EVOLVING NEURAL NETWORK CONTROLLERS TO PRODUCE LEG CYCLES FOR GAIT GENERATION

GARY B. PARKER and ZHIYI LI
Computer Science, Connecticut College, New London, CT 06320
parker@conncoll.edu and zli@conncoll.edu

ABSTRACT

The generation of gaits for hexapod locomotion controllers can be divided into two main parts: the cyclic action of a single leg (leg cycles) and the coordination of all legs to combine individual leg cycles to produce forward movement. In this paper, we use a genetic algorithm (GA) to evolve the structure of an artificial neural network (NN) that produces leg cycles in a hexapod robot. The movement of the robot's leg is controlled by a horizontal servo and vertical servo. The servos are controlled by a NN that generates a cycle of pulses. With minimal restrictions on the structure of the NN a GA is used to find the parameters of neurons and the connections between them. The pulse sequences generated by the evolved NNs resulted in leg cycles that produced efficient forward movement.

KEYWORDS: genetic, learning, neural network, robot, hexapod, control

INTRODUCTION

Because movement and exploration of the environment is fundamental to many other behaviors of an autonomous robot, the design of locomotion is important. In the animal world, the six-legged insect is a successful model that shows great robustness, fitness, and flexibility to variations in the environment and its load. It is difficult to simulate the biological leg movement coordinating system because the biological neural system structure is much more sophisticated than what we can reasonably model using mechanical and electronic devices. Nevertheless, researchers have developed artificial hexapod walking controllers with various levels of complexity and capability. The generation of gaits for these controllers can be divided into two main parts: the cyclic action of a single leg, which we will call a *leg cycle* in this paper, and the coordination of all legs to combine individual leg cycles to effective forward movement. Generating smooth leg cycles is critical and fundamental for generating successful gaits for a robot.

Previous research works on the control of hexapod leg movement are illuminating. Beer and Gallagher used Genetic Algorithms (GA) to develop neural network (NN) controllers for a hexapod [1]. The controller had foot, backward swing, and forward swing motor neurons; two internal unspecified neurons and a single angle sensing neuron. The angle sensing neuron was connected to all 5 of the other neurons and those 5 were fully connected to each other. The 5 control neurons also had a threshold and time constant parameter. After an individual leg cycle was learned for a generic leg, six copies of the leg

controller were made. Each of the leg controllers were connected to their neighbor leg controllers. A GA was used to learn the connection weights between the leg NNs. Lewis, Fagg, and Solidum used this staged evolution also [2]. In their model, the position of each of the two leg joints was driven by the state of a neuron. The two neurons controlling the leg were the leg swing neuron and the leg elevation neuron. The two neurons formed an oscillator, and the oscillators were mapped to a pulse width modulated signal that controlled the position of the motors. A GA was used to find the parameters of this two-neuron network. Initially, the NN started at random values but within several cycles the two neurons fell into an oscillatory pattern, with a phase difference of 90 degrees. Then a network of these oscillators was evolved by using a GA to coordinate the movements of the different legs. In Parker's hexapod gait generation, instead of the NN controller approach, a Cyclic Genetic Algorithm (CGA) [3] was used to learn the pulse sequences that were needed to control servos on each leg [4]. These pulse sequences, that took into consideration the peculiarities of the leg's capabilities, were sent to the servos to generate leg cycles. After each leg was trained, a GA was applied to produce gaits.

In this paper, the structure of a NN is learned that will control the two servos of a leg to produce a leg cycle. Compared with the works discussed above, this research has its own uniqueness described as follows. In the Beer and Gallagher model, the neural controller was designed at a macro level in which the output of neurons represented complete actions such as forward swing, backward swing and foot-down. The GA was used to find the appropriate associated weights and thresholds. Our approach is similar to this except that we have six neurons with two producing control pulses and two sensor inputs marking horizontal and vertical leg extremes. But the main difference is that the outputs of the neurons are pulse widths that control the angular positions of each leg servo.

In Lewis, Fagg, and Solidum's work, they specifically defined the controller of each leg as a simple two-neuron oscillator that generated outputs that were mapped into pulse widths and then sent to servos. The GA was used to learn the weights and thresholds. In the work presented in this paper, the total neural structure will evolve, including the connections, associated weights, and neuron thresholds. We expect that the GA will generate a proper NN that will produce an optimal leg cycle. In addition, this work differs from these works in that the leg cycle learned is for a specific leg taking into account its particular capabilities.

In Parker's CGA work, the leg cycle pulse sequence for the servos of a specific leg is learned as a cycle of instructions. In this paper, it will be generated as the output of the leg controller NN found by a GA. The resultant NN controller will be implemented within a BASIC Stamp II (sold by Parallex, Inc.). And the output of the NN will be sent to the servos to control the movement of a leg.

SERVOBOT

The robot used is ServBot [4], which is a simple six-legged robot with two degrees of freedom per leg. Each leg has two servos that provide vertical and horizontal movements. The servo takes as input a pulse sent from the leg controller (a Basic STAMP II) that will direct it to be set to a specific angular position. The pulse should be repeated every 25ms for the servo to maintain a constant position. The duration of the pulse determines the position. Pulse widths from 0 to 3000 μs cover the full range of angular positions for each servo. These pulses can be converted into leg positions for both the vertical and horizontal servos. Figure 1 shows the relationship between pulse widths and the position

they determine for a particular leg. These are measured capabilities of a robot's specific leg. The Neuron Output column will be explained in the next section. The Pulse Width column shows several possible pulse widths that can control the servo. The X and Y positions show the resultant leg position due to the movement of the horizontal and vertical servos associated with each pulse width. We define the full forward position as the 0 for X, the full downward position as 0 for Y. A pulse with a width of 600μs sent to the horizontal servo will drive it to an angle that will result in the leg moving back 16mm measured at the foot. The same pulse sent to the vertical servo will result in the leg lifting the foot to 8mm off the ground.

Neuron Output	Pulse Width (μs)	X-Position (mm)	Y-Position (mm)
0	0	0	0
1	200	1	0
2	400	7	3
3	600	16	8
4	800	28	14
5	1000	41	21
6	1200	52	28
7	1400	63	34
8	1600	71	39
9	1800	75	43
10	2000	77	45
11	2200	76	48
12	2400	73	49
13	2600	73	49
14	2800	73	49
15	3000	73	49

Figure 1: The neuron output with its corresponding pulse width and the associated X (horizontal servo) and Y (vertical servo) positions.

If the difference between two consecutive pulses is too much, the servo cannot move the leg fast enough to reach the desired position indicated in Figure 1. As a result, it will move the leg at the fastest speed it is capable of (through experimentation, it was found that this fastest speed will result in a maximum leg movement of 5mm in one pulse). For example, if the original pulses are (0, 0), corresponding to (horizontal servo, vertical servo), putting the leg at x, y position (0, 0) and the next pulses sent are (600, 0), the associated position should be (16, 0). However, because this exceeds the servo's capability, the leg will only move to (5, 0).

THE NEURAL NETWORK CONTROLLER

In order to make the two servos for a leg work together and generate appropriate pulse sequences, a NN model was used. The structure of this NN, including the thresholds of neurons and associated connection weights, was learned by a GA. The resultant NN will be implemented within a BASIC Stamp II. The neurons with their thresholds and connections are represented by variables and a program will simulate the stimulation and firing of the neurons. The output of the neural controller will be sent to the servos every 25ms. In future work, there will be 7 such neural controllers, one for each leg and one as a coordinator, and they will be connected by their input and output pins. Through further evolution, the leg controllers will coordinate with each other and generate gaits.

Since the NN will eventually be implemented within a BASIC Stamp II, the limita-

tions of the chip, such as memory and integer range, forces us to make restrictions on the components of the neural model. A description of the NN follows:

1. The coordinator controller will send a signal to all of the leg controllers every 25ms (we refer to this as an iteration). Upon receipt of this signal, internal computations using actual external connections and internal connections will be computed and the new internal values will be set as external signals are sent. Two of these external signals from each controller will go to the leg's servos to control their angular position and subsequently set the linear positions of the leg.

2. Each leg controller will have 6 neurons that are fully connected to each other and connected to the possible external connections. External connections are actual through wiring between pins and internal connections are simulated through variables holding the output.

3. Each neuron has two thresholds: threshold-low and threshold-high. During each iteration the NN changes its state, each neuron will accumulate the effects of those that are connected with it and of the sensors. This is called the accumulation value. The output of a neuron in each iteration is determined by its accumulation value and its two thresholds. The maximum output for a neuron is 15 and the minimum value is 0. If the accumulation of a neuron is smaller than threshold-low, it will fire a 0. If the accumulation of a neuron is bigger than threshold-high, it will fire a 15. If the accumulation of a neuron is between the two, a linear function will transform it to a value between 0 and 15 that will be fired (Figure 2).

```
O = 15*(A-T1)/(T2-T1)
O: output of a neuron
A: accumulation of a neuron
T1: the lower threshold
T2: the higher threshold
```

Figure 2: The linear function determining the output of a neuron.

In case threshold-high is smaller than threshold-low, the neuron will fire 15 when the accumulation is larger than threshold-low and fire 0 when the accumulation is smaller than threshold-low. The selection of the 0-15 range is to allow efficient storage in the BASIC Stamp II. With this range, we can use a nibble (4bits) to represent the output from a neuron.

4. Due to memory limitations within a BASIC Stamp II, the number of neurons is limited to six (Neuron-0 to Neuron-5). Six should be sufficient for the GA to evolve freely. It is not know what role each neuron will play in the controlling process and there is no concern with whether a neuron is useful or not, just that enough neurons are made available to solve the problem. Each neuron's connections with other neurons and associated weights are left for the GA to find. It is possible that some neurons will not be used in the final result. The connection weights are within the range [-15, 15]. If it is 0, there is no connection in between; if it is positive, the effect from the source to target neuron is positive; if it is negative, the effect from the source to target neuron is negative.

5. There are two sensors. Sensor-1 monitors horizontal extreme positions and sensor-2 monitors vertical extreme positions. When the leg hits its full backward position, sensor-1 will fire 15 and keeps firing it until the leg hits its full forward position, where the sensor starts to fire 0. When the leg hits its highest position, sensor-2 will fire 15 until the leg hits its lowest position where the sensor starts to fire 0. The sensors are connected to all

of the neurons. The weights, which are also within the range [-15, 15], are left to the GA to find.

6. The NN will start at an initial state (defined by the GA) and then it will run for 500 iterations, i.e. it will change its state 500 times. Each iteration is started by a signal from the coordinator controller (the 7th BASIC Stamp II). The signal will be sent every 25ms. After the NN gets this signal, it will evaluate itself according to its present state and calculate what the next neuron values should be.

7. For each iteration, outputs from Neuron-0 and Neuron-1, which is within the range [0, 15], will be mapped to associated pulse widths [0, 3000] by multiplying it by 200μs and send to the two servos to control the movement of the leg.

GENETIC ALGORITHM

A GA was used to find the structure, including the thresholds of neurons and associated connection weight, of the NN controller.

Chromosome

The individual chromosome used was a series of numbers that describe the NN controller. It is described in Figure 3. Each individual in the population was evaluated as the neural structure it represents.

```
((n0 t01 t02 w00 w01 w02 w03 w04 w05 ws10 ws20)
 (n1 t11 t12 w10 w11 w12 w13 w14 w15 ws11 ws21)
 (n2 t21 t22 w20 w21 w22 w23 w24 w25 ws12 ws22)
 (n3 t31 t32 w30 w31 w32 w33 w34 w35 ws13 ws23)
 (n4 t41 t42 w40 w41 w42 w43 w44 w45 ws14 ws24)
 (n5 t51 t52 w50 w51 w52 w53 w54 w55 ws15 ws25))
```

`ni:` the original value of the ith neuron. [0, 15]
`ti1`: the lower threshold of the ith neuron [-400, 400]
`ti2:` the higher threshold of the ith neuron [-400, 400]
`wij:` the connection weight from the ith neuron to the jth neuron [-15, 15]
`ws1i:` the connection weight from sensor-1 to the ith neuron [-15, 15]
`ws2i:` the connection weight from sensor-2 to the ith neuron [-15, 15]

Figure3: A chromosome representing a neural network.

Training

The software package GENESIS5.0 [5] was used for training. Five individual tests were conducted using different starting populations. A population of 80 chromosomes was randomly generated. Each chromosome represented a NN structure. The NN ran 500 iterations and thus generated a sequence of 500 pulses for each of the two servos. The two sequences were evaluated with a fitness assigned. Fitnesses consisted of three factors: forward movement, number of times raising the leg, and drag generated. Forward movement was calculated by the movement generated when the leg was on the ground. The number of times the leg was raised and lowered was penalized because it wasted energy and reduced the effect of the forward movement generated. The drag generated was a penalty due to a leg staying on the ground with its full backward position, which would just cause drag.

RESULTS

Figure 4 shows the results of single leg training done on one of the robot's legs. All 5 trials are shown. As can be seen, in all cases the fitness increases quickly at the beginning. After 60 generations cycles develop, but they were typically shorter strides than what would be considered optimal. Since these short cycles were effective in producing forward movement and since minor changes at this point result in lower fitnesses, the GA would get stuck in these local maxima. However, in two of the cases the GA continued to improve and produced better cycles with longer strides.

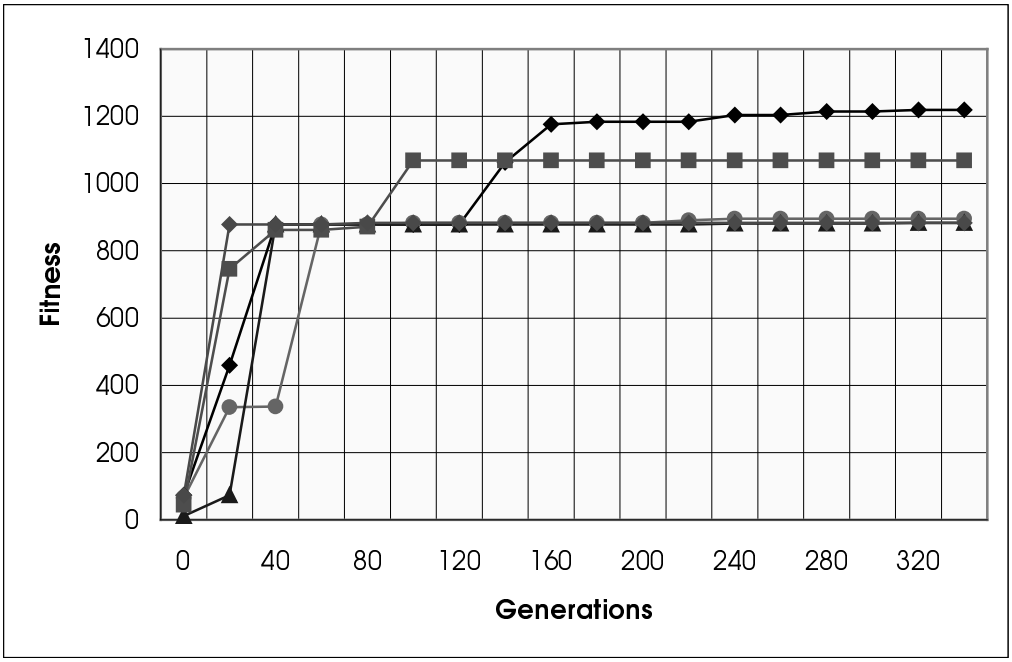


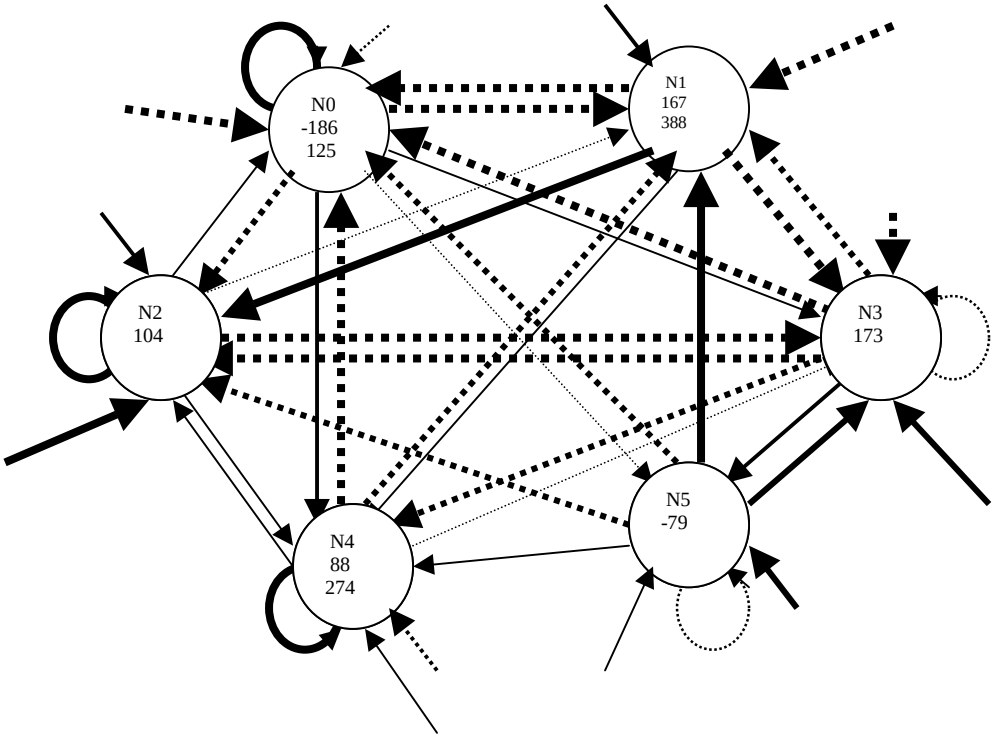
Figure 4. Single leg training of the five distinct populations. The fitness of the best individual at every 20 generations of training is shown.

Figure 5 shows a graphic representation of the best NN structure learned. It produced a sequence of pulses to the servos that resulted in a near optimal leg cycle. At the beginning, with the leg being full forward and full down, this controller moves the leg back while keeping it on the ground. This continues for 15 pulses until the leg arrives at the full backward position. At that point, the controller goes into a cycle of 22 pulses, which produces a continual leg cycle.

CONCLUSION

Using a GA to evolve a NN controller that generates appropriate pulses is an effective approach to produce leg cycles for the ServoBot. By allowing the GA to learn the structure, the resultant NN controller will be better suited for the task, plus leave the system with enough flexibility to adapt to changing situations. In future work, after each leg is

evolved to its proper leg cycle, the NN controllers will be implemented within BASIC Stamp II controllers and connected to each other by wires between the I/O pins of the microprocessor. We expect to use a GA to find the proper connection weights by which the legs will be coordinated and a gait cycle generated.



The lines are positive weights. The dotted lines are negative weights.
The widths of arrows represent their magnitude.
The output from N0 goes to the horizontal servo.
The output from N1 goes to the vertical servo.
The short arrows without sources are connections from sensor 1 (horizontal sensor).
The long arrows without sources are connections from sensor 2 (vertical sensor).

Figure 5. A graphic representation of a best individual neural network.

REFERENCES

1. Beer, R. D., and Gallagher, J. C. "Evolving Dynamic Neural Networks for Adaptive Behavior." *Adaptive Behavior*, 1 (1992) 91-122.
2. Lewis, M. A., Fagg, A. H., and Solidum A. "Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot" *1992 IEEE International Conference on Robotics and Automation*, (1992) 2618-23.
3. Parker, G. and Rawlins, G. "Cyclic Genetic Algorithms for the Locomotion of Hexapod Robots." *Proceedings of the World Automation Congress (WAC'96), Volume 3, Robotic and Manufacturing Systems*. (1996), 617-622.
4. Parker, G. "Evolving Leg Cycles to Produce Hexapod Gaits." *Proceedings of the World Automation Congress (WAC'00), Volume 10, Robotic and Manufacturing Systems* (2000) 250-255.
5. Grefenstette, J. J. GENESIS5.0 Copyright © 1990.