METACHRONAL WAVE GAIT GENERATION FOR HEXAPOD ROBOTS

GARY B. PARKER and JONATHAN W. MILLS
Department of Computer Science, Indiana University, Bloomington, IN 47405
gaparker@cs.indiana.edu, jwmills@cs.indiana.edu

ABSTRACT

The metachronal wave is a gait used by most terrestrial arthropods. Each leg only lifts when the leg behind it is on the ground in position to support the animal's weight. The wave starts from the back as the hind leg lifts and is transferred forward as each leg in turn lifts. At its fastest, it is the tripod gait. In its slowest smooth movement form, only one leg at a time is lifted on each side. This provides increased stability and decreased energy expenditure. Development of stable, smoothly moving metachronal waves for legged robots can provide these same advantages. In this paper we use the Cyclic Genetic Algorithms with an adjustable speed for the robot's legs to produce the desired metachronal wave.

KEYWORDS: genetic, robot, hexapod, gait, control, cyclic, metachronal

INTRODUCTION

Generating gaits is important to successfully develop adaptive autonomous legged robots. Learning algorithms based on evolutionary computation have been used to evolve gaits for hexapod robots. Those striving for maximum speed have been used to generate tripod gaits, which are considered optimal for speed while maintaining static stability. Beer and Gallagher [1] used genetic algorithms to evolve the weights needed for a neural net controller. Spencer [2] used Genetic Programming to evolve gaits for maximal speed. In previous work [3] we developed Cyclic Genetic Algorithms (CGA), which generate the proper sequence of primitive instructions that when continually repeated produce a gait. Tests showed that these algorithms quickly converge to produce optimal tripod gaits on a robot simulation. These gaits were successfully transferred to an actual semi-autonomous hexapod robot [4].

Although maximum speed is often desirable, there are situations where it is secondary in importance to stability. Having more legs on the ground at all times is one way to increase stability. Most terrestrial arthropods use what is called the metachronal wave to produce stable forward movement. Each leg only lifts when the leg behind it is on the ground in position to support the animal's weight. The wave starts from the back as the hind leg lifts and is transferred forward as each leg in turn lifts. At its fastest, it is the tripod gait. In its slowest smooth movement form, only one leg at a time is lifted on each side. This provides increased stability and decreased energy expenditure.

Development of smoothly moving metachronal waves for legged robots provides locomotion that is continuously stable, yet is at least as fast as maximum speed. Robots that use metachronal gaits have increased load-bearing capabilities because more legs are on the ground at any moment in time. In this paper, we use a CGA to produce the desired metachronal wave for hexapod robots.

METHOD

The general method consists of three steps. The first is to take capability measurements of the actual robot for which the gaits are produced. A capability measurement records the position extremes, which define the range of movement, and the rate of sweep of the robot's legs. These measurements are stored in a simple data structure that models the robot's range of possible gaits. The second step is to train the robot model using a CGA. All initial values of the gait-defining chromosome are set at random. The final step is to test the resulting gaits, comparing predicted performance with actual and examining the sequence in which the legs moved to judge optimality.

The robot used was the ServoBot, which was developed at IU by David Braun and Ingo Cyliax for legged robot experimentation. It is an inexpensive hexapod robot that has many of the motion characteristics of larger complex robots. A control sequence is transmitted to a field programmable gait array (FPGA) through lines that connect to a Sparc workstation. Once the control sequence is transmitted, the link can be disconnected. The FPGA will execute the sequence of primitive instructions downloaded. For these experiments, each leg's rate of backward movement is set to half that of its forward movement. With proper sequencing, this provides enough time for each leg to reposition before it is needed for the next step.

The model is a data structure (Figure 1) that can hold the capabilities and current position of each leg needed to determine the state of the legs and the subsequent movement calculated from the control activation input. Measurements to fill the position fields are taken by activating each control individually and recording the leg's maximum throw. An average rate per activation is calculated for horizontal and vertical movement by dividing the maximum throw by the minimum number of activations required to attain it.

Model data structure fields for each leg:
current up -- current vertical position of leg
max up -- position off ground when full up
max down -- position off ground when full down
current back -- current horizontal position of leg
max back -- position relative to full forward when full back
Fields applicable to all legs:
rate up/down -- rate of vertical movement when actuator excited/relaxed
rate back/forward -- rate of horizontal movement when actuator excited/relaxed

Figure 1: Model Data Structure

CYCLIC GENETIC ALGORITHMS

A Cyclic Genetic Algorithm (CGA) is an evolutionary learning algorithm specialized to solve tasks that require continual cycles of sequential instructions. CGAs are based on Genetic Algorithms, which were introduced by John Holland [5], and use the standard selection, crossover and mutation operators. They differ in that the genes of their chromosome represent a cycle of repeated tasks (Figure 2) that are to be completed in a predetermined segment of time. For our purposes, the task to be completed is a sequence of primitive instructions that will result in the desired gait.

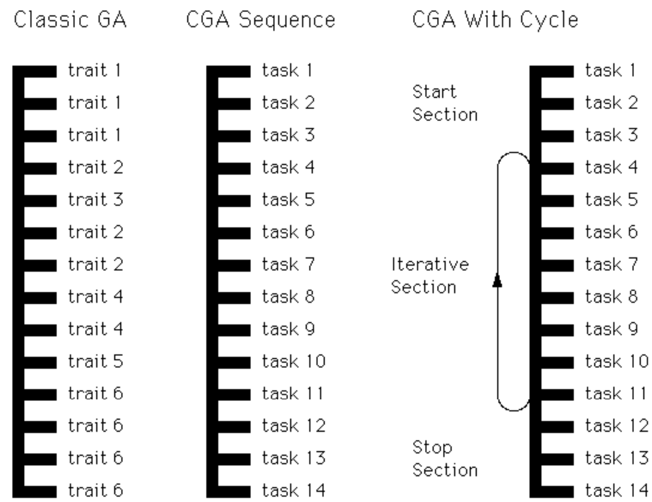

Figure 2: General Chromosome Structure of Genetic Algorithms and Cyclic Genetic Algorithms

Chromosome

The chromosome structure was designed to generate insect-like gaits for the hexapod robot. Individual legs of insects are modeled as being in either of two stages during walking: swing or stance phase. In the stance phase the leg is on the ground and moving back causing a forward force on the insect. In the swing phase the leg is up and moving forward to position for another step. Individual legs are coordinated with the other legs by setting parameters that determine when the leg should shift from the stance phase to the swing phase and back again. Coordinated movement is affected by several factors including the leg's capabilities, current position and load. The load and position of other legs also affect when legs will swing. Without the proper load on neighboring legs, the swing will be inhibited, which helps to evenly distribute the insect's base of support.

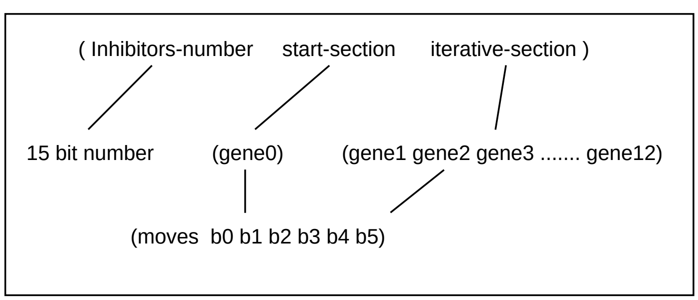

Figure 3: Specific Chromosome Structure

Each chromosome has three parts (shown in Figure 3). The inhibitors section is contained in 15 bits, and allows the CGA to evolve the required inhibitors to produce a gait where load bearing is evenly distributed among the legs. The start section provided a means for pre-positioning the legs to begin cyclic gait motion. The iterative section contained the tasks that are to be continually repeated. The start section contained one gene and the iterative section contained 12. Each gene was made up of two parts; the moves number and the stance/swing indicators (one for each leg, shown as b0 through b5). The stance/swing indicators designate whether the leg should be in the stance of swing phase and the moves number (5 bit) indicates how many moves (100 msec time segments) to repeat the indicated phase.

Inhibitors

As in insects, the inhibitors prevented the swing of a leg if specific other legs were in swing mode. For insects, this happens between adjacent legs on each side and between contralateral legs on the same segment. In our training, we did not designate the legs that would inhibit each other. This evolved in conjunction with the stance/swing ordering. The inhibitors for the set of legs were stored in a single 15 bit number (one bit per possible pair). Each bit designated if there would be an inhibition between two legs. For example, the 2,3 inhibitor allows leg 2 to swing, but not leg 3, prevented both legs 2 and 3 from going into swing at the same time.

Genetic Operators

Fitness was calculated by testing each of the population's 64 individuals (an individual is made up of a single chromosome) on the robot model. Selection for parenting future generations was predicated on these fitnesses. The higher an individual's fitness, the greater the chance that it would be selected to cross with another individual.

Crossover could happen between the sections of the chromosome or happen within each section. In the iterative section, since it could be considered a circle, crossover was performed at two points; equivalent positions in both chromosomes. The effect was to swap sections within the circle. An alternate type of crossover was a gene-by-gene crossover, which would perform crossover in each of the corresponding genes of the two chromosomes. Crosses could happen between the individual members of the list or within the bits of the specific numbers in the list.

Two types of mutation were used: 1) Gene replacement -- each gene had a random chance of being replaced by a new completely random gene. 2) Gene mutation -- each part of the gene had a random chance of having one of its bits altered.

An additional operator was the Gene-by-Gene Evaluation. This was a clean up operator that randomly picked one or two individuals from the population on each set of trails and examined each gene one at a time. Genes were evaluated on the whole and move-by-move by comparing the previous move fitness to the present. Genes that were worse than a preset minimum were eliminated. Genes that were good in the execution of their early repetitions and subsequently dropped below a threshold in the later repetitions were modified by reducing their moves number. Genes that had zero repetitions were moved to the end of the chromosome so that only active genes were at the start of the iterative section. Following these eliminations, if the number of active genes (moves of one or more) or the total number of gene moves fell below some threshold, additional random genes were added until the thresholds were met.

TESTS

Gaits were generated by running a CGA for 500 generations on five starting random populations of 64 individuals. All sections of the chromosome were initialized with a random number within the appropriate range. Intermediate gaits at increments of 10 generations for up to 100 generations were stored. Performance tests on simulations were done to determine the fitness of the resulting populations from each of the five starting populations.

RESULTS

The results show that the CGA generated stable and smooth metachronal wave gaits. The algorithm converged quickly as can be seen in Figure 4, which shows the average fitness of the 5 initial populations at each stage in learning. The average fitness at 100 generations was 212. At 200 generations of training, the majority of the test populations reached their peak with an average fitness of 232.

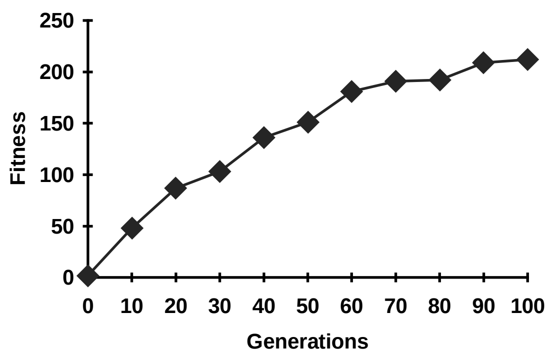

Figure 4: Average Fitness of Five Test Populations Over First 100 Generations

Each of the starting populations settled in one of two categories of gait. Shown in Figure 5 with the progression of configurations going from bottom to top. The solid lines indicate legs in stance; the dashed lines are legs in swing. Both gaits were judged to be near optimal for the model. The gait on the left is an optimal metachronal wave. Adjacent and corresponding segment legs are inhibited and four legs are always kept on the ground. The gait on the right produces near-optimal movement for the model, but is not metachronal. Testing on the actual robot will be required to determine if the second gait is as viable for robot locomotion as the first.

To check the CGA's ability to compensate for reduced capabilities in the legs, tests were done on models that had one leg's stance throw reduced by 50%. Tests were done with a random initial population and with populations already trained using the fully capable robot model. In both cases, similar near-optimal gaits were produced. When training started with a random population it took 200 generations. When training started with an evolved population, only 100 generations were required. Compensation for damage included shortening all cycles, lifting the affected leg when it was no longer of use (going down to three legs on the ground), and shifting to a tripod (only three legs always on the ground) where the short throw leg's cycle was only two-thirds in duration compared to the other cycle.

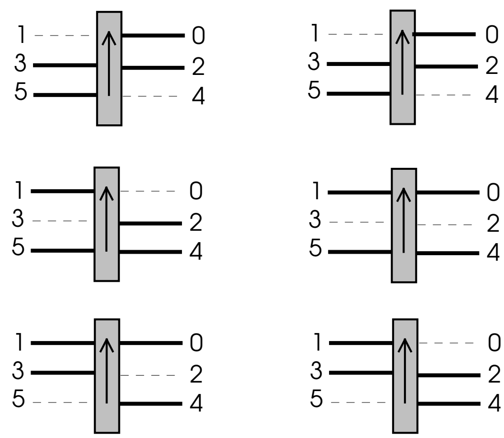

Figure 5: Produced Gaits (Metachronal and Similar)

CONCLUSIONS

CGAs can successfully evolve the two functional gaits of insect-like robots, the metachronal wave and the tripod gaits. The model-generated tripod gaits have been tested successfully on the hexapod robot ServoBot. The metachronal wave gaits are in the process of being tested on the ServoBot. We expect that these tests will confirm that the model-generated metachronal wave gait will work as well as the tripod gait. Future work must address the integration of this low-level cyclic behavior with CGA-evolved higher-level cyclic behavior, for example, the repetitive foraging loops of ants.

ACKNOWLEDGMENTS

This research was supported in part by NSF Graduate Research Traineeship Grant GER93-54898.

REFERENCES

1. Beer, R. D., and Gallagher, J. C. (1992). "Evolving Dynamical Neural Networks for Adaptive Behavior." Adaptive Behavior, 1 (pp. 91-122). Cambridge: MIT Press.
2. Spencer, G. (1994). "Automatic Generation of Programs for Crawling and Walking." Advances in Genetic Programming. (pp. 335-353) K. Kinnear, Jr. (ed.), Cambridge, Ma: MIT Press.
3. Parker, G. and Rawlins, G. (1996). "Cyclic Genetic Algorithms for the Locomotion of Hexapod Robots." Proceedings of the World Automation Congress (WAC '96), Volume 3, Robotic and Manufacturing Systems. (pp. 617-622).
4. Parker, G., Braun, D., and Cyliax I. (1997). "Evolving Hexapod Gaits Using a Cyclic Genetic Algorithm." Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC'97). (pp. 141-144).
5. Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor, Mi: The University of Michigan Press.