Evolving Modular Neural Networks to Solve Challenging Control Problems

Stephane Doncieux Jean-Arcady Meyer
Animatlab - LIP6, France
http://animatlab.lip6.fr
{Stephane.Doncieux,Jean-Arcady.Meyer}@lip6.fr

Abstract

This article describes ModNet, a framework devoted to the evolution of modular neural controllers that affords possibilities of bootstrapping the search for efficient solutions to challenging problems. Initial knowledge may be provided either as modules assigned to specific computations, or as an overall connectivity pattern describing how modules could be connected to each other or to the controller's inputs and outputs. These possibilities are used to automatically design neural networks that control respectively two complex dynamic systems: a cartpole and a lenticular blimp.

Introduction

Using artificial evolution to automatically generate neural networks [Meyer, 1998] has proved to be a promising approach to the control of complex dynamic systems like finless rockets [Gomez and Miikkulainen, 2003] or a variety of rolling, walking, swimming and flying robots [Meyer *et al.*, 2002]. But this evolution of neural controllers is still confronted with difficulties, especially when the corresponding networks must include sub-structures dedicated to specific computations or when they must be connected to numerous inputs and outputs. In the former case, when such a sub-structure is discovered by chance, it may well soon get lost due to the normal action of genetic operators like mutations or crossover. In the latter case, no interesting solution may ever be discovered because the exploration space is simply too large. This article proposes a solution to both these issues, through the management of modules that implement useful capacities - either because they have been specifically designed by a human, or because they have been discovered automatically by the evolutionary process itself. These modules are capable both of being propagated through successive generations and of efficiently bootstrapping the exploratory algorithm. This is possible thanks to the use of a dedicated framework, ModNet, which will first be described in this paper and then put to work in two non-trivial applications where neural networks are used to control respectively a cart-pole and a blimp.

1 ModNet

ModNet is a framework that calls upon modular encoding, thus making it possible to generate neural networks that are collections of modules. In traditional approaches to the evolution of neural networks, the elementary units that are manipulated are either the neuron or the connection. Likewise, traditional approaches to modularity usually consist in evolving a single neural network that is replicated to produce symmetries [Kodjabachian and Meyer, 1998] or in letting evolution decide how predesigned modules may compete for the control of a given system [Nolfi, 1997]. In ModNet, the units that are manipulated by evolution are modules that describe sub-networks whose structures are globally stable throughout the course of evolution and that serve as building blocks on which the evolutionary process may capitalize. These modules may encapsulate some a priori knowledge about the problem to be solved, or they may emerge from the evolutionary process.

1.1 The chromosome

Every chromosome associated with ModNet is made up of three components: a list of model-

modules, a list of modules and a list of links between modules (Figure 1).

The list of model-modules specifies which modules, among a list initially provided by the experimenter, may be included in a given chromosome to generate the network. Each such model-module contains a full description of the sub-network it represents, i.e., it describes its structure and specifies all the parameters necessary to its functioning, like connection weights, slopes of transfer functions, or time constants.

The list of modules specifies which modules, among the list of model-modules, are ultimately incorporated into the developed neural network. Thus, a given module may appear several times within the same controller. This feature of ModNet makes it possible to significantly reduce the amount of information necessary for the description of the final network. It also affords the possibility of simply coding symmetries because the same input may be linked to different outputs through identical modules, as shown below.

The list of links specifies how modules are interconnected in a given network. A link between modules does not correspond to a connection between neurons. Instead, it associates a network's input or a module's output with an input of another module or with an output of the network. This association entails the fusion of both elements involved. For example, because a link associates the output of module 1 to the input of module 2, the output neuron of module 1 is merged with the input neuron of module 2 in the network of Figure 1.

1.2 Genetic operators

The mutation operator may change each component of a chromosome. In particular, it may modify the parameters of a given module which, in the applications described below, are connection weights only. Each of these parameters is represented by a string of eight bits using a binary encoding. It may be mutated with a 0.1 mutation rate. Likewise, only traditional artificial neurons with sigmoid transfer functions are used here.

Structural mutations may also randomly add or delete elements in each of a chromosome's lists. Beside the insertion of new model-modules in the model-module list, mutations may also insert or suppress modules in the network by modifying the other two lists.

Crossover operators are difficult to define and manage with neural network encodings. They are often not used at all, especially when direct encodings are concerned [Pasemann, 1997, Yao and Liu, 1997]. In ModNet, the crossover operator exchanges model-modules between chromosomes with a probability of 0.6. Thus, an efficient sub-structure can easily be propagated to new individuals because it will be manipulated as a non-breakable building block. Furthermore, after a crossover, an individual may benefit from efficient modules transmitted by each of its parents.

1.3 ModNet's bootstrapping

Each model-module that is included to the model-module list, either at initialization or through mutations over the course of evolution, is derived from a pool of modules initially provided by the experimenter. This set of initial modules may be chosen randomly, or it may integrate some knowledge - stemming from an a priori engineer's analysis or from results of previous experiments - which is instantiated in module structures or in parameter values. In particular, it is possible to specify the structure of a module, i.e., which neuron is connected with which other neuron, but not the corresponding connection weights. Likewise, it is possible to specify these weights, or to simply specify the sign of some specific connections within the structure. Be that as it may, any such initial specification may be later overridden by mutations over successive generations: its sole purpose is to bootstrap the evolutionary process.

Additional bootstrapping knowledge may be taken into account in ModNet through the use of so-called "connectivity patterns". Indeed, it is possible to specify a priori how some modules could be connected to each other or to the network's inputs and outputs. Again, such initial patterns may be exploited and propagated from generation to generation or they may get lost.

2 Applications

To illustrate the properties of ModNet, we applied it to two different and challenging control problems. The first concerns the cartpole, a classical benchmark in control [Wieland, 1991]. The second one deals with a lenticular blimp equipped with five sensors and seven motors.

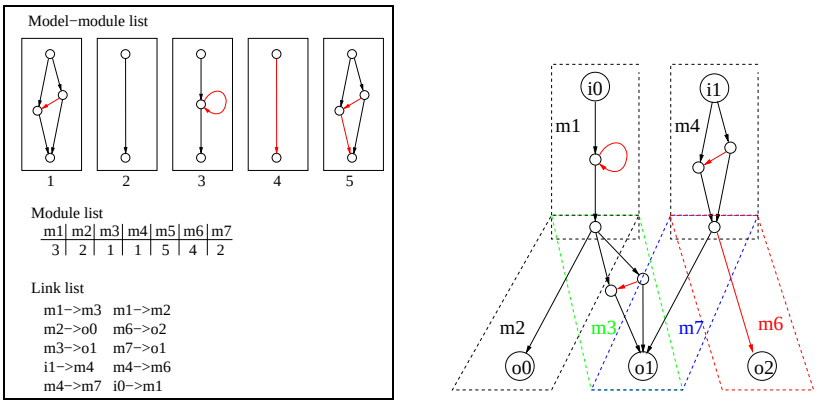

Figure 1: Left: An example of chromosome generated by ModNet. The chromosome is comprised of three components: a list of model-modules, a list of modules and a list of links. Right: The corresponding decoded neural network. Modules m3 and m4 are copies of the same model-module 1. Likewise, modules m2 and m7 are copies of model-module 2.

2.1 The Cartpole

The cartpole consists of a pole mounted on a cart in such a way that the pole can swing in a vertical plane. To swing and to balance the pole, the cart must be pushed back and forth on a rail. Starting from an arbitrary initial position of both the pole and the cart, the goal of the control is to apply a sequence of forces of constrained magnitude to the cart such that the system remains as close as possible to the upright position and to the centre of the rail. We provided the controllers with the instantaneous deviations of the pole's angle and the cart's position with respect to their target values. Although the stabilization of the cartpole requires the derivatives of these values, we did not provide this information to the network, which accordingly had to approximate it in order to solve the problem.

We performed two series of experiments that differed only in the initial pool of modules we provided. In the first series, the initial pool contained a single module with a single input and output, and with a direct connection linking the two. This constituted a control experiment, as it closely ressembled a direct encoding approach. In the second series of experiments we provided a single module also with a single input and output, but we did not specify its structure and let the evolutionary process discover it.

The fitness function we used is the sum of two terms:

$$f(x) = p(x) + \frac{1}{nb_{DOF}} \sum_{i \in DOF} \left(1 - \frac{\sum_t (d_i(x,t)^2)}{T}\right)$$

The first term, $p(x)$, measures the percentage of the maximum evaluation time the cartpole spent before going out of the limiting boundaries we defined (± 0.2 rad for the angle of the pole and ± 2m for the position of the cart). The second term measures the performance of the control, as the average, on each of the degrees of freedom and over the evaluation period, of the square distance between the actual and the target positions (angle=0 and position=0). This term has been normalized, as to lie between 0 and 1. Thus, the total fitness varied between 0 and 101. A value greater than 100 means that the cartpole did not overstep the imposed boundaries during the whole experiment, and a value of 101 means that it stayed at its equilibrium point throughout the entire evaluation.

The experiments that call upon a module with an unspecified structure are much more successful than the others: eight runs of the control experiments failed to keep the cartpole inside the desired boundaries during the whole evaluation, whereas all 20 experiments of the second series succeeded (Table 1). Furthermore, turns out that, although the best networks generated during control runs are able to maintain the cartpole within the assigned boundaries, they do not succeed in bringing it back to the equilibrium point. With such controllers, the cartpole keeps oscillating with a constant amplitude, which implies that the underlying neural networks do not compute any derivative[1]. In the second series of experiments, the generated controllers are able to

[1]Similar results have been reported in [Pasemann, 1997].

Generation	Control experiments				Unspecified structure			
	min	median	max	converged	min	median	max	converged
20	7,72	45,46	100,79	4	81,18	88,18	100,85	8
100	8,69	84,56	100,81	7	84,50	100,81	100,86	16
500	9,11	89,98	100,81	9	100,70	100,84	100,94	20
1000	9,11	100,64	100,81	12	100,70	100,84	100,94	20
2000	9,11	100,65	100,82	12	100,79	100,86	100,94	20

Table 1: Fitness values obtained in two series of experiments on the control of the cartpole. Twenty runs were performed in each condition. "converged" represents the number of experiments for which the best fitness exceeds 100.

drive the cartpole quickly back to its equilibrium point (Figure 2) thus implying that the underlying neural networks did succeed in computing a derivative. The interesting point is that this functionality is implemented within a single module (Figure 3) and doesn't result from the concatenation of several modules. As modules can be exchanged between chromosomes, as soon as one has been generated that computes a derivative, it can quickly propagate to new individuals thanks to the crossover operator. This wouldn't be possible should the corresponding functionality be distributed among several modules.

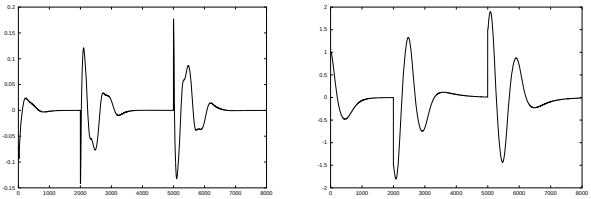

Figure 2: Behavior of the cartpole controlled by the network of Figure 3. The angle, in radians, is represented on the left, and the position, in meters, is represented on the right. To check the robustness of the controller, random disturbances of the pole's angle and the cart's position are evenly applied.

2.2 The lenticular blimp

The goal of this application is to keep a lenticular blimp (Figure 4) above a given visual target, at a given altitude and as horizontal as possible. This work is part of a project that aims at controlling a real platform, but the results presented here are preliminary and only follow from simulation.

Inputs to the evolved controllers are the pitch[2] and roll[3] angles, the altitude and the relative position of the target in the frame relative to the blimp[4]. The outputs of the controllers are sent to the blimp's seven motors. Details about the simulation model and results concerning the separate control of the pitch, roll and altitude are to be found in [Doncieux and Meyer, 2003].

As in the case of the cartpole, we performed several series of evolutionary runs that differed in the initial pool of modules we provided (Figure 5). In the first two series, the initial pool contained three modules of fixed structure, directly connecting one input to one or two outputs. Additional modules were provided to the initial pools of other two series in order to afford capacities for both derivative or integral computations. The design of the simplest derivative module was inspired by the results of the cartpole, as it computed from an ongoing signal the difference between two successive time steps. Two other derivative modules were endowed with two outputs (identical or opposite). As to integral modules, they called upon an intermediate neuron with a self-recurrent connection, a structure that is capable of integrating a signal, provided the corresponding connection weights are set appropriately[5]. Like the others, these modules were endowed with one or two outputs. Moreover, in one of these two series, the signs of given connections within some modules were initially set to a priori rational values in order to narrow the search space. For instance, recurrent connections of integral modules were initially positive, while two connections were initially positive and the

[2] rotation along the lateral axis of the blimp.

[3] rotation along the longitudinal axis of the blimp.

[4] On the real platform, this information is given by a visual tracking system.

[5] This is true only in the linear domain of the transfer function.

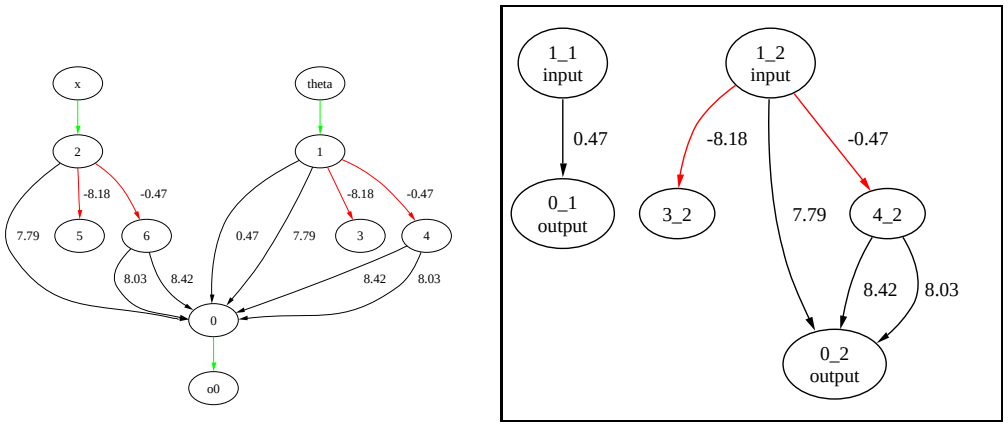

Figure 3: Left: best network generated after 2000 generations in a run of the second series (one module of unspecified structure in the initial pool). Inputs are the position x of the cart and the angle θ of the pole. Light connections concern the network's inputs and outputs, heavy connections are internal. Right: modules included in that network. The right module approximates the derivative of its input signal as it computes the difference between the current input and its value at the preceding time step.

third initially negative, within derivative modules with three connections. Finally, unlike the cartpole experiment, an additional connectivity pattern (Figure 6) was afforded to each of the four series, except the first. This pattern reflected a priori engineering knowledge about which sensor should be connected to which motor, as well as results of careful analyses of some networks evolved during preliminary runs not mentioned herein.

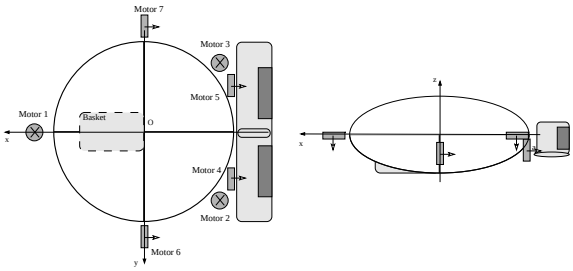

Figure 4: The lenticular blimp and its seven motors.

The fitness function was the same as that used for the cartpole, except that it was averaged over the five DOFs of the blimp. We defined boundaries of ± 0.7 rad for the pitch and the roll, ± 20 m for the horizontal position and 100 m for the altitude.

Although the differences are not considerable between the best fitness attained in each series (Table 2), they correspond to notable differences in behavior. None of the controllers generated in

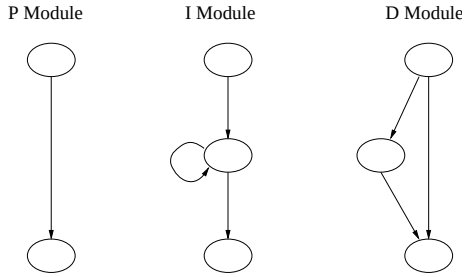

Figure 5: Proportional (P), Integrate (I) and Derivate (D) module. To these modules with one outputs, we have added modules with two outputs (identical or opposite).

the first series are able to control all the DOFs. The best individuals efficiently keep the pitch, the roll and the horizontal position, but none of them are able to maintain the altitude as well[6].

In experiments exploiting an initial connectivity pattern, all the controllers generated were able to control each of the DOFs. Moreover, providing additional knowledge such as module structures and connection signs helped still better solutions to be reached. The behavior generated by the best evolved network is shown in Figure 7.

[6] Individuals that efficiently kept the pitch, the roll and the altitude, but not the position, were also obtained in some runs.

Generation	Data	wo CP	w CP	w CP and PID	w CP and PID 2
20	Maximum	100,775	100,815	100,741	100,745
	Average	100,630	100,701	100,690	100,650
100	Maximum	100,792	100,817	100,826	100,82
	Average	100,663	100,764	100,73	100,693
500	Maximum	100,802	100,820	100,858	100,873
	Average	100,708	100,778	100,77	100,759

Table 2: Fitness values obtained in four series of experiments on the control of the blimp. Ten runs were made for each condition. "wo CP": runs without a connectivity pattern, "w CP": runs with a connectivity pattern, "PID": runs with Derivative and Integral modules, "PID2": runs with Derivative and Integral modules and with some connection signs set initially.

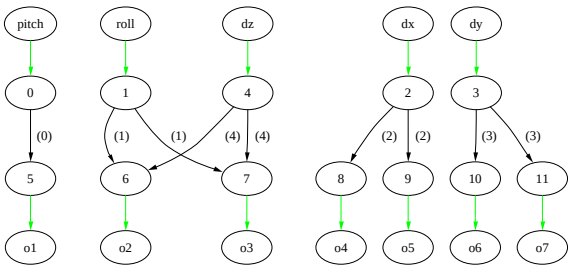

Figure 6: Connectivity pattern used in the blimp experiment. This pattern suggests how modules could be connected with each other or with the blimp's inputs and outputs. Neuron 3, for instance, might be linked to neurons 10 and 11 through a single module (module (3)) that has one input and two outputs. This pattern was inferred from a priori knowledge about the functioning of the blimp and from the study of controllers generated in preliminary experiments.

3 Conclusion

ModNet is a new framework devoted to the evolutionary generation of neural controllers that makes it possible to discover useful modules capable of being propagated from one generation to another or reused from one experiment to another. This framework also provides the possibility of bootstrapping the evolutionary process by taking domain-knowledge into account through the use of predefined modules or connectivity patterns. However, it should be emphasized that, as a result of the "tinkering" of evolution [Jacob, 1977], these initial modules and patterns do not necessarily survive in the final neural networks. This characteristic notably affords the possibility of automatically generating solutions that may turn out to be more efficient that those conceived by a human. Illustrations of such possibilities may be found in [Doncieux and Meyer, 2003, Doncieux, 2003], for example.

The ModNet framework has been successfully used here to tackle two challenging control problems involving an increasing number of DOFs: the control of a cartpole and that of a lenticular blimp. Results concerning the control of a helicopter are described elsewhere [Doncieux, 2003]. In experiments with a 2-DOFs system like the cartpole, evolution discovered "derivative" modules and converged much faster when it could adapt the structure of the modules to be included in the corresponding controllers. In the experiments with a 5-DOFs platform like the blimp, evolution never converged to efficient controllers without specifying a connectivity pattern suggesting which kind of neuronal organization should be tried first. Moreover, offering it the possibility of capitalizing on modules dedicated to useful computations like derivatives and integrals still helped improve the convergence.

These results suggest two things. The first is that, according to present technology and practice at least, it is highly unlikely that efficient neural networks capable of controlling dynamic systems with many more than 5 DOFs will be discovered automatically by artificial evolution left alone. The second is that this rather pessimistic conclusion should probably be tempered provided means for helping the evolutionary process are discovered and applied. This paper describes some bootstrapping procedures that seem to be useful in this respect. ModNet affords other capabilities that have yet to be implemented, notably that of regrouping several modules, belonging to an efficient controller at a given generation,

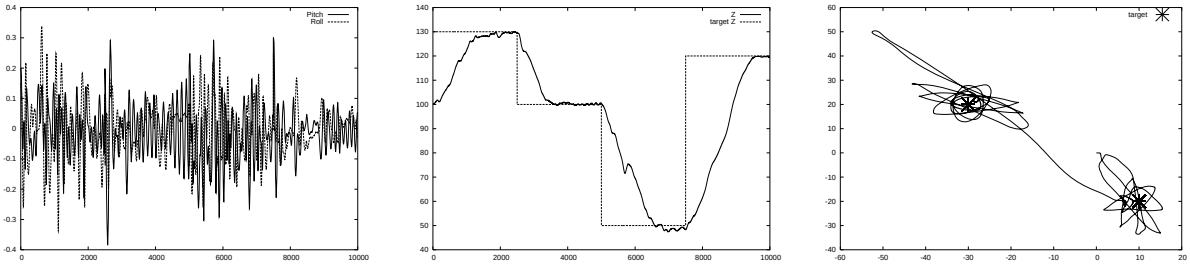

Figure 7: Behavior of the blimp controlled by the best network generated in a "w CP PID2". During the evaluation period (25 sec), the coordinates of the visual target as well as the target altitude are changed respectively once and twice. Meanwhile, the wind direction is changed several times, hence the observed oscillations around equilibrium values. Left: pitch and roll in radians (x axis in 25 ms time steps). Middle: altitude in meters. Right: 2D plot of the corresponding trajectory.

into some supra-module that could be passed on as such to the next generation.

References

[Doncieux and Meyer, 2003] S. Doncieux and J.-A. Meyer. Evolving neural networks for the control of a lenticular blimp. In G. R. Raidl et al., editor, *Applications of Evolutionary Computing, EvoWorkshops2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, EvoSTIM*. Springer Verlag, 2003.

[Doncieux, 2003] S. Doncieux. *Évolution de Contrôleurs Neuronaux pour Animats Volants : Méthodologie et Applications*. PhD thesis, Université Paris 6, 2003.

[Gomez and Miikkulainen, 2003] F. J. Gomez and R. Miikkulainen. Active guidance for a finless rocket using neuroevolution. In Erick Cantù-Paz et al., editor, *Proceedings of the Genetic Evolutionary Conference (GECCO03)*. Springer, 2003.

[Jacob, 1977] F. Jacob. Evolution and tinkering. *Science*, 196(4295):1161–1166, 1977.

[Kodjabachian and Meyer, 1998] J. Kodjabachian and J.-A. Meyer. Evolution and development of modular control architectures for 1-d locomotion in six-legged animats. *Connection Science*, 10:211–237, 1998.

[Meyer *et al.*, 2002] J.-A. Meyer, S. Doncieux, D. Filliat, and A. Guillot. *Biologically Inspired Robot Behavior Engineering*, chapter Evolutionary Approaches to Neural Control of Rolling, Walking, Swimming and Flying Animats or Robots. Springer Verlag, 2002.

[Meyer, 1998] J.-A. Meyer. Evolutionary approaches to neural control in mobile robots. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, San Diego, 1998.

[Nolfi, 1997] S. Nolfi. Using emergent modularity to develop control systems for mobile robots. *Adaptive Behavior*, 5(3/4):343–363, 1997.

[Pasemann, 1997] F. Pasemann. Pole-balancing with different evolved neurocontrollers. In *ICANN'97 - International Conference on Artificial Neural Networks*, 1997.

[Wieland, 1991] A. Wieland. Evolving neural network controllers for unstable systems. In *International Joint Conference on Neural Networks*, 1991.

[Yao and Liu, 1997] X. Yao and Y. Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, 1997.