Published in: Zheng, ed. *Recent Trends in Mobile Robots*, pp 317-331, World Scientific, New Jersey, 1994.

Genetic Algorithms for Gait Synthesis in a Hexapod Robot

M. Anthony Lewis, Andrew H. Fagg and George A. Bekey
Institute for Robotics and Intelligent Systems
and
Center for Neural Engineering
University of Southern California
Los Angeles, CA 90089-0781, USA

Abstract

This paper describes the staged evolution of a complex motor pattern generator (CPG) for the control of the leg movements of a six-legged walking robot. The CPG is composed of a network of neurons. In contrast to the main stream work in neural networks, the interconnection weights are altered by a Genetic Algorithm (GA), rather than a learning algorithm. Staged evolution is used to improve the convergence rate of the algorithm, thus obtaining rapid evolution of behavior toward a goal set. 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. In this way, the designer specifies "islands of fitness" on the way to the final goal, rather than using a single fitness function or determining the explicit solution to the control problem. 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. The system itself determined how to evolve from one island to the next through the GA.

1. Introduction

A unique feature in the design and synthesis of robotic walking machines is the need to develop a method for generation and control of the sequences of leg movements representing specific gait patterns. In many animals such sequences are produced by special neural structures known as central pattern generators (CPGs). For engineers designing walking machines, the gait sequence can be produced by an algorithm, a control law or a pre-programmed gait sequence. Differences between these approaches include the issues of representation of knowledge, robustness with respect to changes in the environment and the facility of the selected language to represent important aspects of the control problem. For example, continuous control laws are well suited for the representation of the dynamics of fast moving walking machines, but they may require knowledge of accurate dynamic models of their behavior. Finite state machines can more easily represent the periodic sequences seen in all forms of animal and machine walking.

Once a representation is selected, the engineer must translate the observed walking behavior into the appropriate code. Such a coded representation is a difficult task and, in general, not a solved problem.

In this chapter we demonstrate an alternative approach to gait synthesis, using neural networks as the 'lan-

guage' of control. We feel that a neural representation has the following strengths:

1) Artificial neural nets (ANNs) can represent differential equations as well as finite state machines
2) ANNs can, in principle, be executed on massively parallel, highly efficient architectures, such as analog VLSI circuits
3) Biological neural nets are used by animals for gait synthesis, thus constituting a proof of sufficiency.

While biological nets are far more complex than our relatively simple ANNs, some of the architectural principles are sufficiently similar to suggest that our networks may also be sufficient for the representation of gait patterns.

Once a paradigm has been selected, a method for synthesis must be developed. Neural networks are usually programmed by using minimization of a cost function to adjust the network weights. The cost function must be selected so as to measure the deviation of the network behavior from the desired behavior. Several variants of gradient search algorithms are used to adjust the weights and minimize the cost function(1). This approach has been applied successfully to a number of problems in robotics(2).

Walking machines present a new challenge in the selection of a cost function which measures the efficiency of forward progression. Unfortunately, such a function displays multiple minima in the parameter space, and most gradient algorithms would be unable to locate a global minimum. In addition, many conceivable cost functions are not analytic and therefore derivative information must be estimated. Such estimation can be costly in high dimensional spaces. For example, an N dimensional space will require a minimum of N+1 function evaluations (even if we optimistically assume that noise is negligible) to estimate a gradient. If each functional estimate represents one experiment then we can see that this approach is impractical in all but the simplest cases.

As an alternative to deterministic gradient methods a number of stochastic methods have been suggested, including simulated annealing and genetic algorithms (GAs). We chose the GA method of simulated evolution to adjust the weights in a neural network designed to produce a CPG for a walking machine.

In adapting GAs for use in robotics tasks we must be aware that GA cost functions are invariably *simulated*. As we adhere to the belief that the world is its own best simulation (3), we choose to evaluate performance in the real world. Later in this chapter we will demonstrate the value of using the world versus simplified models. Since the robotic resource is limited, we have developed an efficient methodology for its utilization. We have developed an approach called 'staged evolution' that accelerates learning such that evaluations can be carried out on real hardware, not simulations.

In staged evolution, initial challenges impose loose constraints upon the controller. Thus, a large area of the parameter space satisfies these initial challenges, and the GA will quickly converge upon an appropriate set of parameters. Successive challenges become more specific, reducing the size of the islands (See Figure 1). It is important that the islands associated with one challenge are contained within the islands of previous challenges. The result is a progressive convergence of a sequence of controller functions to a solution set. Solutions are found to islands of computation. Once a solution is found, the next challenge is posed and the computational solution diffuses to the next island. In this way, complex behavior can be extracted in a relatively efficient manner.

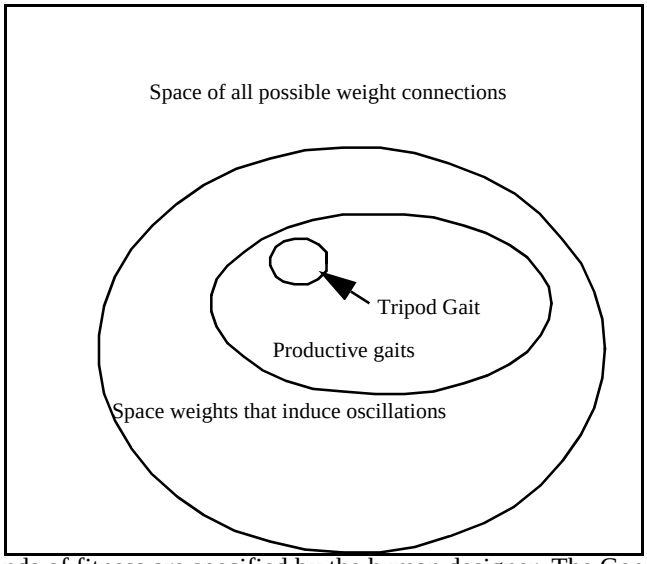

Figure 1. Islands of fitness are specified by the human designer. The Genetic Algorithm guides the system under evolution from looser to more refined controller.

2. Problem statement

The goal is to evolve a neural network to generate a sequence of signals that can drive the legs of a hexapod robot in such a way as to produce consistent forward locomotion along the body axis. Since each leg has two degrees of freedom, the hexapod is a 12 degree of freedom robot. The entire robot is kinematically similar to the Genghis robot designed by Brooks' Lab at MIT(4). This configuration was a good starting point because it has been demonstrated by Beer (5) that a relatively simple network of neurons was capable of producing the required motion for a hexapod. The system was not too complicated to daunt nascent effort, and yet was not completely trivial.

While the legs of our robots were controlled by neural networks, the use of neural units is not essential to the discussion. Finite state machines could have been used, as well as lines of 'C', or Lisp code, but neural nets have the property that small changes in structure usually lead to small changes in behavior. This is not usually the case with a network of finite state machines or bits of program code.

3. Previous work

The design of a multi-legged robot requires a strategy for coordination of the leg movements in order to obtain reliable locomotion. This sequencing or coordination of the legs can be obtained in a number of ways. The first, and most straight forward approach is to generate a sequence of leg or joint motions using something analogous to a finite state machine. The second method is to use an algorithm for the synthesis of gaits. The final method is to build biological simulations and use the output of the simulation to drive the leg joints.

Most robot designers pre-program the desired leg sequence for a particular desired gait pattern,e.g., (6, 7, 8, 9, 10, 11) . In some cases the specific sequence and the associated phase relationships between leg movements is obtained from a study of animal locomotion patterns (12). In addition, Beer (13) and his collaborators have simulated the motor pattern generators (CPGs) of certain insects and obtained a variety of insect walking patterns simply by adjusting parameters in the CPG which determine the inhibition and excitation of the coupling signals of adjacent legs. In this paper we report on a different approach, based on models of evolution.

Alexander has examined the gaits of animals and proposed optimization criteria for gaits and compares animal and robot walking in terms of efficiency(14, 15). In addition Alexander and Full have both remarked

on the use of energy storage in muscles (16, 17). It is clear that muscles systems in animals represent oscillators in themselves. There is evidence that CPGs should be analyzed as systems of coupled oscillators with the muscles systems as an oscillatory component and that the system should be studied as a dynamic whole (17).

A number of authors have analyzed central pattern generators for the control of walking (18, 19) (20).In addition, (21) has created a self-organizing system for the control of a walking insect. This systems had the nice property that individual components did not interact strongly with each other. Thus credit assignment was simplified. However, it is not clear how this system could be scaled for the design of more complex systems.

Beer (5) has also worked in the design of a simple walking insect with a neural network. Beer's approach was to reproduce the known interconnections of the cockroach's nervous system. He then selected weights by hand. It was felt that this approach would not be suitable for very complex systems either. Recently, Beer and Gallagher (22) have reported using Genetic Algorithms for the design of CPG for a simulated walking machine.

Other authors have applied genetic algorithms to the design of neural networks for control of non-walking robots. The most prolific effort seems to be from a group at the University of Sussex. Here Cliff and collaborators have applied the GA approach to the design of visually guided simulated robots(23), the design and analysis of robotics control systems (24, 25).

4. Present approach

Our approach is based on the idea of *Staged Evolution.* In staged evolution, the engineer is informed by principles of biological development.

Coghill, in his essay on the development of *Amblystoma* (26) discusses the set of stages through which this salamander 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. The forelimbs show this development earlier than the hindlimbs. 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).

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 transitions 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.

In order to adapt these principles to our problem, 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.*

When looking at the structure of an area such as the neocortex, we find a rather regular structure in the placement of particular types of neurons and the growing of their connections. In other words, our genetic information provides two things: a set of instructions for constructing several types of neurons and their local interconnections; and a description of how the local regions are connected together on a more global basis (of course, this is greatly simplified for this example).

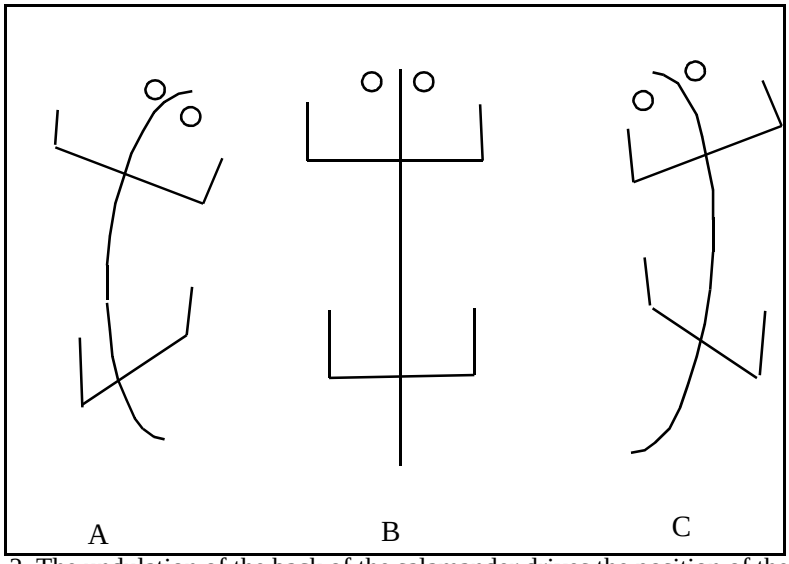

Figure 2. The undulation of the back of the salamander drives the position of the legs. The undulation enforces the necessary phase relationship to create a productive gait.

In the context of building neural controllers, we may take similar measures. The genetic string provides two types of information: how a set of neurons local to a particular leg are connected; and how these groups (which are replicated for each leg) are then connected. The local set of connections really provides a simple oscillation system that may be used to drive the two joints of the leg. The more global connections are used to coordinate the various oscillators such that an overall walking motion is produced. Note also, in this case, that the number of connections has been greatly reduced, therefore significantly reducing the size of the search space.

5. 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 Time Processor Unit (TPU).

The GA simulator used for this set of experiments is GENESIS which (27)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) (28), 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 3).

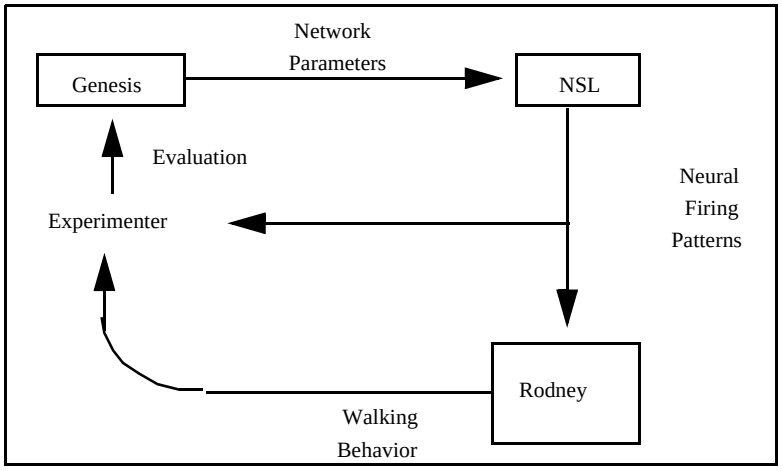

Figure 3. The Experimental Setup.

6. The Neural Model

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

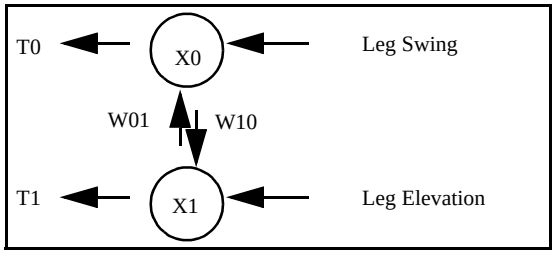

Figure 4. A oscillator circuit.

Figure 4 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

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

where t_i is the threshold, w_{ji} is the weight from unit j to unit i,
and $f(\,)$ is a sigmoidal function NSLsigm(), defined in NSL (28)as :

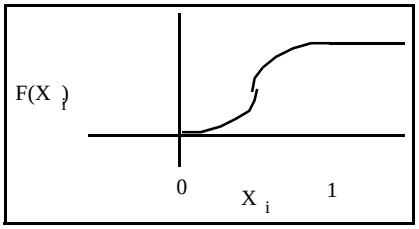

Figure 5: The Non linearity NSLsigm().

The oscillators where mapped to a Pulse Width Modulated signal that controls the position of the motors, given in degrees, by the function given in eqn (2).

$$\theta = x_i \cdot m_i + b_i \qquad (2)$$

Here the terms mi and bi are calibrated so that produces an output swing of the swing joints of

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 6).

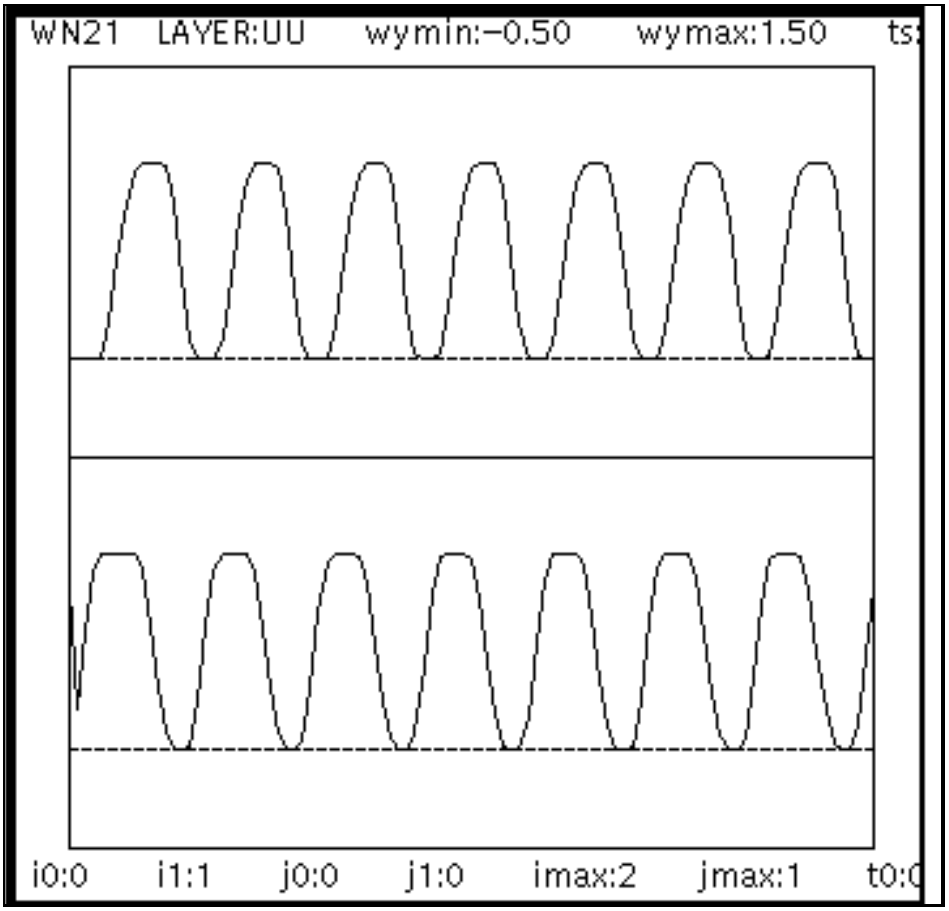

Figure 6. Neural firings for *x0 (*top) and *x1* (bottom) plotted against time.
Note that *x1* precedes *x0* by 90 degrees.

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 and they may work against each other.

This problem is approached by the addition of connections between the oscillator units (Figure 7). These connections can enforce particular temporal constraints between the units. From the work of (29, 30, 31) 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.

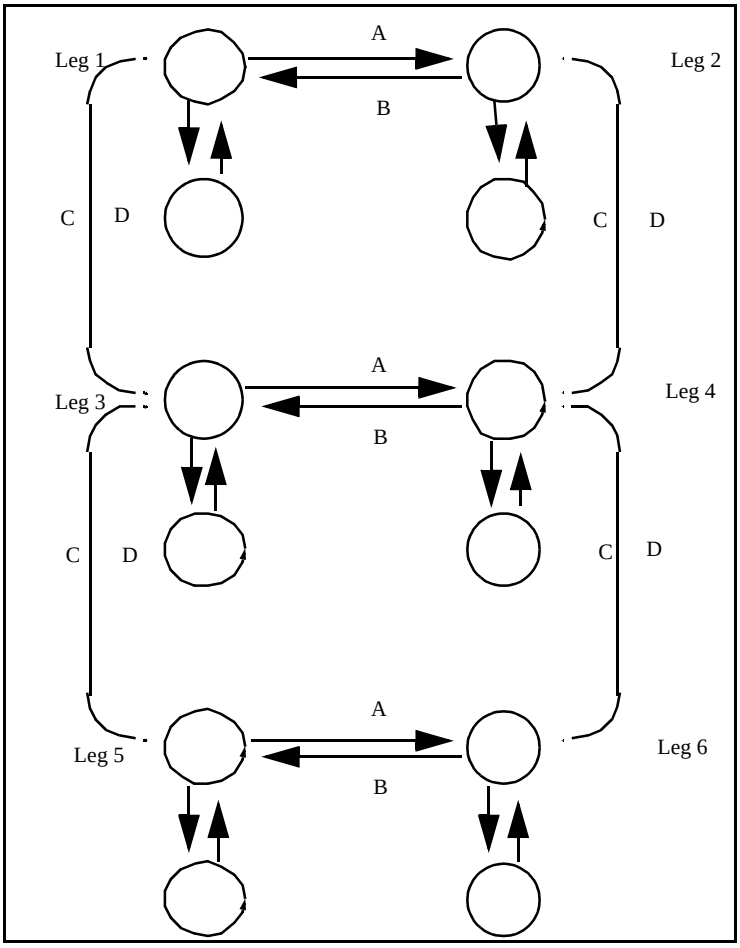

Figure 7. The Coupled Oscillator Circuit. Four additional parameters have been added to the circuit : A, B, C, D. Network symmetries are enforced by requiring several weights to take on the same value.

7. Experimental Process

7.1. 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 (See Table 1). 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.

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.

Table 1: Fitness function for oscillator evolution

Score	Description
0	Both neurons states f(xi) 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
41-59	Increasing magnitudes of oscillation.
60	Oscillation over the entire range [0,1].

7.2. 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 7. 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 simpler than during the first. Scores are assigned as follows:

Table 2: Fitness function for evolution of walking.

Score	Behavior
0	No movement (no oscillations)
$10 + L - \frac{T}{10}$	Oscillation, movement backwards
$10 + L - \alpha\frac{T}{10}$	Oscillations, movement forward

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. L is measured along a straight line, along which Rodney is initially oriented.

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 (swing forward to swing backwards), then a transition from walking backwards to walking forwards may be easily made

8. Results

The genetic string used for this set of experiments consisted of 65 bits. Eight bits were used for each of eight parameters : W01, W10, T0, T1 (single oscillator circuit), A, B, C, D (oscillator interconnections). Each parameter encodes a weight value of the range [-8.0, 8.0], using a gray 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]

When a random (normally distributed across [-8.0, 8.0]) set of inter-oscillator weights are introduced to a set of oscillators, many combinations of weights may completely suppress the oscillations in the network. This is especially the case when these random weights are large. Thus, when an experiment transitions from the first to the second phase, the inter-module weights are initialized to zero before selection and mutation/ crossover are applied to generate the first generation of the second phase. The result is relatively few non-zero initial interconnections. More substantial weights are then introduced more slowly through mutation and crossover.

For this set of experiments, the genetic parameters were set to the following values given in Table 3:

Table 3: Genetic Algorithms Parameters

Parameter	Value	Description
Population Size	10	Number of genetic strings maintained at one time
pmutation	.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 propotate to the next generation

1.The gray 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 other binary representation.

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

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.

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 8 shows a plot of the best and average scores versus generation for the first phase of learning.

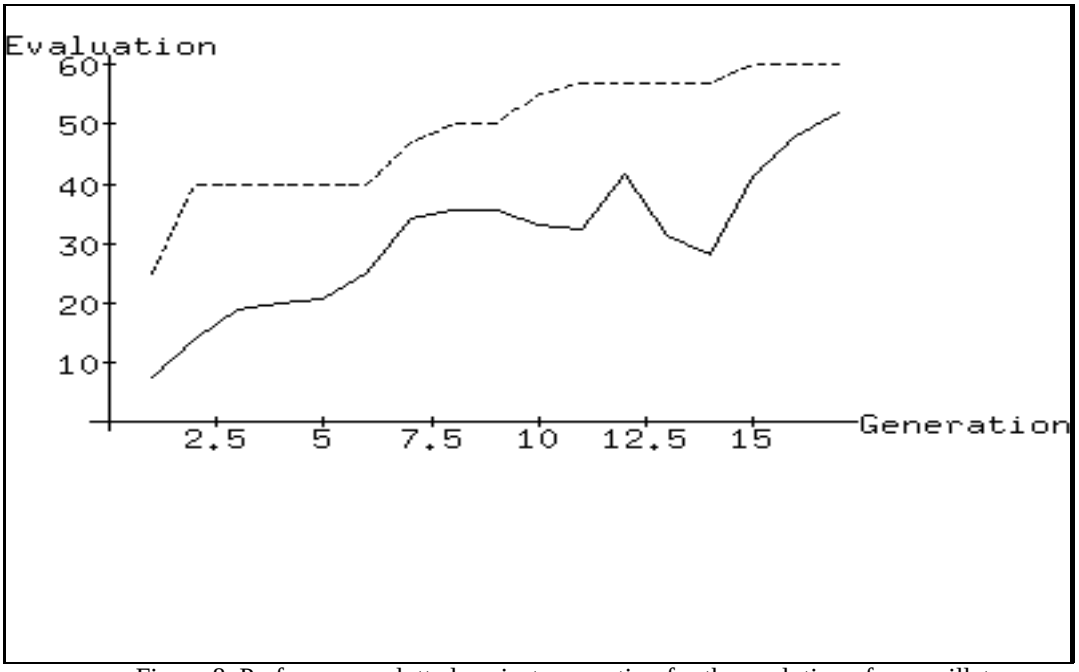

Figure 8. Performance plotted against generation for the evolution of an oscillator. The solid line shows the average performance of the population, and the dotted line represents the performance of the best individual at that generation.

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 motion of Rodney's "tail", the resulting controllers tended to turn Rodney as much as 90 to 120 degrees. In this case, the distance to Rodney's rear was maximized, but the distance traveled by the rest of the body was not as desired.

Thus far, this chapter has concentrated on the generation of a tripod gait. Other gaits besides the tripod gait have been observed in insects(13). One of the more common is the wave gait, which is characterized by rear to front waves of leg motions. In other words, reciprocal legs perform the identical movement at the same time, with more rostral (forward) legs following the lead of the caudal (posterior) legs.

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 outperforms the wave gait.

The most surprising result of our experiments was that, in all experiments (in which), Rodney preferred to walk backwards over forwards. This difference in efficiency appears to be due to angle at which the feet push against the floor in the particular mechanical structure of our robot. This effect could not have been anticipated without a very detailed analysis of the dynamics of the robot's interaction with the environment.

Further experiments, with , result in the domination of the forward walking controllers. The results of these experiments indicated that around , 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.

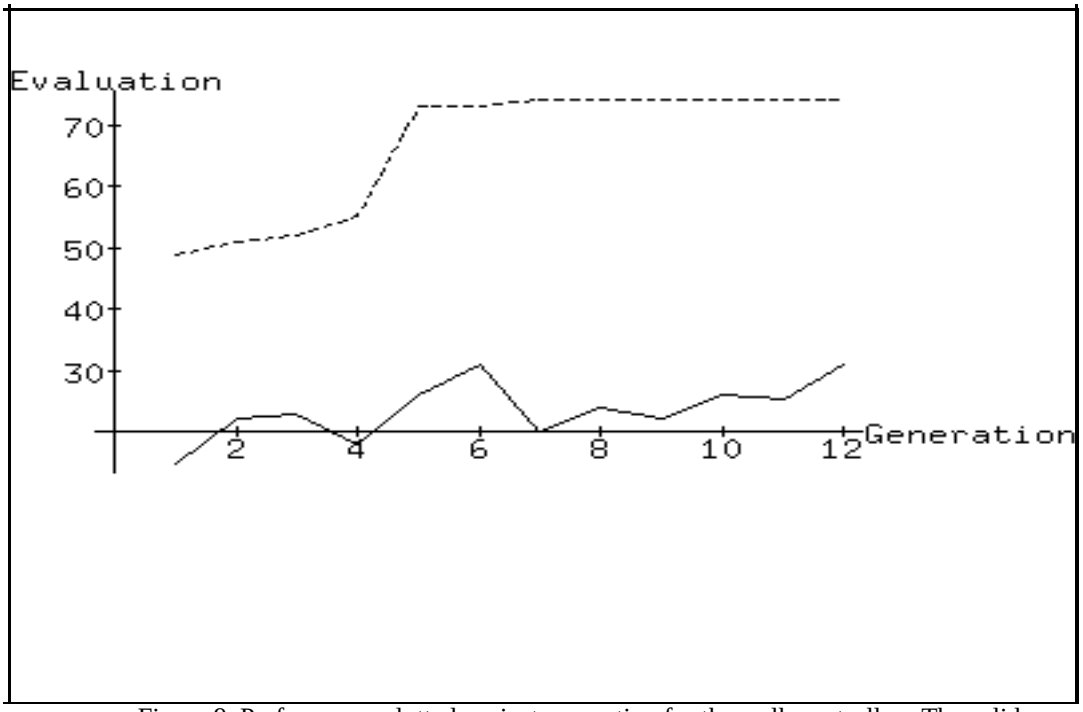

Figure 9. Performance plotted against generation for the walk controller. The solid line shows the average performance of the population, and the dotted line represents the performance of the best individual at that generation.

8.1. 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. Information from the environment can also be used to modify the behavior of the net. For instance, certain phase transitions might be signaled by input from the environment. When the robot's foot touches the ground, this may cause the robot's leg to move backward. It would be interesting to evolve such reflex action.

Application of Different Fitness Functions In nature, wave gaits are seen at low speeds

and tripod gaits are observed at high speed. It may be that the efficiency of each gait is dependent on the speed of locomotion. In the case of Rodney the fitness function emphasized speed over measures of efficiency. A more flexible Fitness function may make use of energy utilization per unit time or per distance traveled. This may lead to a greater repertoire of behaviors.

Evolution of Perceptual Systems Many lower vertebrates exhibit approach and avoidance behavior (32). 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 interesting to evolve processing modules for visual input to detect looming stimuli and prey stimuli and to evoke the appropriate motor response.

Formalization of the cost function selection procedure. Currently, the designer must rely heavily on her insight into the nature of the problem to design the proper set of cost functions. It would be beneficial to construct a set of general guidelines for selecting the evaluation functions.

Application to a 4-legged walking machine. The USC Robotics Laboratory has constructed a 4-legged walking machine. The control of this device is much more difficult than in the case of Rodney. First, 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 Second, the device may incorporate a wider array of sensors, which may include foot switches and attitude detectors. This will allow the device to negotiate rough terrain.

Simulation of the robot as a way to quickly create a general controller, which is then applied to a real system. Although we have argued for the need for working with a real robotic system during the evaluation of a controller, the simulation of the robotic system can be useful in the early stages of evolution. Simulation allows for much quicker evaluation of the robotic system, although it cannot account for many of the important effects. Once the controller performs well in the simulated environment, evaluations may be transferred to the real robotic system since the problem has been solved to a large degree already. It is during this stage that the controller can become more specific to the nuances of the real system. Such an approach may significantly reduce the time required for evolving an appropriate controller.

9. Conclusions

In this chapter we have demonstrated the use of staged evolution for the evolution of CPG for a walking machine. The use of a staged evolution approach can significantly improve convergence to a solution. This approach relies on the careful selection of intermediate solutions, or *fitness islands,* on the way to a goal set. The design of the cost functions 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) .

It would be impractical to apply Genetic Algorithms, in a straight forward manner, to the design of a neural network to control a robot. Thus the approach introduced here makes practical the use of GAs with real robots.

It should be noted that we have not proved rigorously that the "islands of convergence" methods speeds convergence. Our argument has been based on a general appeal to intuition. Future work should address this problem. The point of the chapter is to open up the question of 'why' these methods should work, and to invite further investigation.

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 backward walking solution.

If these techniques can be developed sufficiently, they may lead to a general methodology for building

highly complex robotic systems.

References

1.J. Hertz, A. Krogh, R. G. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley Publishing Co., Redwood City, 1991).

2.G. A. Bekey, K. Y. Goldberg, *Neural Networks in Robotics* (Kluwer Academic, Boston, 1993).

3.R. Brooks, *IEEE Journal of Robotics and Automation* **2**, 14-23 (1986).

4.R. Brooks, A Robot that Walks; Emergent Behaviors from a Carefully Evolved Network, IEEE International Conference on Robotics and Automation 1989), pp. 692-694.

5.R. D. Beer, *Intelligence as Adaptive Behavior* (Academic Press, San Diego, 1990).

6.R. B. McGhee, *Mathematical Biosciences* **2**, 67-84 (1968).

7.R. B. McGhee, Finite State Control of Quadraped Locomotion, Proc. Third Int. Analogue Computation Meeting Opatija, Yugoslavia, 1966),

8.S.-M. Song, K. J. Waldron, *Machines That Walk: The Adaptive Suspension Vehicle*. P. H. Wilson, J. M. Brady, Eds., The MIT Press Series in Artificial Intelligence (The MIT Press, Cambridge, 1989).

9.R. B. McGhee, in *Advances inn Automation and Robotics* G. N. Saridis, Eds. (Jai Press, Inc, Greenwich, 1985).

10.S. Hirose, O. Kunieda, *The International Journal of Robotics Research* **1991**, 3-12 (1991).

11.S. Hirose, A. Morishhima, *Int. J. Robot. Res.* **9**, 99-114 (1990).

12.R. D. Quinn, K. S. Espenschied, in *Biological Neural Networks in Invertebrate Neuroethology and Robotics* R. D. Beer, R. E. Ritzmann, T. McKenna, Eds. (Academic Press, Inc, Boston, 1993).

13.R. D. Beer, H. J. Chiel, L. S. Sterling, *American Scientist* **79**, 444-452 (1991).

14.R. M. Alexander, *Physiological Reviews* **4**, 1199-1227 (1989).

15.R. M. Alexander, *The International Journal of Robotics Research* **3**, (1984).

16.R. M. Alexander, *Int. J. Robot. Res.* **9**, 53-61 (1990).

17.R. J. Full, in *Biological Neural Networks in Invertebrate Neuroethology and Robotics* R. D. Beer, R. E. Ritzmann, T. McKenna, Eds. (Academic Press, San Diego, 1993) pp. 3-20.

18.J. S. Bay, H. Hemami, *IEEE Transactions on Biomedical Engineering* **BME-34**, 279-306 (1987).

19.R. Beer, H. J. Chiel, L. S. Sterling, in *Designing Autonomous Agents* P. Maes, Eds. (MIT Press, 1991) pp. 169-186.

20.D. Graham, *Biological Cybernetics* **26**, 187-198 (1977).

21.P. Maes, R. Brooks, Learning to Coordinate Behaviors, Proceedings of the AAAI 1990), pp. 796-802.

22.R. Beer, J. C. Gallagher, *Adaptive Behavior* **1**, 91-122 (1992).

23.D. Cliff, P. Husbands, I. Harvey, University of Sussex, Evolving Visually Guided Robots (1992).

24.D. Cliff, P. Husbands, I. Harvey, University of Sussex, Analysis of Evolved Sensory-Motor Controllers

(1992).

25.D. Cliff, I. Harvey, P. Husbands, University of Sussex, Incremental Evolution of Neural Networks Architectures for Adaptive Behaviour (1992).

26.G. E. Coghill, *Anatomy and the Problem of Behavior; Lectures Delivered at University College, London* (Hafner Pub. Co., New York, 1964).

27.J. J. Grefenstette, Naval Center for Applied Research, A User's Guide to GENESIS (1987).

28.A. Weitzenfeld, University of Southern California, NSL Neural Simulation Language (1991).

29.N. Kopell, in *Neural Control of Rhythmic Movements in Vertebrates* A. H. Cohen, S. Rossignol, S. Grillner, Eds. (Wiley-Interscience, 1988).

30.R. H. Rand, A. H. Cohen, P. J. Holmes, in *Neural Control of Rhythmic Movements in Vertebrates* A. H. Cohen, S. Rossignol, S. Grillner, Eds. (Wiley-Interscience, 1988).

31.G. Rinzel, B. Ermentrout, in *Methods in Neuronal Modeling: From Synapses to Networks* C. Koch, I. Segev, Eds. (MIT Press, 1989).

32.M. A. Arbib, *The Metaphorical Brain 2: Neural Networks and Beyond* (John Wiley & Sons, New York, 1989).