AUTHOR(S): JAGADEESAN, A., MAXWELL, G. and MACLEOD, C.

TITLE: Evolutionary algorithms for real-time artificial neural network training.

YEAR: 2005

Publisher citation: JAGADEESAN, A., MAXWELL, G. and MACLEOD, C. 2005. Evolutionary algorithms for real-time artificial neural network training. Lecture notes in computer science [online], 3697, Proceedings of the 15th international conference on artifical neural networks (ICANN 2005): formal models and their applications, 11-15 September 2005, Warsaw, Poland, part 2, pages 73-78. Available from: https://dx.doi.org/10.1007/11550907_12

OpenAIR citation: JAGADEESAN, A., MAXWELL, G. and MACLEOD, C. 2005. Evolutionary algorithms for real-time artificial neural network training. Lecture notes in computer science, 3697, Proceedings of the 15th international conference on artifical neural networks (ICANN 2005): formal models and their applications, 11-15 September 2005, Warsaw, Poland, part 2, pages 73-78. Held on OpenAIR [online]. Available from: https://openair.rgu.ac.uk

Evolutionary Algorithms for Real-Time Artificial Neural Network Training

Ananda Jagadeesan, Grant Maxwell and Christopher MacLeod

School of Engineering, The Robert Gordon University
Schoolhill, Aberdeen, UK

Abstract. This paper reports on experiments investigating the use of Evolutionary Algorithms to train Artificial Neural Networks in real time. A simulated legged mobile robot was used as a test bed in the experiments. Since the algorithm is designed to be used with a physical robot, the population size was one and the recombination operator was not used. The algorithm is therefore rather similar to the original Evolutionary Programming concept. The idea is that such an algorithm could eventually be used to alter the locomotive performance of the robot on different terrain types. Results are presented showing the effect of various algorithm parameters on system performance.

1 Introduction

Artificial Neural Networks (ANNs) are normally trained before use, off-line. On-line learning, in complex Neural Networks - for example, those designed for pattern classification - has proved difficult and complex solutions like Adaptive Resonance Theory [1] have had to be applied. This is because learning new patterns effectively means altering the State Landscape of the network.

However, in some cases - for example, in many Control Systems - On-Line Learning is possible because, in contrast with Pattern Recognition, the network parameters may only have to change gradually as the controlled system changes. As an example, consider a legged mobile robot walking across a series of different ground types (for example, sandy, rocky or boggy). Obviously, in this case, for optimum locomotive efficiency, the robot will have to alter its gait pattern in response to the conditions underfoot. It may have to shorten its stride, for example, when moving from a hard to a sandy surface. One possible way to achieve this is for the robot's leg parameters (such as stride length, etc) to be under the control of a series of learned or pre-programmed gait patterns. However, this has the disadvantage of complexity and inflexibility. This latter point is illustrated when we consider the situation which might occur if the robot meets a surface for which it has not been prepared. Since it has no way of finding a suitable gait pattern, it has to lumber on with the "best guess".

The idea behind the Real-Time Evolutionary Algorithm (RTEA) is to constantly alter the robot's locomotive algorithms by a small random amount, evaluating corresponding changes to the fitness function (in the first approximation,

the efficiency of walking). Mutations which cause beneficial changes in fitness are kept; those which make the situation worse are discarded. In this way, the robot's control system is constantly seeking a better solution to the walking problem and will move towards such a solution, even when the robot moves onto a different surface.

Artificial Neural Networks were used to control the robot's legs, rather than direct control (for example, an algorithm generating a rhythmical step pattern), so that lessons learned from the experience could be applied more generally to other neural network controlled systems. Such networks are commonly used to control a variety of different mechanical and mechatronic systems.

2 Robotic Test System

It was decided to use a quadruped robotic system as a test bed for the neural network since it represented a generalised control system and also it has been used successfully several times in the past. The system is similar to that used by Muthuraman [2] and McMinn [3] for work on Evolutionary ANNs. Due to space restrictions here, only a brief overview of the system is given. The leg model used is a simulated linear servo mechanism with two active degrees of freedom, as shown in Fig. 1. The leg advances forward by one unit on receiving a positive pulse for one clock cycle and backward on receiving a similar negative going pulse. A similar arrangement is used to lift and lower the leg. The robot is driven forward when the legs are in contact with the ground and moving in the correct direction, the distance moved being equal to leg units moved under these conditions.

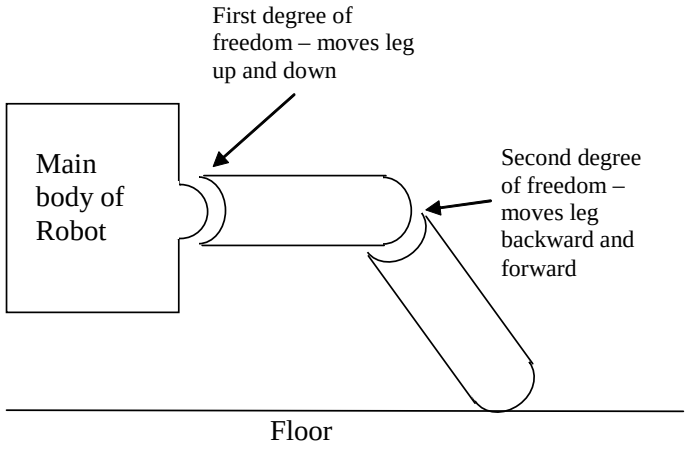

Fig. 1. Model of Robot Leg (Front View).

The neurons used in the network produce a positive followed by a negative pulse to drive the leg as shown in Fig. 2. Four evolvable parameters (D_1, D_2, T_{on1} and T_{on2}) are associated with each neuron; these are trained by the real-time EA.

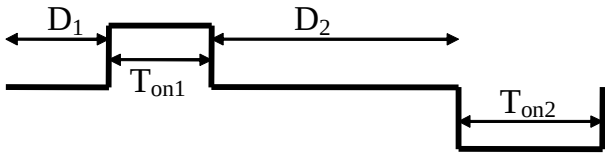

Fig. 2. Neuron Output.

The fitness function of the robot is a programmable combination of three factors - Stability, Fuel Consumption and Distance Travelled. The weighting between these can be changed in order that the system can develop different gaits. Stability takes into account the inclination of the robot and how many feet it has on the ground over its walking cycle. Fuel Consumption is related to the powered phase of leg use. Finally, Distance Travelled is measured, as indicated above, using the leg movements.

3 Neural Network Design

Several common network topologies and neuron types were tried at the start of the project. From these experiments, two important principles were discovered.

Firstly, neuron types with built in threshold functions (for example, Threshold Perceptron types) performed poorly. This was because either many small or a single large mutation is required to overcome the threshold and this causes large jumps in the network's fitness function, rather then small adjustments, making the global minima difficult to find. Therefore, for success, it is important to choose a unit which alters its behaviour gradually in order to avoid this problem.

Secondly, the network topology is equally important. Fully connected networks perform badly because mutations in one side of the network affect the other side. In the case of the legged robot used in the experiments, this meant that changes in one system of legs caused changes in an unrelated system and so the network fitness tended not to increase. Of course, it is often useful to provide some connection between different synchronised networks within a system like this and an effective network was one which took its overall timing from a master clock. Networks with recurrent connections performed poorly for similar reasons. The network topology is shown in Fig. 3. The outputs are taken from the child neurons (marked C); two are required for each leg (one for each degree of freedom).

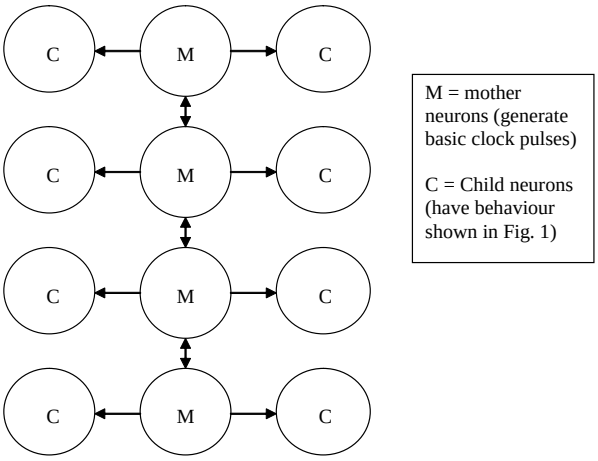

Fig. 3. Network Topology.

4 Real-Time Evolutionary Algorithm Parameters

The parameters of the RTEA were tested to establish their effect on system performance. The first to be tested was the effect of mutating all the parameters in the system compared with mutating only one at a time. Figure 4 shows the effect of mutating all the parameters compared with the fitness profile of only mutating one at a time. This experiment was done over a wide variety of system setups and parameter variation (over 2000 experiments representing all possible variations of system setup). Each point on the graph is the average of 81 such experiments. We might expect this effect to be even greater if we used mutation with an expectation value of zero [4]; however, this was not tested.

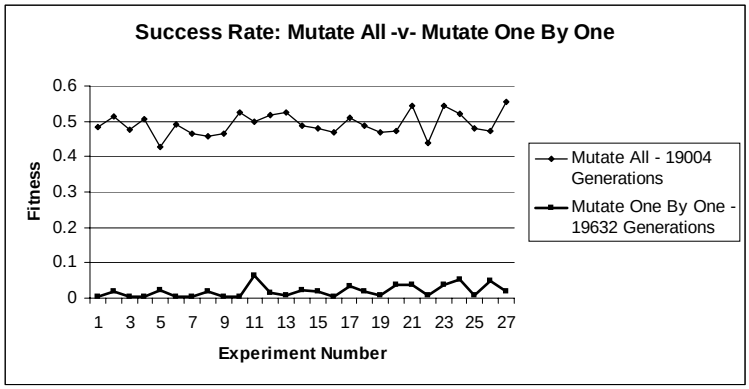

Fig. 4. The effect of mutating each parameter singly or in groups.

The effect of choosing a uniform versus a normal (Gaussian) distribution [4] is shown in Fig. 5. In this case the normally distributed numbers proved more effective. As before, the graph shows the effect over the whole range of system setup.

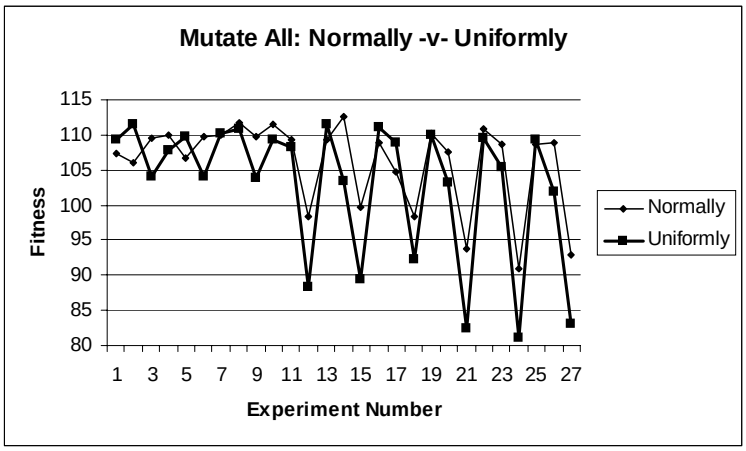

Fig. 5. The effect of Gaussian and Uniform mutation.

When performing these experiments, we have not allowed error increases to escape from local minima (which some algorithms such as simulated annealing do allow). The reason for this is that, in the case of this system, we are operating quite close to equilibrium. Should the system have to change gait (for example, from a walk to a trot), then this facility would have to be utilised.

In many such systems, mutation size is controlled by distance from equilibrium; examples include simulated annealing, where an artificial temperature controls mutation size [5], and Evolutionary Strategies with its one in five rule [4]. Tests were therefore conducted to establish how distance from equilibrium interacted with mutation size. It was found that these generalisations do not necessarily apply; Fig. 6 demonstrates this. In this graph disturbance range means distance from equilibrium (higher is further) and mutation rate is the average size of the mutation. Again, this is averaged over all system parameters.

5 Conclusion

The conclusions which were drawn from these experiments were that Real-Time Evolutionary Algorithms can optimise Neural Networks on-line in certain applications. Such applications are principally those in which small changes are needed to "tune" the performance of the system and include many types of control system.

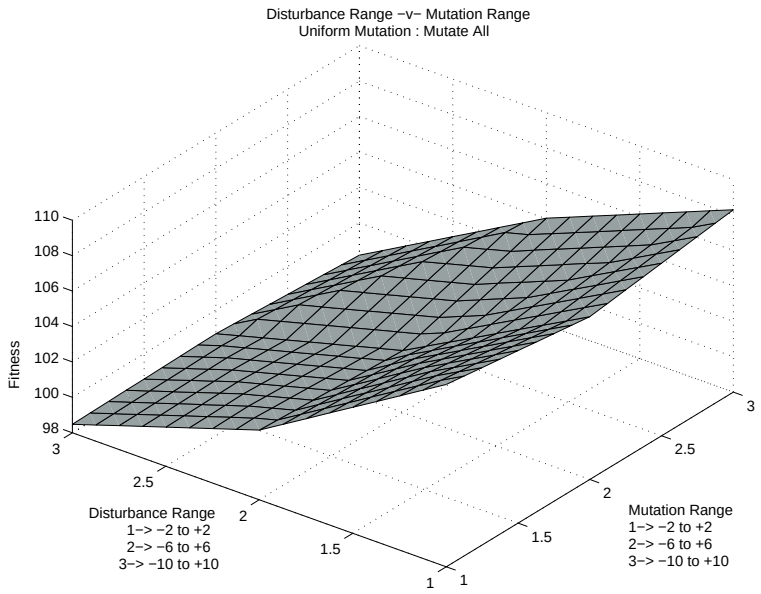

Fig. 6. Relationship between average mutation size and distance from equilibrium.

In such systems the type of neuron unit and network topology types are critical. The neuron unit should be one where output changes gradually with input and the network topology should be a feedforward type where unnecessary lateral connections are suppressed.

The experiments confirm that many of the recommendations for off-line optimization using EAs also apply on-line. These include that normal distributions and variable mutation rates are effective.

References

1. G. A. Carpenter and S. Grossberg, ART - 2: self organisation of stable category recognition codes for analog input patterns, *Applied Optics*, 26:4919-4930, 1987.
2. S. Muthuraman, G. M. Maxwell and C. MacLeod, The Evolution of Modular Artificial Neural Networks for Legged Robot Control. In *proceedings of the International Conference on Neural Networks and Neural Information Processing* (ICANN/ICONIP 2003), 488-495, Istanbul (Turkey), June 2003.
3. D. McMinn, C. MacLeod and G. M. Maxwell, Evolutionary Artificial Neural Networks for Quadruped Locomotion. In *proceedings of the International Conference on Neural Networks* (ICANN 2002), 789-794, Madrid (Spain), August 2002.
4. H.-P. Schwefel. *Evolution and Optimum Seeking*, Wiley, New York, 1995.
5. P. P. C. Yip and Y. Pao, Growing Neural Networks using Guided Evolutionary Simulated Annealing. In *proceedings of the International Conference on Evolutionary Programming*, 17-20, Istanbul (Turkey), June 1994.