have the biggest contrast between the 2 outputs.

In our case, arbitrary $\gamma_c(x, \mathbf{v})_1$ correspond to the non-target response, whereas $\gamma_c(x, \mathbf{v})_2$ is the target response. Let define $R(x, \mathbf{v})$ for the response of the network:

$$R(x, \mathbf{v}) = \begin{cases} 0 \text{ "Target" if } (\gamma_c(x, \mathbf{v})_2 - \gamma_c(x, \mathbf{v})_1) > 0 \\ 1 \text{ "Non-Target" otherwise} \end{cases} \tag{2}$$

Now we can compare the network response with the expected response:

$$E(x, \mathbf{v}) = \begin{cases} 0 \text{ if } (R(x, \mathbf{v}) = Q(x, \mathbf{v})) \\ -1000 \text{ otherwise} \end{cases} \tag{3}$$

where $Q(x, \mathbf{v})$ is the expected response and $E(x, \mathbf{v})$ the evaluation note. We test also the discrimination $D(x, \mathbf{v})$

Table 1. Parameters used in experiment 1 (with map-based encoding and direct encoding).

Parameter /Genotype	Map-based	Direct enc.
min./max. nb. of nodes (rand.gen.)	1 / 5	4 / 20
min/max. nb. of links (rand. gen.)	1 / 5	4 / 20
prob. to add/remove a node	0.05 / 0.05	0.05 / 0.05
prob. to add/remove link	0.05 / 0.05	0.05 / 0.05
prob. to change each label	0.1	0.1
σ for gaussian mutation	0.05	0.05

between the two outputs:

$$D(x, \mathbf{v}) = \begin{cases} 0 \text{ if } (\|\gamma_c(x, \mathbf{v})_2 - \gamma_c(x, \mathbf{v})_1\| > 0.8) \\ -1 \text{ if } (\|\gamma_c(x, \mathbf{v})_2 - \gamma_c(x, \mathbf{v})_1\| > 0.3) \\ -2 \text{ otherwise} \end{cases} \tag{4}$$

Let I be a set of N letters vectors. So the fitness to maximise is:

$$F(x) = \sum_{\mathbf{v} \in I} (D(x, \mathbf{v}) + E(x, \mathbf{v})) \tag{5}$$

The maximum value is 0 which indicates that the network has solved the problem without any error. In these experiments, N was fixed to $N = 1000$ and the same vectors were employed to evaluate all individuals.

The search is restricted to networks that converge during the simulation time and where activation level on output is positive. The first constraint $C_1(x)$ ensures that $\gamma_c(x, \mathbf{v})_i$ is strictly positive:

$$C_1(x) = \begin{cases} 0 \text{ if } \gamma_c(x, \mathbf{v})_i > 0 \\ 1 \text{ otherwise} \end{cases} \tag{6}$$

The second constraint, $C_2(x)$ checks that the tested neural network converges to a constant output vector before the end of the experiment, for all the tests performed on the neural network:

$$C_2(x) = \sum_{\mathbf{v} \in I} (K(x, T_c, \mathbf{v})) \tag{7}$$

These constraints are enforced with the penalty method [22]: an arbitrary large penalty is added to the fitness each time a constraint is violated. Instead of maximizing $F(x)$, we thus maximize $F_c(x)$:

$$F_c(x) = -K(C_1(x) + C_2(x)) + F(x) \tag{8}$$

where K is an arbitrary large constant (e.g.10^{10}).

4.3 Experimental setup

Our goal is to obtain a versatile working memory neural network. Although not included in this work, future work will consider learning abilities to adapt the behavior of the network online. For a neuroevolution method, it is not difficult to connect only the inputs corresponding to the interesting letters while ignoring the others. For such a result, changing the features of the task – for instance inverting the role of the letters – implies structural changes and can't thus be done with an online learning algorithm. Likewise, in an evolutionary setup, if the working memory module is only a part of a more complex neural network controller, the evolutionary search is expected to face more difficulties when complex structural changes are required to adapt memory module behavior.

Our objective is then twoflold: (1) obtain a network topology able to perform an AX-CPT task; (2) test this topology on a different instance of this task with synaptic weights changes only to check its versatility. The fitness function of the first step may take into account the versatility, but this would require a complex evaluation process. We have then chosen to test *a posteriori* the versatility.

For (1) we use EvoNeuro encoding to evolve network structures and parameters. As a control experiment, neural networks are evolved with a simple direct encoding in which mutation directly adds, removes neurons or connections or changes weights, as in [1]. 10 independent evolutionary runs, with a budget of 400,000 evaluations each (2000 generations with a population of 200), have been performed for each experimental setup. In a second step, we test the stability of the networks obtained. In these experiments, a network is considered to stably converge to an equilibrium state when 200 successive outputs (instead of $S = 10$ previously) have a difference of less than ε (here, $\varepsilon = 10^{-6}$).

For (2) the best evolved networks of experiment (1) are kept and their weights only are evolved with BY as a target sequence instead of AX. One run is performed for each evolved structure, there are then 10 parameter optimization runs with networks generated by EvoNeuro encoding and 10 more optimization runs with networks generated by the direct encoding. We have chosen to use the same evolutionary algorithm with a constant structure and a fixed number of parameters, but the parameters may have been optimized by other optimization algorithms.

The same evolutionary algorithm, the same fitness and the same model of neurons were employed in all experiments; only the genotype/phenotype mapping was changed. The chosen evolutionary algorithm is a single-objective implementation of NSGA-2 [22], an elitist tournament-based evolutionary algorithm. The framework used to run all these experiments is Sferes2 [24]. Parameters are provided in table 2 and the source code is available at: `http://www.isir.fr/evorob_db`.

5 Results

For experiment (1), with direct encoding, 9 of 10 runs find an optimal solution with an average of 656 generations (131,200 evalutions). For step 2, when we check neural network stability, results shows that only 4 of 10 networks have a constant output after 200 steps.

The EvoNeuro encoding finds an optimal solution for 8 of 10 runs, within 1016 generations on average (203,200

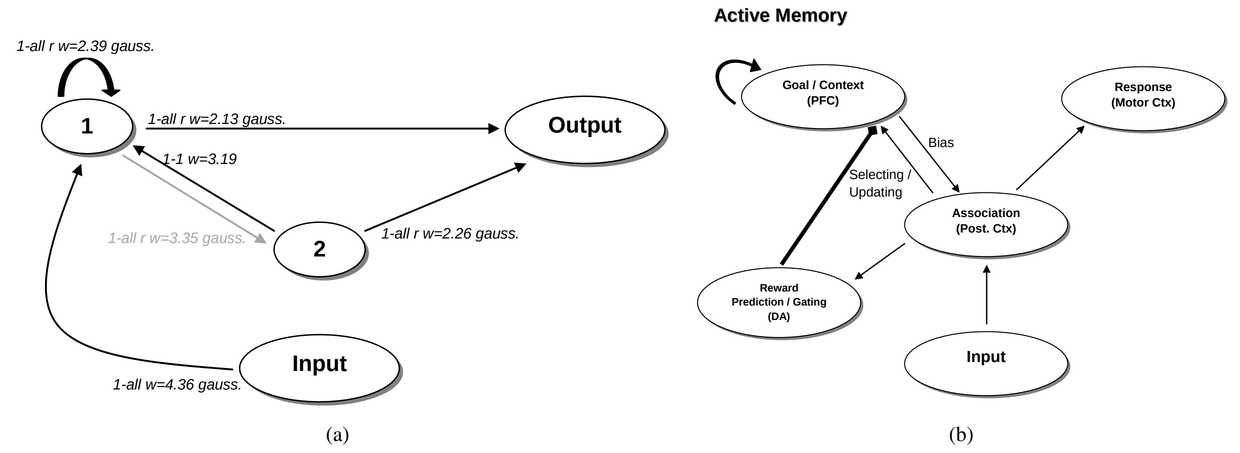

Figure 2. (a) Example of a module obtained with map-based neuroevolution. In this case, each map is composed of 4 neurons. 1-1 represents one to one connections between maps, 1-all, one to all connections, gauss., weights following a gaussian distribution; (b) Minimalist canonical model of cognitive control proposed by Braver [9]

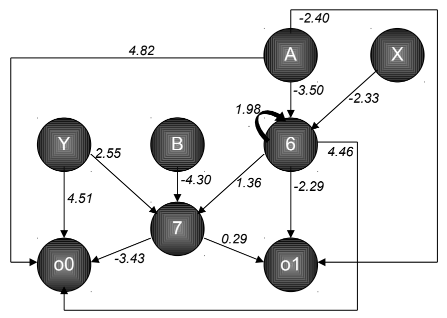

Figure 3. Minimalist neural network obtained by direct encoding. In this case, each circle represents one neuron.

evaluations). 5 of 10 networks perform the stability test. The two setups show then similar results..

But surprisingly unlike direct encodings (figure 3), some generated networks present interesting features. As shown in figure 2(a), the evolved network shares some similarities with existing models of cognitive control. We can identify a topology close to the simple canonical model proposed by Braver and represented in figure 2(b) [9]. 3 key computational principles of context processing mechanism are defined: (1) active memory through recurrent connections; (2) top-down bias through feedback connections ; and (3) regulated access of contextual input through modulatory gating connections. We can find the same active memory through recurrent connections in our model (map 1), the feedback connection between PFC and posterior cortex could be seen in the connections between map 1 and map 2. The reward prediction is not present in our network, indeed in Braver's model the timing of gating signals is learned through a reward prediction learning mechanism, whereas in our model no learning mechanism is included (and the fitness function doesn't enforce learning).

For experiment (2), we have tested every networks which has performed experiment (1) (4 with direct-encodings, 5 with map-based encoding). With classic direct-encoding, none of the 4 networks can be adapted to the new target sequence with connection weights changes only, whereas with Evoneuro encoding 4 of 5 networks are able to perform the task after an optimization of the weights (with an average of 462 generations). These results confirm our hypothesis that neural networks evolved with EvoNeuro encoding, are more versatile than neural networks obtained with a simple direct encoding.

6 Conclusions and future work

Our experiments have shown that with the help of several computational neuroscience building blocks (leaky integrator neurons, map of neurons, projection schemes), neuroevolution can build simple working memory models. Indeed our generated networks have successfully performed a simple, but very common in human behavior studies, working memory task: AX-CPT. In the second step we have proved that our map-based generated networks are more versatile than those generated by a simpler neuroevolution method (with direct encoding)

Our future work will be to test the framework on different and more complex working memory tasks like 12-AX task which is an extension of the AX-CPT task [20, 21], or the Stroop task [13] to simulate multiple working memory tasks in a single model like the PBWM model [20, 21] then test such models in the context of robot behavior design. An other objective will be to implement a learning mechanism on evolved neural networks to solve different tasks with keeping the same topology.

7 Acknowledgement

This project was funded by the ANR EvoNeuro project, ANR-09-EMER-005-01.

References

[1] J.-B. Mouret, S. Doncieux, and B. Girard. Importing the Computational Neuroscience Toolbox into Neuro-Evolution—Application to Basal Ganglia. In *Proc. of GECCO*, 2010.

[2] A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*. Springer, 2003.

[3] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Press, MIT, 2000.

[4] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99127, 2002.

[5] T. Ziemke and M. Thieme. Neuromodulation of Reactive Sensorimotor Mappings as a Short-Term Memory Mechanism in Delayed Response Tasks. *Adapt. Behav.*, 10(3-4), July 2002.

[6] D. Durstewitz, J. K. Seamans, and T. J. Sejnowski. Neurocomputational models of working memory. *Nat. neurosci.*, 3 Suppl:1184–1191, November 2000.

[7] C. K. Machens, R. Romo, and C. D. Brody. Flexible control of mutual inhibition. *Science*, 307(5712):1121–4, 2005.

[8] T. S. Braver, J. D. Cohen, and D. Servan-Schreiber. A computational model of prefrontal cortex function. *Nips*, page 141148, 1995.

[9] T. S. Braver and D. M. Barch. A theory of cognitive control, aging cognition, and neuromodulation. *Neurosci. biobehav. r.*, 26(7):809–17, November 2002.

[10] D. Zipser. Recurrent Network Model of the Neural Mechanism of Short-Term Active Memory. *Neural comput.*, 3(2):179–193, June 1991.

[11] T. Gisiger, M. Kerszberg, and J. P. Changeux. Acquisition and Performance of Delayed-response Tasks: a Neural Network Model. *Cereb. Cortex*, 15(5):489–506, May 2005.

[12] J Mitchell and D. Zipser. Sequential memory-guided saccades and target selection. *Vision Res.*, 43(25):2669–2695, 2003.

[13] M M Botvinick, T. S. Braver, D. M. Barch, C. S. Carter, and J. D. Cohen. Conflict monitoring and cognitive control. *Psychol. rev.*, 108(3):624–52, July 2001.

[14] T. S. Braver, D. M. Barch, and J. D. Cohen. Cognition and control in schizophrenia: a computational model of dopamine and prefrontal function. *Biol. psychiat.*, 46(3):312–28, August 1999.

[15] T. S. Braver, J. L. Paxton, H. S. Locke, and D. M. Barch. Flexible neural mechanisms of cognitive control within human prefrontal cortex. *Proc. Natl. Acad. Sci. U.S.A.*, 106(18):7351–6, May 2009.

[16] S. Doncieux and J.-A. Meyer. Evolving Modular Neural Networks to Solve Challenging Control Problems. In *Proc. EIS 2004*, 2004.

[17] J.-B. Mouret and S. Doncieux. Evolving modular neural-networks through exaptation. *IEEE Congress on Evolutionary Computation*, 2009.

[18] K. O. Stanley, D. D. Ambrosio, and J. Gauci. A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks. *Artif. Life*, 15(2):1–39, 2009.

[19] H. E. Rosvold, A. F. Mirsky, I. Sarason, E. D. Bransome Jr., and L. H. Beck. A continuous performance test of brain damage. *J. Consult. Clin. Psych.*, 20(5):343–350, 1956.

[20] T. E. Hazy, M. J. Frank, and R. C. O'Reilly. Towards an executive without a homunculus: computational models of the prefrontal cortex/basal ganglia system. *Philos. T. Roy. Soc. B.*, 362(1485):1601–13, September 2007.

[21] R. C. O'Reilly and M. J. Frank. Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural comput.*, 18(2):283–328, February 2006.

[22] K Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley and Sons, 2001.

[23] B. Girard, N. Tabareau, Q. C. Pham, A. Berthoz, and J. J. Slotine. Where neuroscience and dynamic system theory meet autonomous robotics: a contracting basal ganglia model for action selection. *Neural Networks*, 21(4):628–41, May 2008.

[24] J.-B. Mouret and S. Doncieux. Sferesv2: Evolvin' in the Multi-Core World. In *IEEE Congress on Evolutionary Computation 2010 CEC 2010*, 2010.