# Incremental Growth in Modular Neural Networks

**Christopher MacLeod, Grant Maxwell, Sethuraman Muthuraman**

School of Engineering, The Robert Gordon University, Schoolhill, Aberdeen AB10 1FR, Scotland

**Email:** chris.macleod@rgu.ac.uk **Tel:** + 44 (0) 1224 262412 **Fax:** 262444

## Abstract

This paper outlines an algorithm for incrementally growing Artificial Neural Networks. The algorithm allows the network to expand by adding new sub-networks or modules to an existing structure; the modules are trained using an Evolutionary Algorithm. Only the latest module added to the network is trained, the previous structure remains fixed. The algorithm allows information from different data domains to be integrated into the network and because the search space in each iteration is small, large and complex networks with a modular structure can emerge naturally. The paper describes an application of the algorithm to a legged robot and discusses its biological inspiration.

**Keywords:** Genetic Algorithm, Modular Artificial Neural Network, Robotics.

## 1. Introduction

The earliest animals in the fossil record were simple single-celled protozoans. From these developed more complex multicellular forms and thence the various invertebrates. Latter came fish, then amphibians, reptiles and finally mammals. Although there have been many twists and turns on this path, there is an underlying trend from simple forms to complex ones. This trend may be spurred on by evolutionary “arm’s races” between predator and prey or animal and parasite

necessitating a more complex response. Whatever the cause of the increase of complexity described above, in any complex organism, development must take the form of small incremental changes. This principle is aptly illustrated by the evolution of the brain.

The brains of higher animals contain billions of neurons and trillions of synaptic connections - an enormous search space. Individual changes to the brain must therefore have been small. There is good evidence that as the brain grew in evolutionary time, it did so by adding small numbers of neurons on top of existing structures already in place. Indeed, it is difficult to see how else it could have grown to its current complexity.

McLean formalised this idea in terms of the “Triune” model of the human brain (Restak, 1979). In his model, McLean imagines the brain built in layers, rather like an onion. In the centre lie the oldest structures, concerned with basic survival and metabolic functions; one top of these are layers dealing with instinct and finally, in the youngest outer layers, the processes associated with higher thought. Fritzsch (1998) outlines one way in which small groups of neurons can become available for just such an expansion.

Such incremental growth of the network from simple to complex contrasts with the approach taken by traditional Evolutionary Algorithms like the Genetic Algorithm. Here the analogy is with a population of the animals at a single point of time crossbreeding to produce the fittest individual.

The algorithm outlined here combines the two approaches described above. It allows the network to increase in complexity and, at each stage, uses a traditional Evolutionary Algorithm to train the added modules. The final algorithm is based on a series of previous advances made to the incremental evolution idea by MacLeod (2001) and Muthuraman et al (2003a, b, 2005). A straight-forward practical description of the algorithm is given in MacLeod (2009). See also Capanni (2006) for a discussion of its search space and computational efficiency.

## 2. Operation of Algorithm

The Incremental Algorithm used to evolve the robots described here is based on the ideas outlined above. It works by adding small Neural Network modules to an existing network structure. Only the added modules are trained, the previous structure (including the added modules from previous iterations) remains fixed. The idea is that a large network can be built up piece by piece in this way.

To facilitate the growth of the network it is also necessary to evolve the body plan of the robotic test-bed in complexity at the same time. This is an analogue to the development in complexity in the body plans of animals as discussed above.

The technical reason for building the mechanics of the robotic system along with its control network can be seen if one considers the situation from a practical viewpoint. Because each added network is fairly small (in order to minimise the training algorithm search space), it is necessary to start with a robot which is easy to control. Once control of this simple situation is achieved, then the robot can be allowed to become more complex (for example by adding an extra pair of legs) and new modules

added to control these added functions. One can see that, using this idea, new sensors can easily be added into the system one at a time. In the case of this project the robot underwent the development shown in fig. 1 below.

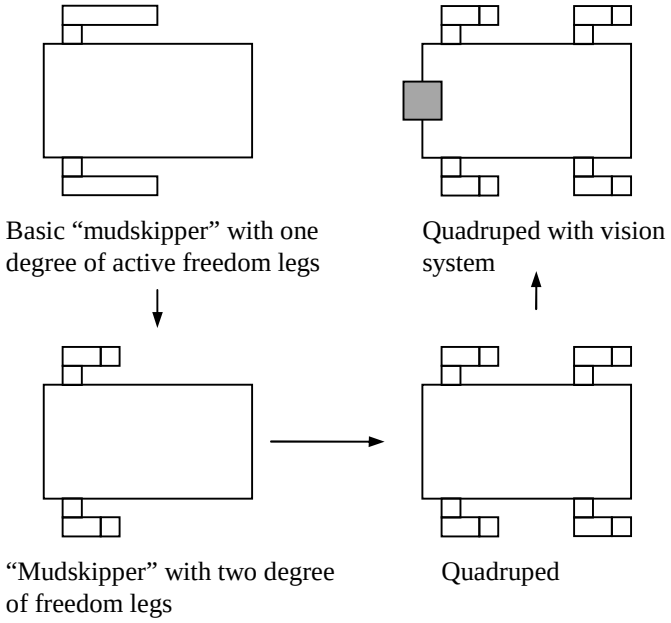


Fig. 1. the development of the robot’s body plan.

The initial body plan was that of a two-legged robotic “mudskipper.” The leg in this case being a simple, one degree of freedom, “peg” with a locking knee joint, described in the section below. This was replaced, in the next iteration, by a leg with two driven degrees of freedom (a hip and knee joint). The networks evolved to drive the one degree of freedom leg were assigned to the hip in this case and new modules added to control the knee. Next, a further two legs were added next to make the robot a quadruped. Finally, a vision system was added to control the response of the locomotion system.

The algorithm operated as follows:

1. Initially the robot's body plan was made as simple as possible and a basic neural network was added and trained using an Evolutionary Strategy. Once this network had trained to its maximum fitness, it was fixed and the weights were not altered on subsequent iterations.
2. If maximum performance had not been achieved using the first network, a network module was added and trained. This was repeated until the required performance was achieved - again only the added module is trained.
3. Once satisfactory performance had been reached using the added modules, the robot's body plan, behaviour or sensor configuration was allowed to become more complex - for example by the addition of further limbs, extra degrees of freedom or new sensors. New networks were then added on top of the previous set-up, following the procedure outlined in the second point above.
4. This process continues until the required level of complexity is achieved.

**3. Robotic Test-bed**

The main testing was done using a robot simulator. The simulation was developed in conjunction with a physical robot and has been used in several different evolutionary systems in the past (McMinn et al, 2002). The actuators in the simulation may be chosen to be a Linear type with the transfer function shown in equation 1.

$$\frac{d}{dt}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 1 \\ 0 & 0\end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0 \\ 1\end{bmatrix}V \quad (1)$$

Where *V* is the applied signal and θ is the angular displacement. Alternatively, a DC Motor type may be chosen as shown in equation 2.

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix}=\begin{bmatrix}0 & 1 & 0\\ 0 & -\frac{b}{J} & \frac{K}{J}\\ 0 & -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix}+\begin{bmatrix}0\\ 0\\ \frac{1}{L}\end{bmatrix}V \quad (2)$$

Where the additional symbols have there usual electrical meaning. Both types of actuator were used successfully in the tests.

The simplest leg had one active and one passive degree of freedom as shown in fig. 2.

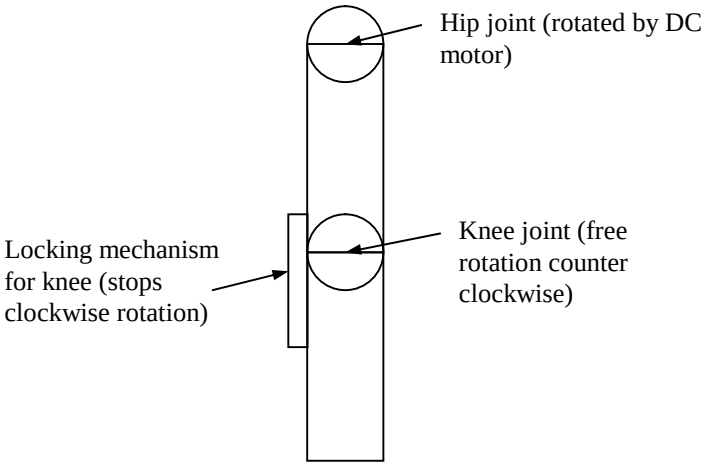


Fig. 2. The simple leg

In simulations using this model, Leg position 0 corresponds to a fully back, horizontal position. Position 90 is vertical and position 180 is the fully forward position. The leg is in contact with the ground between positions 81 and 99.

The second leg used has two active degrees of freedom and is shown in fig. 3.

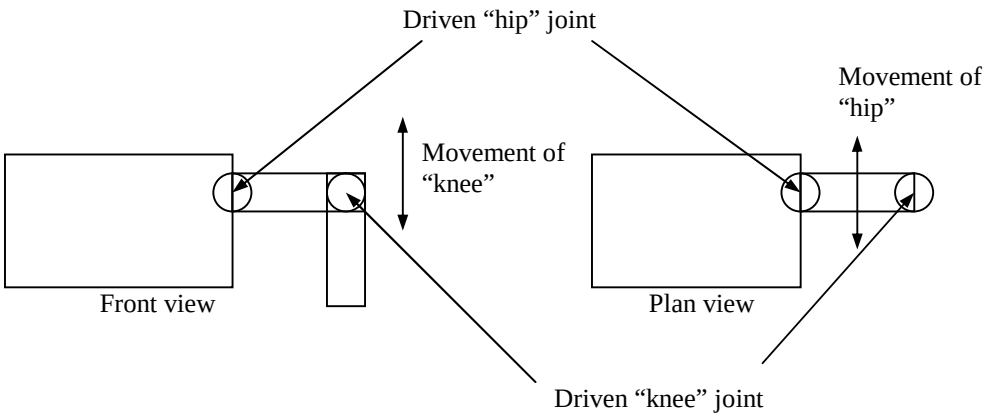


Fig. 3. The more complex leg

The joints of this leg have the same dynamics as those of the simpler leg outlined above, although because of the configuration of the leg, it is in contact with the ground between positions 45 and 135.

The fitness function is the distance walked by the robot in 500 time steps. The gait which the robot is evolving towards may be written as a logic function. For example, in the case of a quadruped trotting gait, were opposite legs are in phase, the two designated driving legs A and C must be down and moving forward together before the robot moves. The two non-driving legs B and D must be off the ground. This condition can be written as shown in equation 3.

$$A.C.\overline{B}.\overline{D} + B.D.\overline{A}\overline{C} \quad (3)$$

In the case of a pace, where legs on the same side are moving together, the condition is shown in equation 4.

$$A.D.\overline{B}.\overline{C} + B.C\overline{A}\overline{D} \quad (4)$$

The legs can move by one unit forward or backward in each time step of the algorithm and if the correct legs are down, moving forward and in contact with the ground - between positions 99 and 81 in the case of the biped as explained above - then the distance moved increases by one unit for each unit of leg movement. The other gaits may be similarly written.

**4. The Neuron and EA Implementation**

In the results which are outlined later in the paper, a [μ + λ] Evolutionary Strategy was used to evolve the connection pattern, neuron parameters and weights of each network module. This had an initial population size of 700 strings. The neuron parameters (explain below) were arranged in the string as shown in fig 4. On the first run, the strings are sorted in order of fitness and the worst performing 600 are deleted. The remaining 100 are selected randomly and recombined using Discrete Recombination. Each gene is taken individually and the fitter string has a 17 in 20 chance of contributing its particular gene to the offspring. The mutation operator uses a Normal Distribution with a standard deviation of 0.05. 25% of the offspring are mutated. This cycle continues until a new population of 700 individuals is generated. Testing and fitness evaluation then begin again.

The neuron used in the robot control system was Shigematsu's "Spike Accumulation and Delta-Modulation" (Shigematsu et al. 1996). This is a standard pulsing neuron

model and was chosen because it was relatively simple to code, equations 5 - 7 give the governing equations.

$$U(k) = \sum_{j=1}^{n} W_j(k) X_j(k) + aV(k-1) \quad (5)$$

$$Y(k) = G\left[U(k) - T\right] \quad (6)$$

$$V(k) = U(k) - pY(k) \quad (7)$$

Where *U(k)* is the normal leaky integration of the inputs (*a* is the feedback factor). *G(z)* is the threshold function and *T* is the firing function. *Y(k)* is the output and *p* is a constant which controls refractory time.

Although the Evolutionary Strategy and neuron model, outlined above, were used to obtain the results presented in section 7, this was simply because these techniques were at hand. It should be noted, that the Incremental Growth Algorithm itself is universal and works with any suitable set-up - provided that the rules outlined in section 6 are followed. The algorithm also worked well with several other neural systems, including Multilayer Perceptrons, which were used in the vision modules. In later experiments, Artificial Biochemical Networks (Capanni et al, 2005, 2006) proved particularly useful as they allowed the simple implementation of a pulsing unit which can be used for both actuator control and pattern recognition.

| First neuron | | | | | |
|---|---|---|---|---|---|
| Threshold Value ***G*** | Feedback Parameter ***a*** | Feedback Factor ***p*** | First weight Connection On/Off | First Weight Value | Other weights and neurons |

Fig. 4. Neuron coding in the Evolutionary Algorithm

## 5. The Vision System

The vision system was developed in the same way as the rest of the control system. Fig. 5 shows its development.

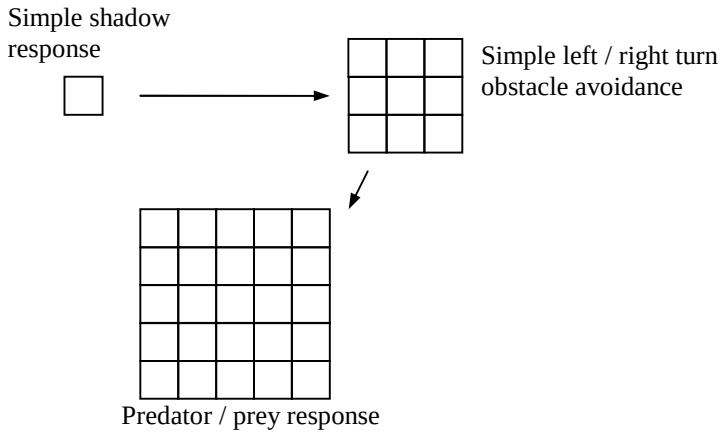


Fig. 5. The development of the vision system.

The system started with a single pixel which, when dark, caused the robot to retreat and when white caused it to proceed forward. This corresponds to an animal retreating when in shadow. From here the system was built up to a 25 pixel grid. With each addition, further, more complex patterns were added. Although space precludes an in-depth discussion of these patterns, they are follow explained in the references. The patterns were based on the behaviour of Toads as reported by Ewert, developed by Arbib and implemented by Reddipogu (Reddipogu et al. 2002). Using this model also allowed us to compare the results with those published in previous work and provides a group of patterns particularly useful in robotic work - for example, obstacle avoidance. The fitness function was altered to incorporate the vision system by allowing distance moved to be counted only when the correct pattern was received.

## 6. Underlying Principles

As experimentation with the algorithm proceeded, it became clear that the basic principles outlined in the sections above were not enough to always achieve a good outcome (a consistent increase in fitness). There were a set of further rules which needed to be applied to the system. These further rules were discovered through a process of trial and error, although in hindsight they are fairly obvious.

The rules are outlined below:

1. There is a minimum number of neurons required in each module added.

It was found that the overall fitness would sometimes not increase unless there was a certain minimum number of neurons present in a module. Fig. 6 shows how this effects the overall fitness of the system (after training of new module).

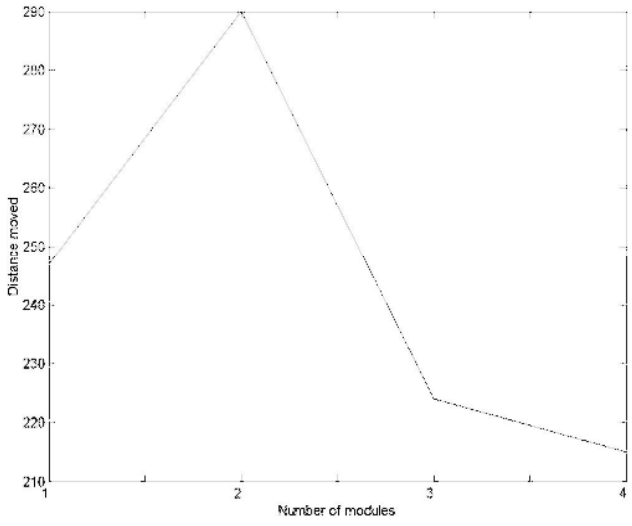


Fig. 6. Effect on fitness profile of having too few neurons in added module.

It is obvious from this that, in this case, adding the module actually has a detrimental effect on the fitness profile. The reason for this is that the added module must have sufficient flexibility, in terms of its functionality to perform the required mapping; this is analogous to the exclusive OR problem with simple Perceptrons. If a complex function is demanded of the improved system and the new module is not capable of delivering this mathematically, then it stands to reason that the system will fail to evolve past the point of what is possible. It is easy to built an automatic function into the algorithm which continues adding neurons into the module until it succeeds in improving fitness.

2. The neuron functionality is important.

This point is linked to that made above. In complex time-dependent systems, the neural units must be capable of producing the required signals. Again, we conclude that added module must have the necessary mapping capabilities. As previously mentioned, in later research, the team developed a more flexible and useful connectionist model the “Artificial Biochemical Network” (Capanni et al, 2005, 2006).

3. Connection patterns must be allowed to evolve as well as weights.

Initial tests were tried using fully connected modules. However, these proved ineffective, producing a similar result to that for too few neurons, as shown in fig. 7.

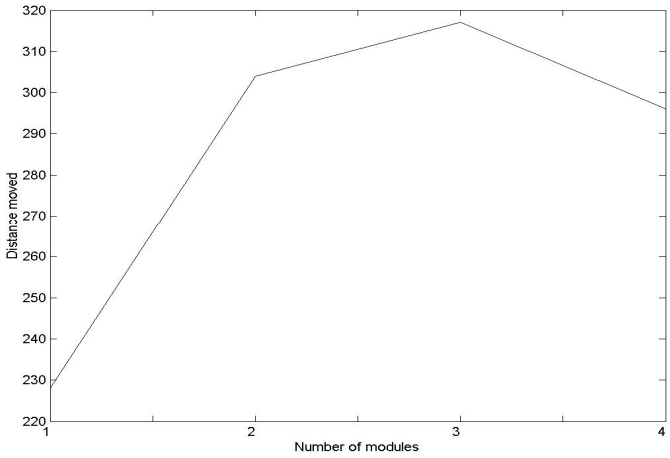


Fig. 7. The effect of using fully connected networks.

The reason for this is that connections which are not required but present effect neurons in the network unduly. For example, if two sets of legs (or other systems) are required to operate independently, it is not useful to have connections between them which may effect their synchronisation. It could be argued that unwanted weights will evolve towards zero anyway; however, the value of zero represents a very small target in the search space of the all possible weights (and consider the added dimensionally of many weights having to evolve towards zero) and is difficult to achieve and even a small residual weight can have a detrimental effect the network Taking all this into consideration, it is much easier to simply allow the Evolutionary Algorithm to choose whether connections exist or not as well as their weight values.

4. Module placement is important.

Consider growing a network to control a robotic leg. The two possible strategies for this are shown in fig. 8. We could add the new module in the signal chain before the previous one or after it.

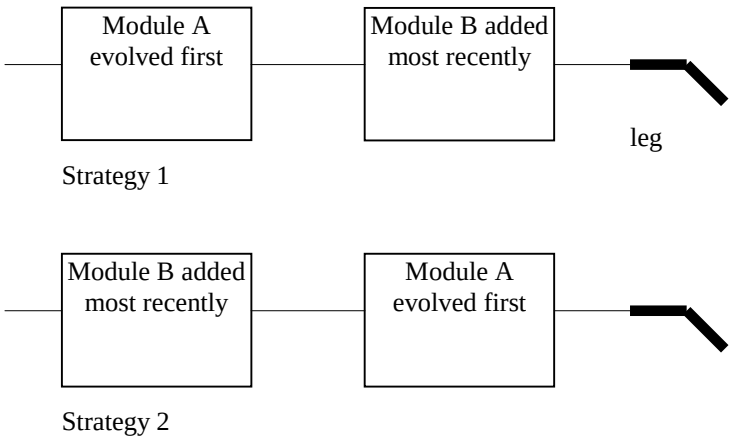


Fig. 8. possible placement of a new module.

The most successful strategy depends on the circumstances in which the network is being used. However, in this case the second strategy is often better as it allows the new module to produce the mapping which is best for the original module to interpret.

Although the initial work reported here was done using neural networks, there is every indication that this technique can be used as a Growth Algorithm in many different applications. This possibility is discussed in the conclusions.

The two statements below therefore generalise the points made above in the neural context to general systems.

1. The modules must be sufficiently flexible.
2. Placement and connectivity of the module with respect to others is important.

## 7. Results

Having discovered the necessary rules for growth, as outlined above, the algorithm ran without further problems. The Evolutionary Algorithm described above was chosen to train the weights of the added module, specify the network connections and choose the neuron parameters. As before, it is worth noting that these parameters were not critical. The algorithm was initially run with each new module containing one new neuron and this was increased if the fitness failed to improve as explained in the section above.

The initial tests focused on the simple “mudskipper” configuration shown in figure 1. Once the previously explained issues in regard of algorithm parameters were sorted out evolution proceeded well. A typical fitness profile from this stage of the tests is shown in fig. 9.

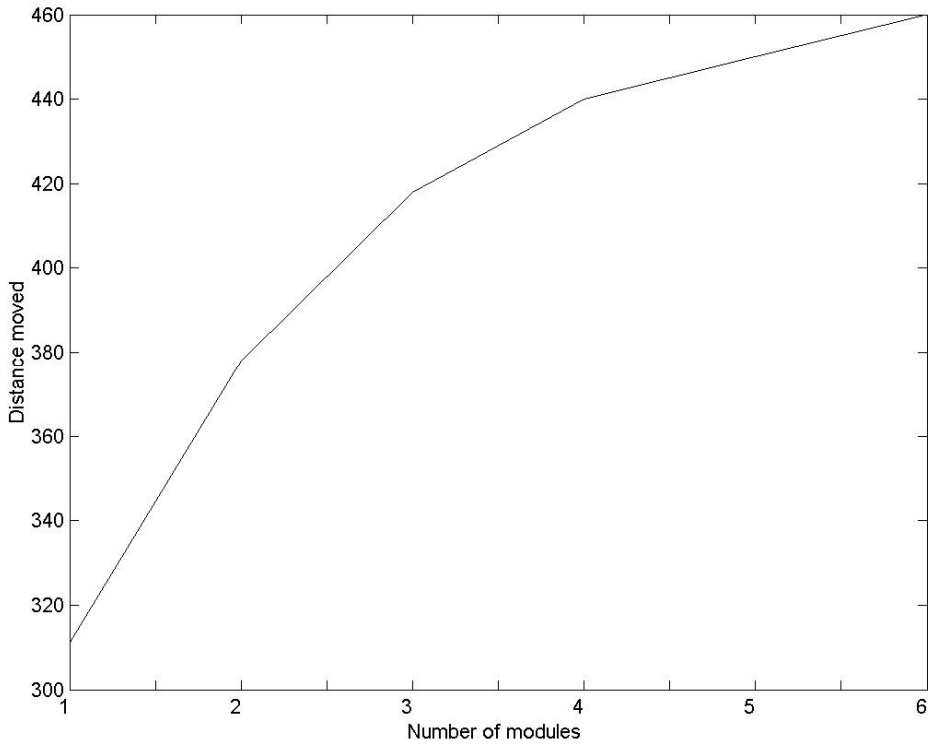


Fig. 9. Evolution with a single degree of freedom leg.

After evolving this, the next stage was to add the second degree of freedom to the leg. In this case the previously evolved network now controls the hip joint and the newly added modules control the knee joint. A typical fitness profile is shown in fig. 10.

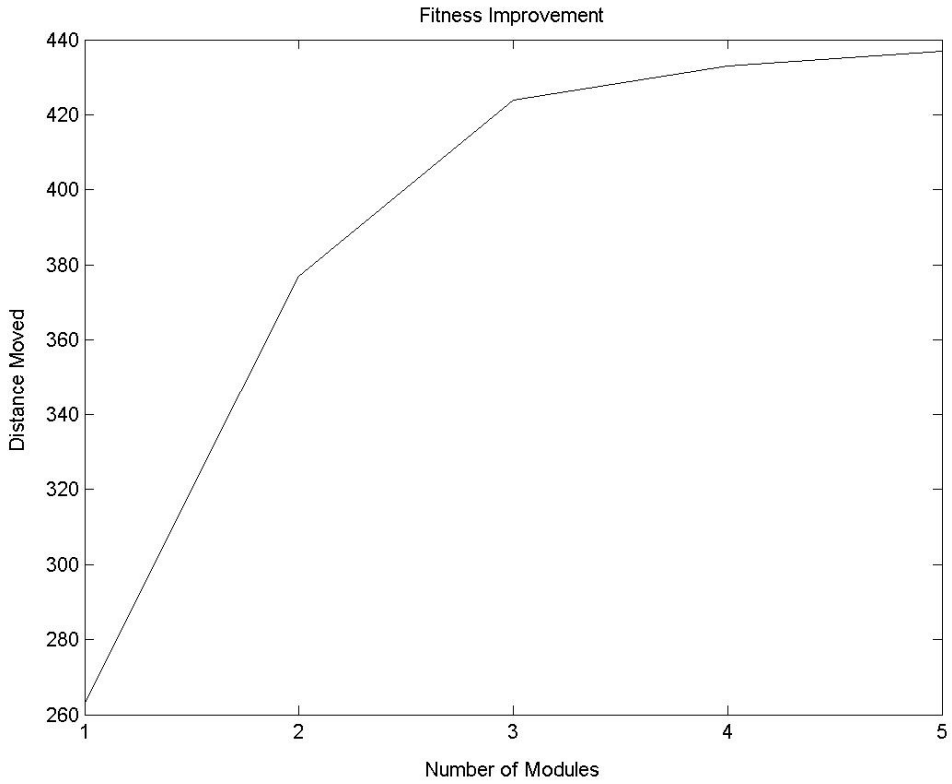


Fig. 10. fitness profile with two degree of freedom legs.

To test the robustness of the system, all the normal quadruped gaits were evolved in this manner, including Trot, Pace, Gallop, Pronk and Walk - these gaits were successfully produced by the system without any major difficulty. As an example, Fig. 11 shows the leg outputs for a trot gait (this is the gait which is also shown in the fitness profile in fig. 10).

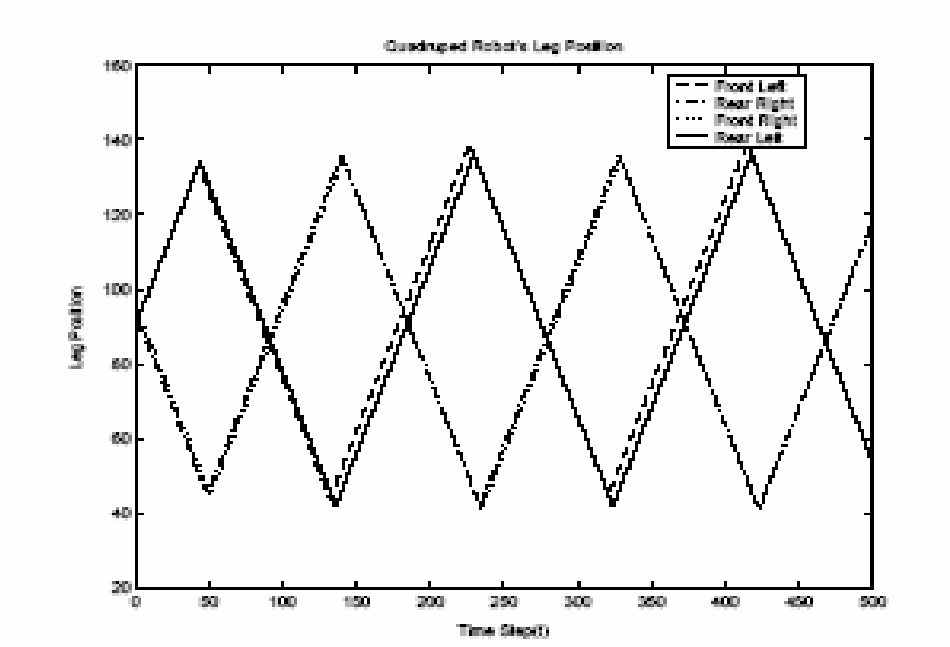


Fig. 11. Leg outputs for a trot gait.

Finally, once the locomotive system had been explored, the vision system was added.

The fitness profile for this is shown in fig. 12.

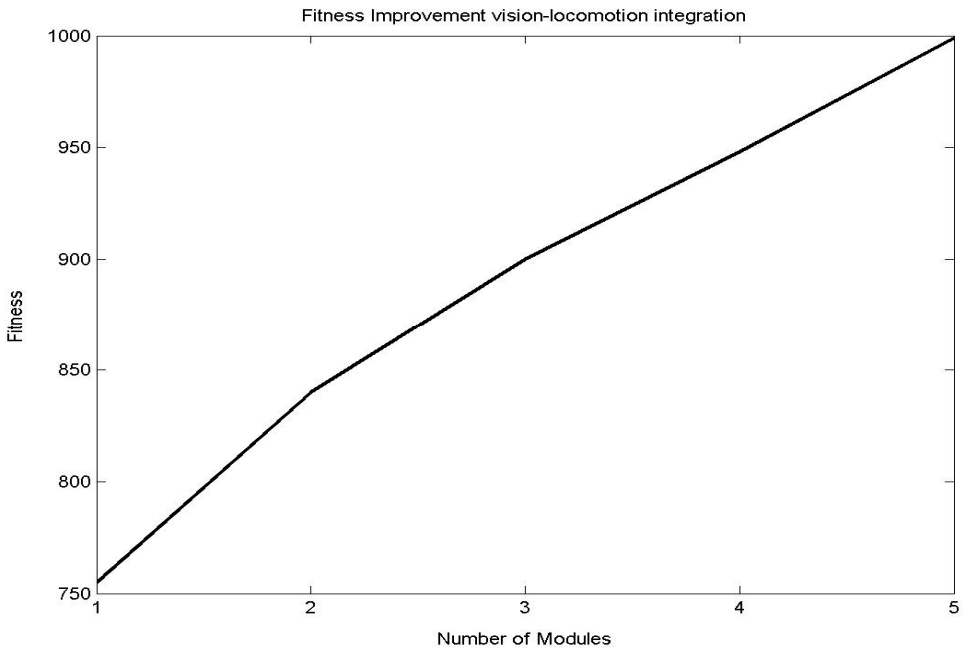


Fig. 12. system with vision modules.

A system overview is shown in fig. 13. In the end, the evolved network had over 200 neurons.

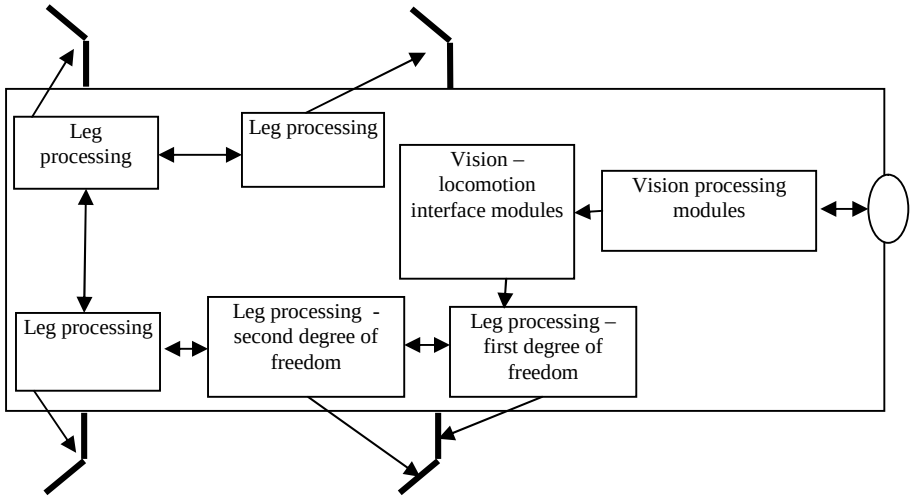


Fig. 13. Evolved system overview

## 8. Comparison with other work

A number of other workers have produced algorithms which incorporate incremental learning - for example, the Tiling algorithm (Mezard and Nadal 1989), Tower algorithm (Gallant 1986) and the Cascade Correlation algorithms (Fahlman and Lebiere 1990). Similarly, some algorithms allow the body plan of a robot to develop (usually without the controller also growing) - for example, the simulated creatures of Sims (1994), which are constructed from reusable blocks. Others following similar paths include (Lipton and Pollock 2000; Komosinski 2000) and Ventrella's (1994) walking stick figures. These workers' research is summarised by Taylor and Massey (2001) and Hornby and Pollack (2002). However, these are all constrained algorithms and do not allow an unbounded system to grow in complexity as illustrated above - this is important because it facilitates the growth of systems of arbitrary complexity, with many different sensor and actuator domains. Simple robotic controllers like those of Gruau (1993, 1995, 1996) and Kodjabachian and Meyer (1995) also fall foul of this requirement.

The algorithm's efficiency depends largely on the Evolutionary Strategy employed (in terms of speed and efficient search of the connection space), but in the case illustrated here the controllers produced in quadruped two degree of freedom simulations only had four more neurons than the optimum couple oscillator models of Kimura et al (1998) - assuming one coupled oscillator is equivalent to two cross coupled neurons.

## 9. Discussion and Conclusions

As illustrated in the paragraphs above, the algorithm provides a method by which multi-domain sensor information may be integrated into a controller which can potentially grow to an arbitrary complexity. This is in contrast with most other Evolutionary Algorithms which don't allow for growth. In terms of a biological analogy, most algorithms represent a population of organisms evolving to an optimum phenotype over a short period of time with no additions to their generic material. The algorithm presented here is more akin to organisms seen developing in complexity over a much longer period of (geological) time. This raises the possibility that it could be used to evolve extremely complex and intelligent robotic systems.

The algorithm also has some potential disadvantages. For example, the method is quite complex - especially if one were to try and integrate all the features into one automatic program. Also the networks produced are not always as efficient as those designed formally.

In future work, the focus will be on two main areas. Firstly, the engineering of the fitness function and robotic body-plan in order to provide a realistic "path" from a simple to a complex system. Secondly, how real-time, high-level learning may be in integrated into the system (can modules be added as the system is in operation).

Finally it should be noted that the algorithm has application outside those of artificial intelligence (MacLeod, 2009). For example, in high frequency electronics design, filters could be evolved by allowing the system to start as a low order circuit and then grow, piece by piece until a complex transfer function is realised. In mechanical

engineering, a good example is the design of aerodynamic surfaces, starting with simple shapes and then adding and building up additional sections to achieve a required performance. A final potentially important application is in the development of control systems for advanced prosthetic limbs. In such cases, there is an obvious incremental development possible from one degree of freedom (all joints locked) to many degrees (all joints active).

## References


Capanni, N., MacLeod, C., Maxwell, G., Clayton, W., 2005. Artificial Biochemical Networks. In: Proceedings of The International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA/IAWTIC '2005), Vienna, Austria, vol 2, pp. 98 – 102.

Capanni, N., 2006. The functionality of spatial and time domain artificial neural models. Ph.D. thesis., The School of Engineering, The Robert Gordon University., Aberdeen, UK. pp 23 - 29, 110 - 189.

Fahlman, S.E., Lebiere, C., 1990. The Cascade-Correlation Learning Architecture. In: Touretzky, D., (ed.), Advances in neural information processing systems 2. Morgan Kaufmann Publishers., Los Altos CA. pp. 524 – 532.

Fritzsch, B., 1998. Evolution of the Ancestral Vertebrate Brain, In: Arbib, M.A., (ed.), The handbook of Brain Theory and Neural Networks. MIT Press., Cambridge MA. pp. 373 – 376.

Gallant, S.I., 1986. Three constructive algorithms for network learning. In: Proceedings of The 8th Annual Conference of the Cognitive Science Society. Cognitive Science Society., Amherst Ma. pp. 652 – 660.

Gruau, F., 1993. Genetic Synthesis of Modular Neural Networks. In: Proceedings of The 5th International Conference on Genetic Algorithms. Morgan Kaufmann., San Francisco, CA. pp 318 – 325,

Gruau, F., 1995. Automatic Definition of Modular Neural Networks. Adaptive Behaviour. 3, 151–183.

Gruau, F., 1996. Modular Genetic Neural Networks for Six-Legged Locomotion. In: Alliot, J.M., (ed.), Artificial Evolution. Springer-Verlag., Berlin. pp. 201 – 219.

Hornby, G.S., Pollack, J.B., 2002. Creating High-level components with a generative representation for body-brain evolution, Artificial Life. 8, 223 - 246.

H. Kimura, H., Sakurama, K. & Akiyama, S. (1998). Dynamic Walking and Running of the Quadruped Using Neural Oscillator, In proceedings of *The 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems - Innovations in Theory, Practice and Applications*, 50–57. Victoria BC: IEEE.

Kodjabachian, J., Meyer, A., 1995. Evolution and development of control architectures in animats. Robotics and Autonomous Systems. 16, 161-182.

Komosinski, M., 2000. The world of framsticks: Simulation, evolution, interaction. In: Virtual worlds 2, Lecture notes in Artificial Intelligence. Springer-Verlag., Berlin. pp 214 – 224.

Lipson, H., Pollock, J.B., 2000. Automatic design and manufacture of robotic lifeforms. Nature. 406, 974 - 978.

MacLeod, C., McMinn, D., Reddipogu, A.B., Capanni, N.F., Maxwell, G.M., 2001. Evolution and devolved action: towards the evolution of systems. In: Appendix B of McMinn, D., Using Evolutionary Artificial Neural Networks to design hierarchical animat nervous systems, Ph.D. thesis. The School of Engineering, The Robert Gordon University, Aberdeen, UK.

MacLeod, C., 2009. Minds for Robots, Electronics World, Vol 115, number 1873. pp 16 - 19.

McMinn, D., Maxwell, G.M., MacLeod, C., 2002. Evolutionary Artificial Neural Networks for Quadruped Locomotion. In: Proceedings of the International Conference on Artificial Neural Networks, Springer-Verlag., Madrid. pp 789 – 794.

Mezard, M., Nadal, J.P., 1989. Learning in feed forward networks - the tiling algorithm. Journal of Physics A. 22, 2191 - 2203.

Muthuraman, S., Maxwell, G., MacLeod, C., 2003a. The evolution of modular Artificial Neural Networks for legged Robot control, In: Proceedings of The International Conference of Artificial Neural Networks / International Conference of Neural Information Processing (ICANN/ICONIP), Spring-Verlag., Istanbul. pp 488 – 495.

Muthuraman, S., MacLeod, C., Maxwell, G., 2003b. The development of Modular Evolutionary Artificial Neural Networks for Quadruped Locomotion. In: Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing. ACTA Press., Banff. pp 268 – 273.

Muthuraman, S., 2005. The evolution of modular artificial neural networks. Ph.D. thesis., The School of Engineering, The Robert Gordon University., Aberdeen, UK.

Reddipogu, A.B., Maxwell, G.M., MacLeod, C., 2004. An Innovative Neural Network Based On The Toad's Visual System. In: proceedings of Advanced Concepts for Intelligent Vision Systems Conference. ACIVS., Brussels. Paper 24.

Restak, R.M., 1979. The brain: The last frontier. Warner Books., New York. pp 50 – 52.

Shigematsu, Y., Ichikawa, M., Matsumoto, G., 1996. Reconstruction studies on brain computing with the neural network engineering. In: Ono, T., et al (ed.) Perception, memory and emotion: fronters in neuro-science. Elsevier., Amsterdam. pp 581 – 599.

Sims, L., 1994. Evolving 3D morphology and behaviour by competition. In: Proceedings of The 4th workshop on Artificial Life. MIT press., Cambridge MA. pp 28 – 39.

Taylor, T., Massey, C., 2001. Recent developments in the evolution of morphologies and controllers for physically simulated creatures. Artificial Life. 7, 77 - 87.

Ventrella, J., 1994. Exploration in the emergence of morphology and locomotion behaviour in animated characters. In: The proceedings of the 4th workshop on Artificial Life. MIT Press., Cambridge MA. pp 436 – 441.