Neural Networks Learning: Some Preliminary Results on Heuristic Methods and Applications

D. Nicolay and T. Carletti

Department of Mathematics and naXys, University of Namur, Belgium
delphine.nicolay@unamur.be

Abstract. The aim of this paper is to present some preliminary results about the emergence of modular structures resulting from an evolutionary-like mechanism in the framework of artificial evolution. Besides this, we also focus and discuss the mathematical tools we exploit, namely artificial neural networks and genetic algorithms.

1 Introduction

A large number of biological networks exhibit modular structures resulting from a Darwinian evolution on a varying environment. Our research question is to study the emergence of such structures in the field of *artificial evolution*, that is we consider (virtual) robots controlled by artificial neural networks trained, for instance using genetic algorithms, to achieve abstract competing tasks, more precisely we focus on the following research question (see also Fig.1):

> Assume to have two neural networks, say N_1 and N_2, each one proved to be the best one to achieve a given task, say T_1 and T_2, in the absence of the other. The network, N, obtained by juxtaposition of N_1 and N_2, plus few nodes to control the joint outputs of N_1 and N_2, can perform the two tasks, but is its performance as good as the one of N_1 and N_2 separately? Can we find a better network N' where parts of N_1 and N_2 do merge together? Starting from a generic neural networks where links and thresholds are randomly generated, does the learning phase shape the network in such a way we can recover N_1 and N_2?

The rationale of this question is that under the pressure of the changing environment, here the necessity to perform tasks T_1 and T_2, the learning procedure will eventually promote networks having modules able to perform (less efficiently) both tasks instead of being well specialised in a single task.

The above question can be restated in our framework in the following way: two robots are trained to achieve each one a given task among reaching a global maximum (T_1) and moving around by avoiding "dangerous"zones (T_2). For each robot, we keep the neural network responsible for the best observed behaviour, that is whose fitness is the highest; then, we investigate if the juxtaposition of these two networks leads to the best behaviour when controlling a new robot

dealing with both tasks. To get our goal, we study the connections, i.e. synapses linking pieces of network responsible to solve the imposed tasks, to conclude if they are achieved independently or collectively. We also compare structures got by modular and global learning, i.e. by learning tasks sequentially or synchronously.

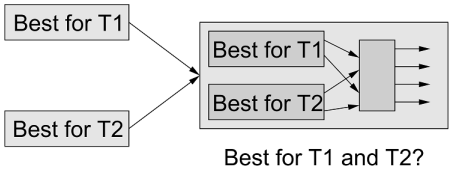

Fig. 1. Illustration of our research question.

This research is a part of a larger project whose aim is to study the emergence of structures because of a learning process, in order to unravel the mechanisms responsible for the learning process. In this paper, we discuss the mathematical tools we exploit and we present some preliminary results of this research.

We focus our preliminary analysis on the optimisation of our GA and on the improvement of the best networks, i.e. the ones that are responsible for the best observed behaviours. We implemented two versions of GA, namely a weighted GA and a GA based on Pareto-optimality. The comparison of these GA leads us to the conclusion that they both reach equivalent results. Moreover, we also notice that the performance of the GA based on Pareto-optimality depends on the method employed for the fitness allocation. Concerning our best networks, we remark that the optimisation process doesn't take into account the density of the networks. We thus add a penalty on the number of links to deal with it. Further to following experiments, we remark that this third objective slows down the optimisation and we consider other solutions to reduce the number of links. We envisage to make use of random methods but they lead to strongly random results. We finally introduce the future analysis we envisage to study the structures of our networks.

2 Artificial Neural Networks

In our study, we consider robots controlled by artificial neural networks, for short ANN, which are calculations methods inspired by biological neural networks [7, 8]. Such ANN are made of inputs, outputs and hidden neurons linked together by synaptic connections. They receive information from input nodes and give answers through output nodes according to their connections and their weights. In Fig. 2 we report the model of neural networks that we exploit in the following.

In our model, the topology of the network is completely unconstrained, except for the self–loops that are prohibited. Each neuron can have two internal states:

activated (equal to 1) or inhibited (equal to 0). The weight of each link is 1, if the link is excitatory, or -1 if it is of inhibitory nature. Given the present state for each neuron, (x_i^t), the updated state of each neuron is given by the perceptron rule:

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

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

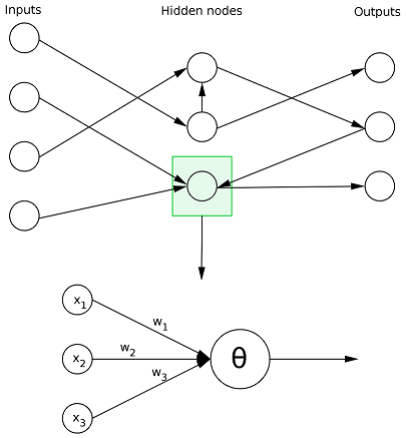

Fig. 2. Schematic presentation of the artificial neural networks model used in our research.

3 Research Study

We decided to follow the ideas presented by Beaumont [1] and thus to work on an abstract application where virtual robots, controlled by neural networks, are trained to learn two different tasks in a virtual arena. This arena (see Fig. 3) is a discretised grid of varying sizes, hereby for a sake of simplicity we fixed it to 20×20, with a torus topology, i.e. periodic boundary conditions on both directions, on which robots are allowed to move into any neighbouring square at each displacement, that is the 8–cells Moore neighbourhood. Assuming the arena is not flat and it possesses one global maximum, the first task we proposed involves reaching this global maximum, represented using a colour code in Fig. 3; the second task consists in moving as long as possible in the arena by avoiding "dangerous"zones, represented by black boxes in Fig. 3, i.e. zones where the robots loose energy.

The robots that we consider are built using 17 sensors and 4 motors. The 9 first sensors help robots to check the local height, given by the value of a normal bivariate function, of the square on which the robot is with respect to the squares around it. The 8 other ones are used to detect the presence of "dangerous"zones around it. The 4 motors control the basic movements of robots, i.e. north–south-east and west, and combining them, they also allow diagonal movements. Hence, the neural networks that control the robots are made of 43 nodes: 17 inputs, 4 outputs and 22 hidden neurons. This number has been set large enough to let sufficient possibilities of connections and to compare global and modular learning.

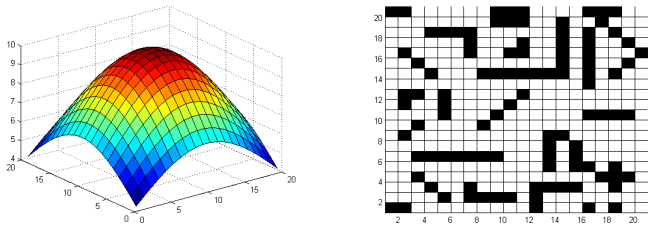

Fig. 3. Arenas where robots are trained. Left panel: arena for task T_1, the peak level is represented as a color code: red high level, blue lower leves. Right panel arena for task T_2, black boxes denote the "dangerous zones"to be avoided as much as possible.

As we described during the presentation of our neural networks model, the state of each neuron is binary. So, we transform the information acquired by the sensors in binary inputs to introduce them in the networks. Consequently, the state x_i of the nine first inputs is calculated according to the following formula:

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

where h_i is the height of sensor i and h_i^{min} and h_i^{max} respectively represent the minimal and maximal heights among the 9 available ones for the robot. For the last eight inputs, their state is 1 if their associated sensor detects a dangerous square and 0 otherwise.

We achieve the reverse walk to transform the outputs in motors' movements. With this aim, we devote two motors to the west-east movements and the two others to the north-south movements. Then, the robot movements follow the rule detailed in Table 1.

Table 1. Movement achieved by the robot according to the network's outputs.

Output 1	Output 2	Movement
0	0	-1
0	1	0
1	0	0
1	1	$+1$

For each task, the quality of robot's behaviour is quantified in order to determine the network that performs the best control. In the estimation of the first task, i.e. reaching the global maximum of the arena, the robot carries out 20 steps for each of the 10 initial positions of the training set. For each of these displacements, we save the mean height of the last five steps. Then, we calculate a fitness value, *fit*, which is the mean height on all the displacements. The quality of the robot's behaviour is finally given by:

$$quality_1 = \frac{fit - h_{min}}{h_{max} - h_{min}}$$

where $h_{min} = 3.93$ and $h_{max} = 9.98$ are respectively the minimal and maximal heights of the grid. A robot, and so a network, has a quality of 1 if, for each initial position, it reaches the peak at least at the sixteenth step end stays on this peak until the end of the displacement.

For the assessment of the second task, namely avoiding "dangerous"zones, the robot is assigned [1] an initial energy equal to 100 for the 5 initial positions in each of the 3 configurations of the training set. For this fifteen cases, it performs 100 steps in the arena. If it moves to a dangerous square, its energy is decreased by 5. Moreover, if it moves to a square that it has already visited, its energy is reduced by 1. This condition is necessary to force the robot to move instead of remaining on a safe cell. We save the final fitness of each case and we compute the mean final energy. If the energy becomes negative, we set its value to zero. The quality of the robot's behaviour is then calculated as:

$$quality_2 = \frac{\text{final energy}}{\text{maximum energy}}$$

A robot has a quality of 1 if, for each initial position of each configuration, it moves continuously on the grid without visiting dangerous squares and squares already visited.

When both tasks are performed simultaneously, we have to slightly adjust our training and our quality measures. The training set is composed of the three configurations of the second task and their five respective initial positions. The robot is assigned an energy equal to 20 instead of 100 and it performs 20 steps in the arena. The rule for the loss of energy remains the same as well as the way of calculating the mean height on all the displacements. Nevertheless, when the final energy of one displacement is zero, we assigned a value of zero to its mean height on the last five steps. If the value *fit* is lower than h_{min}, we set the quality measure to zero. This alteration in the quality of the fitness is introduced to prevent robots from having a very good behaviour for one task and a very bad behaviour for the other one. Let us note that these two tasks are competing each other. Indeed, in the first task, we hope that robot stops on the peak and in the

[1] Let us observe that such parameters have been set to the present values after a carefully analysis of some generic simulations and by a test and error procedure.

second one, we ask it to move continuously. So, we look for a trade-off between the two tasks in the robot's behaviour. Let us also remark that the second task is more difficult to achieve than the first one.

Now that we know how to measure the quality of the robot's behaviour, we want to find neural networks responsible for good quality performance. We look for the suitable topology and the appropriate weights and thresholds to achieve particular tasks. For this stage, we decide to make use of a heuristic optimisation method, namely genetic algorithms.

4 Optimisation: Genetic Algorithms

Genetic algorithms (GA) are heuristic optimisation methods that draw their inspiration from the biological evolution of species [6] and that have been used abundantly in the literature. We decided to use genetic algorithms as optimisation method for two reasons. Firstly, the perceptron rule is a discontinuous threshold function and we can not apply the well-known backpropagation algorithm directly. Secondly, we can use genetic algorithms not only for optimising heuristically the weights and thresholds but also for finding a network's topology got used to the achievement of the trained task.

We implemented two versions of multi-objective genetic algorithms [4]. The first one is a weighted GA, i.e. where the fitness is a combination of quality measures got for each task achievement. The second one is based on the Pareto-optimality theory [3]. Individuals are sorted according to their rank of non-dominance and receive a fitness depending on this rank. Figure 4 presents the diagram of this second GA [2].

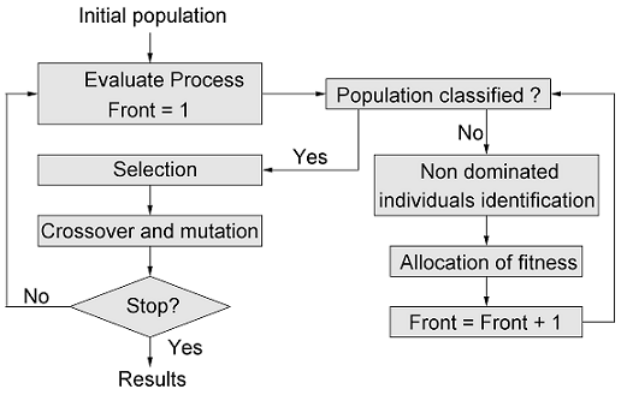

Fig. 4. Diagram with the general functioning of multi-objective GA based on Pareto-optimality.

[2] The first one is a standard GA and, thus, its diagram is not presented in this paper.

For these two GA, each individual is represented by the adjacency matrix associated to the neural network. As we prohibit self-loops, the thresholds are saved on the diagonal of this matrix. The selection process is a roulette wheel selection. We choose a one point crossover got used to our individuals, i.e. we randomly select a column of the matrix and we switch the following columns between the parents to obtain the children. The mutation process is a 1-inversion. To prevent the population from converging too fast to one single individual, we introduce new individuals at each generation.

A preliminary analysis leads us to get optimal parameters for our genetic algorithms. The GA is run during 10000 generations. Indeed, Fig. 5 shows that the most relevant part of the optimisation procedure happens before this limit. The crossover and mutation rate are fixed to a value of 0.9 and 0.01 respectively. Let us note that genetic algorithms are the tools we chose to train our networks but other heuristic methods could be considered.

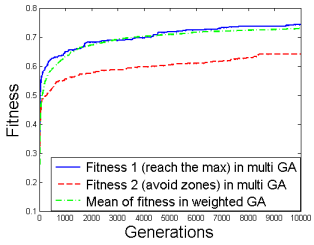

Fig. 5. Evolution of the maximum fitness according to the number of generations.

With the weighted GA, we run different simulations in order to analyse the characteristics of our final networks. Firstly, we train robots to manage only one task. Secondly, robots are trained to achieve both tasks together. Finally, we take the neural networks got after the training on one task and we add a training on both tasks. We hope to find differences between robots that learn both tasks together and robots that learn one task before the other. The first analysis of our networks leads to the conclusion that they are thick and that a large number of "useless"links, i.e. links that can be removed without changing the networks fitness, remains after the optimisation process.

We then switch to the GA based on Pareto optimality. In this case, both tasks are trained together and the GA provides a set of solutions instead of one unique solution (see Fig. 6). With this GA, we can easily add a third quality measure, i.e. a third objective function, in order to control the number of links in the networks. This third function is calculated as

$$quality_3 = 1 - \frac{\text{number of links}}{\text{maximum number of allowed links}}$$

The advantage of this GA, unlike the weighted one, is that it doesn't require any assignment of a combination of weights for the fitness.

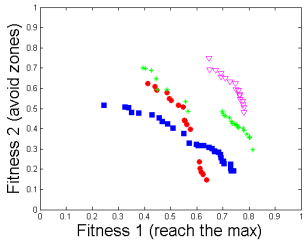

Fig. 6. Pareto-optimality fronts. Each symbol (color on line) represents the front obtained in different simulations.

5 Results and Perspectives

The comparison of the two genetic algorithms leads to the observation that, for the same number of generations, we got better fitness for weighted GA than for the one based on Pareto-optimality. The picture on the left in Fig. 7 represents the ten best networks got for ten simulations of the weighted GA and the Pareto fronts of ten simulations of the other GA. Our conjecture is that this is due to the way we defined the fitness according to the rank of non-dominance.

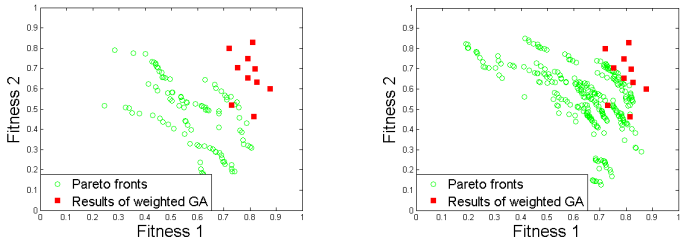

Fig. 7. Comparison between quality measures of the best networks optimised by our two GA. On the left panel, the allocation method to assign fitness in GA based on Pareto-optimality is the one developped by Deb [3]. On the right panel, the fitness is set to 1 for individuals of rank 1 and then it decreases by 20% for each subsequent rank.

To check this hypothesis, we test another way of defining fitness of individuals according to the rank of non-dominance. Instead of using the measure following by Deb [3], we assign a fitness of 1 to individuals of rank 1 and we decrease the assigned fitness by 20% for each subsequent rank. Results got with this new allocation method are presented in Fig. 7 on the right. We remark that this allocation seems to give better behaviours than the previous one. Indeed, with this allocation, we acquire networks that have approximately the same quality

measures than the ones got with the weighted GA. Moreover, the Pareto fronts appear more hung out, which means that we explore more possibilities during the optimisation. Nevertheless, if we consider a third objective, we observe in Fig. 8 that Deb's allocation seems to be more appropriate. Our hypothesis is that, as the dimension of the objectives space increase, there are more possibilities to be a non-dominated individual and, as almost all individuals have a fitness equal to one, the considered solutions are sparse in the space of possibilities. On the contrary, the Deb's allocation method takes into account the distance between solutions and favours solutions that are isolated.

As we have already introduced, although neural networks so far obtained lead to the expected robots behaviours, they exhibit many "useless"links. So, we consider different methods to decrease the number of links. Firstly we try to decrease the number of links by adding a third objective, namely a penalty on the number of links, in our genetic algorithms.But, concerning the weighted GA, the choice of the combination of weights to introduce is not trivial and the reduction (of the number of links) obtained is very low and slow. On the other hand, for the GA based on Pareto optimality, adding a third objective function prevents best networks from reaching equivalent tasks performance to the one that they have in the case of two objectives optimisation.

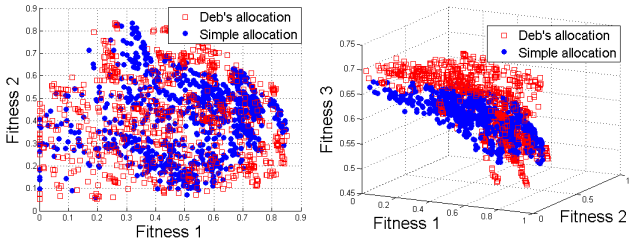

Fig. 8. Comparison between the two allocation methods when the optimisation is done on three objectives. The picture on the left panel presents the quality measures for the two tasks, that is a projection on the plane $T_1 - T_2$, we can clearly observe that both methods provide equivalent results. In the 3D view on the right panel, we can remark a significant difference to the advantage of Deb's allocation.

Then, we replicate the best networks got with simulations performed without any links constraints by removing "useless"links as long as possible. To remove these links, we begin by choosing one link randomly. Then we remove it and we check if the fitness decreased, in this case then the link is reinserted in the network, otherwise it is definitely removed and we go on by choosing another link. We stop this part of the algorithm when we get a sequence of 50 randomly chosen links that can not be removed. Next, we check every remaining link to see which ones can be removed without changing the fitness and we choose one of them to be removed. We iterate this algorithm until it's no longer possible to keep the same fitness by removing any link. We observe that this method returns

limited results. Indeed, it's a random method and the final matrix depends on the order in which we removed the links. Moreover we notice that removing what we think to be a "useless"link will sometimes reduce the fitness. The explanation is that, for each node, the sum of stimuli are divided by the number of incoming links and compared with a threshold to see if the neuron is activated or not. By removing links, we change the divisor and we influence the comparison.

In order to overcome this limit, we modify the thresholds each time that we remove a link. Some preliminary experiments shows that this change leads to networks with a lower number of connections. We also consider to introduce this approach in the genetic algorithms themselves. Even if this new method brings comparable quality measures with the previous one, we notice that it results in a significant decrease (30%) in the number of improvements observed during the optimisation phase. Another way that we could explore would be to develop a kind of simulated annealing for removing links with a certain probability. This method could be less sensitive to the order in which links are considered.

Once we get the networks responsible for the best robot's behaviour, we look for finding particular structures. We realise that two kinds of modularities can be defined. The first one is a topological modularity [5] that can be studied with some community detection methods such as the Louvain method [2] or by using measures of centrality or similarity. In order to analyse the second one, namely the functional modularity, that is to group together neurons that have similar dynamic behaviours, we follow two trails. We plan to use some recent information-like indicators [9]. We also decide to check the node relevance by randomly removing hidden nodes.

6 Conclusion

This paper is the presentation of the first phase of our research in artificial evolution, in which robots are controlled by neural networks and trained to achieve two abstract competing tasks with genetic algorithms. Our aim is to compare the behaviour of robots created by juxtaposition of the best networks trained for the achievement of one particular task and by global learning. We presented the context in which we realise the research and the mathematical tools that are exploited. We also detailed the preliminary results and the work that we plan to study the modularity in our networks.

Acknowledgements We would like to thank Andrea Roli for his help and his involvement in this project. This research used computational resources of the "Plateforme Technologique de Calcul Intensif (PTCI)"located at the University of Namur, Belgium, which is supported by the F.R.S.-FNRS. This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its authors.

References

[1] M.A. Beaumont. Evolution of optimal behaviour in networks of boolean automata. *Journal of Theoretical Biology*, 165:455–476, 1993.

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

[3] K. Deb. Multi-objective evolutionary algorithms: Introducing bias among pareto-optimal solutions. *Advances in Evolutionary Computing*, Part I:263–292, 2003.

[4] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons Ltd, Chichester, 2008.

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

[6] D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, 2005.

[7] P. Peretto. *An introduction to the modeling of neural networks*. Alea Saclay. Cambridge University Press, Cambridge, 1992.

[8] R. Rojas. *Neural networks: A systematic introduction*. Springer, Berlin, 1996.

[9] M. Villani, A. Filisetti, S. Benedettini, A. Roli, D. A. Lane, and R. Serra. The detection of intermediate-level emergent structures and patterns. In *Advances in Artificial Life, ECAL*, volume 12, pages 372–378, 2013.