Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot

M. Anthony Lewis [*] Andrew H. Fagg [†]
Alan Solidum

Center for Neural Engineering and
Institute for Robotics and Intelligent Systems
University of Southern California, Los Angeles, CA 90089-2520

Abstract

This paper describes the staged evolution of a complex motor pattern generator (MPG) for the control of a walking robot. The MPG is composed of a network of neurons with weights determined by Genetic Algorithm (GA) optimization. Staged evolution *is used to improve the convergence rate of the algorithm. First, an oscillator for the individual leg movements is evolved. Then, a network of these oscillators is evolved to coordinate the movements of the different legs. By introducing a staged set of manageable challenges, the algorithm's performance is improved. These techniques may be applicable to other complex or ill-posed control problems in robot control.*

1 Introduction

It is well known that intelligent robots must interact closely with the world. This interaction might be considered a discourse between the the computational structure of the robot and structure of world, mediated by sensor and actuators.

The design of this computational structure is a formidable task. The engineer must shape the structure so that the robot accomplishes a desired goal. She has analytical tools and an introspection about the world in which the robot will behave. Sometimes, this is not enough.

Outside the domain of human perception, the engineer may flounder. Micro-machines, for example, exhibit dynamics that defy common assumptions about the nature of friction.

Robotic colonies pose another challenge. There are few analytical tools that can be applied to the general case of a society of robots.

As a complement to well established analytical methods, it may be useful to explore techniques that exhibit self-organization, with a minimum of guidance by the design engineer. For this reason, we began to explore the technique of optimization by Genetic Algorithm (GA)[8] for the purpose of designing a controller for a robotic application. GAs are interesting because they seem to be applicable to a wide range of problems [7], they are well behaved when they are parallelized, and they are not brittle in terms of the form of the evaluation function.

GAs are inspired by biological evolution, and use a process of variation and selection to search a parameter space.

We feel that the the speed of convergence is the principle problem to be overcome in applying GAs to the control of a robot. Each point in the parameter space considered during the search must be evaluated by reference to a physical robot. The controller must be instantiated in the robotic hardware and executed for a period of time. This is a slow and time consuming process, creating a bottleneck in the search. Therefore, rapid convergence is highly desirable.

It would be naive to remove the engineer completely from this process. We believe that an engineer is still needed to specify a series of intermediate evolutionary subgoals that will minimize the time of convergence of the computational structure to a goal set.

1.1 Problem Statement

The goal is to evolve a neural network to generate a sequence of signals that can drive the legs of a 12 degree of freedom hexapod robot.

This configuration was a good starting point because it has been demonstrated by Beer [4] that a relatively simple network of neurons was capable of producing the required motion. The system was not too complicated to daunt nascent effort, and yet was not completely trivial.

1.2 Previous Work

A number of authors have analyzed central pattern generators for the control of walking [2, 4, 9]. In addition,

[*]mlewis@pollux.usc.edu
[†]ahfagg@pollux.usc.edu

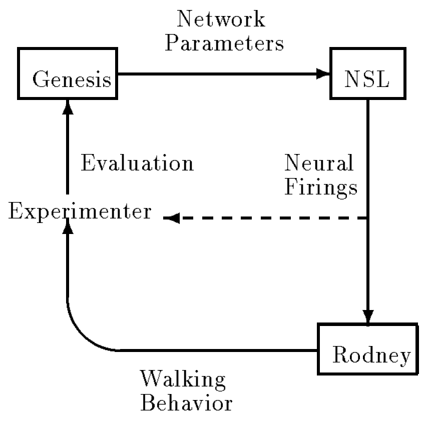

Figure 1: The Experimental Setup

Maes [12] has created a self-organizing system for the control of a walking insect. However, it is not immediately clear how these systems can be scaled for the design of hypercomplex systems.

Immediately relevant is the work of de Garis [6]. He has applied Genetic Algorithms to the design of neural network for control of *simulated* bipedal walking machines.

De Garis introduced the idea of *Behavioral Memory.* The key observation is that the evolution of a controller can be affected by the starting conditions of the search. It is therefore possible to pre-evolve a controller using another fitness function, that may have a smoother fitness landscape or have more easily satisfiable conditions. Once the system has been brought into the ballpark of an appropriate solution using a secondary fitness function, the primary fitness function is applied, which further refines the solution.

Using the concept of Behavioral Memory, De Garis has significantly reduced the length of the search. Unfortunately the evolution still requires a large number of individual evaluations (about 10000 to evolve a walking controller), far too many to performed online, with a real robot.

1.3 Present Approach

Our approach is based on the idea of *Staged Evolution*, which may be considered a refinement of Behavioral Memory. In staged evolution, the engineer is informed by principles of biological development.

In his essay on the development of *Amblystoma*, Coghill [5] discusses the stages through which this amphibian passes as it learns to walk on land. The undulatory behavior that it uses for swimming acts as a basis for the more refined walking motor control program. Each of the developing legs initially learns to coordinate their movement with the undulation of the body. As the body turns towards the right, the leftward forelimb reaches forward, and the rightward limb pulls back (the opposite is true for the hindlimbs).

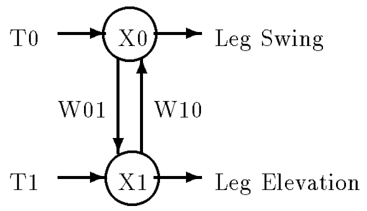

Figure 2: A simple oscillator circuit

As *Amblystoma* begins to move on land, the degree of undulation decreases significantly. Thus, the coordination of the limbs comes to depend less upon the undulatory movements, and more upon direct connections between the limb controllers. In addition, the control of the individual limbs becomes more refined.

We can conclude that the developing nervous system transition through the following phases of organization:

1. A single oscillator for undulation drives all limbs through mechanical coupling.
2. The limb movement later differentiates into refined, coordinated movement.

Certain modifications to the natural course of evolution were made to accommodate the kinematic limitations of the hexapod. Specifically, our robot was not capable of undulation. During the first phase of evolution the leg oscillations are constrained to be identical. The second stage of differentiated movement proceeded as in *Amblystoma.*

2 Experimental Apparatus

The experiments described in this paper were carried out on a six legged, Brooks-style insect robot named Rodney. Rodney's body is approximately 14 inches long and five inches wide. Each of the six legs are two-DOF and are actuated by Futaba servo motors. The motors provide limb swing and elevation motions. The servo motors are controlled by an on-board Motorola 68332 processor through the processor's TPU.

The GA simulator used for this set of experiments is GENESIS [10] which provides the GA engine. The evaluation function translates the genetic code into a neural network description. This network description is input into the Neural Simulation Language (NSL) [15], which simulates and displays the neural firings. The sequence of firings is then downloaded to Rodney, where the neural program is executed. The resulting behavior is scored by the experimenter. The score is then used as feedback to the GA (Figure 1).

3 The Neural Model

The position of joint i is driven by the state of a neuron x_i. The two neurons that are associated with a particular leg are capable of implementing an oscillator

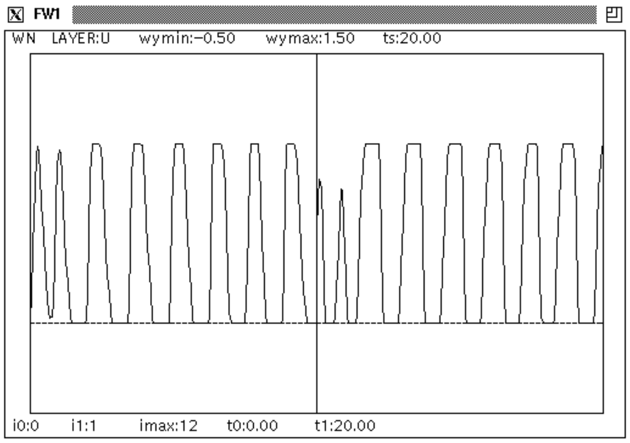

Figure 3: Neural firings for X_0 (left) and X_1 (right) plotted against time. Note that X_1 precedes X_0 by 90 degrees.

circuit. Figure 2 shows this local leg circuit. By setting the appropriate weight and threshold parameters, the two neurons will begin to oscillate at a particular frequency and phase. The neural dynamics are specified by the leaky integrator equation :

$$\tau \frac{dx_i}{dt} = -x_i + (\sum_{j=0}^{n-1} f(x_j) * w_{ji}) + t_i \tag{1}$$

where t_i is the threshold for unit i, w_{ji} is the weight from unit j to unit i, and f(x) is the logistic function:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2}$$

As an example, consider a network with parameters : T_0 = -0.5, T_1 = 2.0, W_{01} = -4, and W_{10} = 4 (note also W_{00} = W_{11} = 0). Initially, the neural states start at random values, but within several cycles, the two neurons fall into an oscillatory pattern, with a phase difference of 90 degrees (Figure 3).

The first phase of genetic learning is devoted to discovering a reasonable set of parameters that will implement a leg oscillator. When one unit is connected to the swing of the leg and the other to the elevation, the above oscillator will produce a stepping motion. Once this oscillator reaches a criterion behavior, it is replicated for each leg of Rodney. Simply connecting the individual oscillators to the legs of Rodney will not necessarily produce an effective walking behavior, due to the fact that there is no form of coordination between the legs. In this case, the legs will typically work against each other.

This problem is approached by the addition of connections between the oscillator units (Figure 4). These

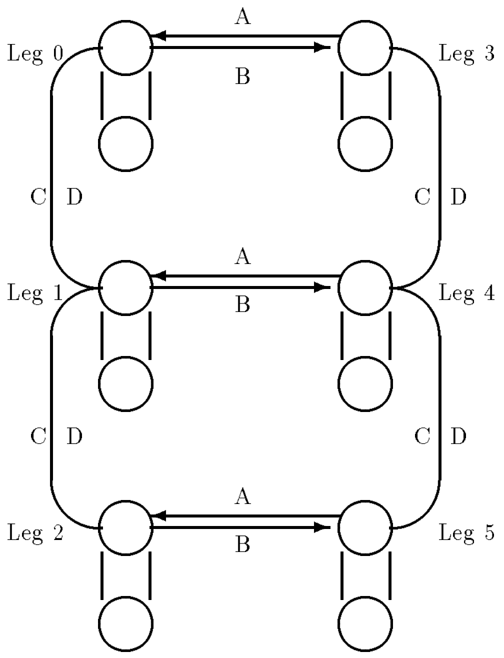

Figure 4: The Coupled Oscillator Circuit. Four additional parameters have been added to the circuit : A, B, C, D. Network symmetries are enforced by requiring several weights to take on the same value.

connections can enforce particular temporal constraints between the units. From the work of [11, 13, 14], we know that when a pair of non-linear oscillators are mutually connected through positive weights, the oscillators will tend to oscillate in phase. In addition, the oscillators will fire 180 degrees out of phase when the mutual connections are negative, and, as has been seen above, when one connection is positive and the other is negative, the first will precede the second by 90 degrees.

4 Experimental Process

4.1 Phase 1 : Evolving Oscillators

The first phase of learning concentrates on the creation of the oscillator circuit. The genetic code specifies the four parameters involved in this circuit: the two weights and two thresholds. Evaluation of this phase was performed through visual inspection of the temporal behavior of the oscillator neurons. The evaluation space is partitioned into a set of evolutionary stages, with the intention of leading the neural network to a point where it oscillated consistently. Note that these evaluations are only qualitative, and do not require knowledge of the internal construction of the oscillator.

Score	Description
0	Both neurons states (f(x_i)) are either > 0 or < 0
5	One neuron state is > 0 and the other < 0
10	For at least one of the neurons, f'(xi) > 0 and then changes to < 0 (or vice-versa)
25	Damped oscillations (damping to 0)
40	Oscillations that do not converge to 0, but the magnitude of oscillation is less than 1.
-	Increasing magnitudes of oscillation.
60	Oscillation over the entire range [0,1].

Stage two (score = 5) enforces an out-of-phase relationship between the two neural firings. The third stage (score = 10) rewards a network when an increase of firing affects the firing of the other neuron. The remaining stages present progressively more specific oscillation conditions.

Once at least half of the population begins to achieve scores near 60, the experiment transitions to the next phase.

4.2 Phase 2 : Evolving to Walk

The genetic string for the second phase not only specifies the circuitry for the oscillator, but also the set of connections between the oscillators. Four additional parameters are used to specify these connections, as shown in Figure 4. A single parameter specifies the value of several weights. For example, the connections that cross the midline from the left to right side take on identical values, no matter the location along the spinal cord.

Evaluation during this stage is considerably more simple that the first :

Score	Behavior
0	No movement (no oscillations)
$10 + L - T/10$	Oscillations, movement backwards.
$10 + \alpha L - T/10$	Oscillations, movement forwards.

where L is the number of inches walked along the axis of the body at its initial position, and T is the number of degrees turned during the walk. The T term is intended to favor solutions that keep Rodney oriented along the direction of travel.

Note that walking backwards is rewarded, as is walking forwards. The reasoning behind this stems from the fact that walking forwards versus backwards requires a similar set of parameters. The only difference is the phase relationship of the swing and elevation neurons If one were to simply alter the mapping from swing neuron to swing joint ($x_i = 1 \rightarrow$ swing forward to $x_i = 1 \rightarrow$ swing backwards), then a transition from walking backwards to walking forwards may be easily made.

5 Results

The genetic string used for this set of experiments consisted of 65 bits. Eight bits were used for each of eight parameters : W_{01}, W_{10}, T_0, T_1 (single oscillator circuit), A, B, C, D (oscillator interconnections). Each parameter encodes a weight value of the range [-8.0, 8.0], using a grey code. [1] Initially, the four oscillator parameters are randomly selected within the [-8.0, 8.0] range. On transition to the second phase, the four inter-oscillator parameters are initially set to 0.

The final bit (65th) determines the mapping from the leg swing neuron to the joint actuator. In one case, the firing of this neuron causes a forwards motion, and the other, a backwards motion. This allows for an easy transition between walking backwards and walking forwards, rather than requiring a complete recoding of the weights. The reason this is necessary is due to the fact that the set of goals posed by the first phase of evolution does not bias the relative phase relationship of the two neurons (the swing neuron preceding or following the elevation neuron by 90 degrees). Thus, by not allowing the backwards solution to easily transition to the forwards direction, about one half of the experiments would lead to massive extinction once the experiments transitioned to the second phase. In addition, allowing partial credit to those controllers that walk backwards gives these controllers the opportunity to make this transition through mutation [2].

For this set of experiments, the genetic parameters were set to the following values :

Parameter	Value	Description
population size	10	Number of genetic strings maintained at any one time.
pmutation	0.04	Probability of mutation of a single bit.
pcrossover	0.1	Probability that crossover will occur between two strings.
elitist factor	2	Number of best-performing strings that are guaranteed to propagate to the next generation.

The pmutation parameter was carefully chosen such that the weights were always changing, but not so high that well-performing individuals were not eliminated from the population at a rate higher than they could be added. In addition, the elitist factor causes the best two individuals in a particular generation to propagate (undisturbed) to the next generation.

[1] The grey code guarantees that it is possible to change 1 bit of the representation and cause a 1 unit change in the value of the representation. This is not the case with binary representation.

[2] Such a mutation may require a number of generations, due to the limited size of the population

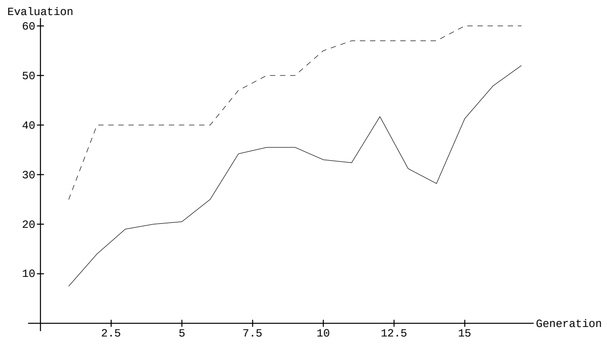

Figure 5: Performance plotted against generation for the evolution of an oscillator. The solid line shows the average performance of the population, and the dotted line represents the performance of the best individual at that generation.

Given these parameters, the first phase of evolution typically required 7 to 17 generations before at least half of the population began to oscillate over the entire [0,1] scale. In all cases, the parameters produced a solution in which the two neurons fired 90 or 270 degrees out of phase. The occurrence of the two solutions occurred with equal probability. Figure 5 shows a plot of the best and average scores versus generation for the first phase of learning.

In general, the second phase of evolution required on the order of 10 to 35 additional generations before the performance peaked. Figure 6 shows a typical performance curve for this phase. In this case, the average performance is significantly lower than the best performer of each generation. This is due to the controller's sensitivity to the high amount of mutation.

By the close of each experiment, Rodney had learned to produce a tripod gait. In this gait, the left- forward and backward legs move in phase with the right-middle leg. Thus, at any one time, at least three legs are on the ground. This gait is frequently seen in insects.

When the T factor (penalization for turning) is not included in the fitness function, and the distance of travel was measured based on the forward distance of the Rodney's bum, the resulting controllers tended to turn Rodney as much as 90 to 120 degrees. In this case, the distance to Rodney's bum was maximized, but the distance traveled for the rest of the body was not as desired.

Other gaits besides the tripod gait have been observed in insects [3]. One of the more common is the wave gait. The wave gait is characterized by rearward to forward waves of leg motions.

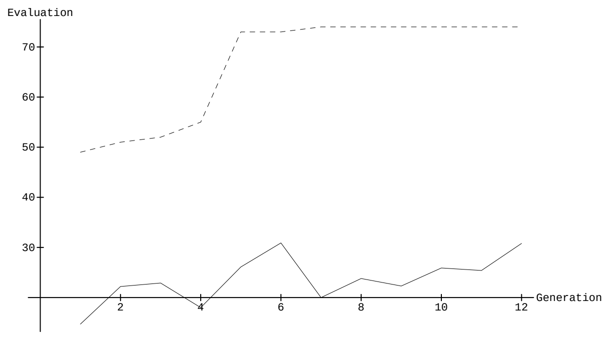

Figure 6: Performance plotted against generation for the walk controller. The solid line shows the average performance of the population, and the dotted line represents the performance of the best individual at that generation.

We found it very interesting that during several experiments, particular generations produced both individuals that performed the tripod gait, as well as those that generated the wave gait. In general, this splitting between two very different solutions lasted for several generations before the tripod gait came to dominate the population. This domination is caused by the fact that, in Rodney's case, the tripod gait tends to be more efficient than the wave gait. Once the tripod gait begins to make fine adjustments to the weights, it very quickly begins to outperform the wave gait.

The most surprising result of our experiments was that, in all experiments (in which $\alpha = 2$), Rodney preferred to walk backwards over forwards. This difference in efficiency appears to be due to angle that the feet push against the floor. This effect is something that cannot be anticipated without a very detailed analysis of the dynamics of the robot's interaction with the environment.

Further experiments, with $\alpha = 4$, result in the domination of the forward walking controllers. The results of these experiments indicated that around $\alpha = 3$, that the population would maintain an even distribution of both types of controller. These types of results can give us insight into the redesign of the controller software, or even of the robotic hardware.

6 Future Work

In the future, the design of more complicated, and difficult neural networks will be explored. Future extensions may include:

- *The use of control inputs to the network.* In real biological neural networks, as in most robot control applications, it is desirable to impose control on the network from higher levels. These may include start/stop commands, turn commands, and gait speed commands. In addition, reflex actions can be evolved to match the controller characteristics to terrain conditions.

- *Application of Different Fitness Functions* In nature, wave gaits are seen at low speeds and tripod gaits are observed and high speed. The transition may be do to variations in efficiency of each gait with speed. By using a fitness function that emphasizes efficiency, a greater repertoire of behaviors may be exhibited in the controller.

- *Evolution of Perceptual Systems* Many lower vertebrates exhibit approach and avoidance behavior [1]. When a large object approaches, the animal will flee. When a small moving object is present, the animal may show a predator response. It would be interested to evolve processing modules for visual input to detect looming stimuli and prey stimuli and to evoke the appropriate motor response.

- *Application to a 4-legged walking machine.* The USC robotics lab has constructed a 4-legged walking machine. The control of this device is much more difficult than in the case of Rodney. Firstly, balance is a critical factor. While Rodney is always statically balanced, the 4-legged machine will force the control network to consider the issue of *dynamic* balance. Secondly, the device may incorporate a wider array of sensor, which may include foot switches and attitude detectors. This will allow the device to negotiate rough terrain.

7 Conclusions

It would be impractical to apply Genetic Algorithms, in a straight forward manner, to the design of a neural network to control a robot. This is due to the enormous size of the search space and the high cost of evaluation.

It was demonstrated here, that the use of a staged evolution approach can significantly improve convergence to a solution. The staged evolution approach relies on the careful selection of intermediate solutions, or *fitness islands*, on the way to goal set. The design of the cost function of these fitness islands uses inspiration from biology. Specifically, the set of problems that biological systems had to solve as the original walking creatures evolved (formation of oscillators, to coordination of oscillations, to coordination of limbs) [5].

Finally, we have demonstrated the use of Genetic Algorithms in the design of robot controllers. GAs have the advantage that a complete understanding of the robot/environment dynamics is not necessary in order to design an effective controller. In fact, it is possible for the experimenter to better understand the problem based upon the solutions that are discovered by the GA, as was seen in our experiments through the discovery of the backwards walking solution.

If these techniques can be developed sufficiently, it may lead to a general methodology for building highly complex robotic systems.

8 Acknowledgements

We would like to thank Dr. George Bekey and Dr. Michael Arbib for their continued support throughout the course of our work. This research was supported in part by the Kaprielian Technology Innovation Fund at the University of Southern California, and by a fellowship from the University of Southern California Graduate School.

References

[1] Michael A. Arbib. *The Metaphorical Brain: II.* John Wiley and Sons, 1989.

[2] John S. Bay and Hooshang Hemami. Modeling of a Neural Pattern Generator with Coupled Nonlinear Oscillators. *IEEE Transactions on Biomedical Engineering*, BME-34(4):279–306, April 1987.

[3] Randall D. Beer, Hille J Chiefl, and Leon S. Sterling. An artificial insect. *American Scientist*, 79:444–452, 1991.

[4] Randall D. Beer, Hillel J. Chiel, and Leon S. Sterling. A biological perspective on autonomous agent design. In Pattie Maes, editor, *Designing Autonomous Agents*, pages 169–186. MIT Press, 1991.

[5] George Ellett Coghill. *Anatomy and the Problem of Behavior.* Hafner Publishing Co., 1964.

[6] Hugo de Garis. Genetic Programming : Building Nanobrains with Genetically Programmed Neural Network Modules. In *Proceedings of the International Joint Conference on Neural Neetworks*, July 1990.

[7] David E. Goldberg. *Computer-aided Gas Pipeline Operation using Genetic Algorithms and Rule Learning.* PhD thesis, University of Michigan, Dissertation Abstracts International 44(10), 3174B, 1983.

[8] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[9] D. Graham. Simulation of a Model for the Coordination of Leg Movement in Free Walking Insects. *Biological Cybernetics*, 26:187–198, 1977.

[10] John J. Grefenstette. *A User's Guide to GENESIS.* Naval Center for Applied Research in Artificial Intelligence, August 1987.

[11] Nancy Kopell. Toward a theory of modelling central pattern generators. In A. H. Cohen, S. Rossignol, and S. Grillner, editors, *Neural Control of Rhythmic Movements in Vertebrates.* Wiley-Interscience, 1988.

[12] Pattie Maes and Rodney A. Brooks. Learning to coordinate behaviors. In *Proceedings of the AAAI*, pages 796–802, 1990.

[13] R. H. Rand, A. H. Cohen, and P. J. Holmes. Systems of coupled oscillators as models of central pattern generators. In A. H. Cohen, S. Rossignol, and S. Grillner, editors, *Neural Control of Rhythmic Movements in Vertebrates.* Wiley-Interscience, 1988.

[14] G. Rinzel and B. Ermentrout. Analysis of neural excitability and oscillations. In C. Koch and I. Segev, editors, *Methods in Neuronal Modeling: From Synapses to Networks.* MIT Press, 1989.

[15] Alfredo Weitzenfeld. NSL : Neural Simulation Language, Version 2.1. Technical Report USC-CNE-91-05, Brain Simulation Laboratory, Center for Neural Engineering, University of Southern California, July 1991.