CYCLIC GENETIC ALGORITHMS FOR THE LOCOMOTION OF HEXAPOD ROBOTS

GARY B. PARKER and GREGORY J. E. RAWLINS
Department of Computer Science, Indiana University, Bloomington, IN 47405
gaparker@cs.indiana.edu, rawlins@cs.indiana.edu

ABSTRACT

Robotics control problems, such as gait coordination, require sequential solutions where a series of actions is continually repeated. Genetic Algorithms that do parameter optimization have not been widely applied to these cyclic sequential decision problems; although some form of evolutionary computation would be well suited for the adaptability required. In this paper we introduce Cyclic Genetic Algorithms, which were developed precisely for this purpose. The specific problem addressed, adaptive gait development for hexapod robots, was the impetus for this new kind of evolutionary computation, but it can be applied to other robotics domains.

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

INTRODUCTION

Six legged robots have been developed with varying degrees of capability, complexity, and expense. Gait establishment has been an issue and in general has been addressed through the employment of mostly static approaches. Donner [1] authored a special programming language, Brooks [2] used subsumption architecture, Beer and Gallagher [3] used a neural net, and Spencer [4] used Genetic Programming. Our goal was to develop a learning system that used limited "a priori" information, was continually adaptive to robot capabilities, and could be used to eventually train an actual robot. In addition, we wanted it to be a quickly converging algorithm suitable for any-time learning [5]. Cyclic Genetic Algorithms were developed with four variations that were tested and found capable of producing reasonable gaits. One variation produced the optimal tripod gait and was robust enough to adapt to significant changes in the capabilities of the robot model.

METHOD

The general approach was to develop a model capable of representing all states of the robot and use a cyclic genetic algorithm to train this model to walk forward. The robot used in the formation of our model was the Stiquito II; developed by Mills [6] as an alternative to the larger, more complex and expensive norm in six legged robots. It was selected due to its potential for expanded research and its elementary locomotion; which makes it a perfect platform for learning to walk using low-level primitives. The model was a data structure that could hold the essential information needed to determine the state of the legs and the subsequent movement calculated from the signal input. Fields for each leg were included to store the leg's capabilities and current position.

CYCLIC GENETIC ALGORITHMS

Some alterations to the general theme for Holland's genetic algorithms [7] were required to develop locomotion control. The time factor necessitated some way to represent behavior over time. The resultant algorithm is referred to as a "cyclic genetic algorithm." In the CGA the individual is represented by a series of nodes each of which is the individual's behavior control exhibited over a set segment of time. The nodes (or genes as we call them) can be of any size as long as the task they are to complete can be done in the segment provided. In addition, the gene can hold repetition information, telling how many time segments will be dedicated to that gene.

With a cyclic genetic algorithm there is some portion of its chromosome that is repeated a number of times. This separates it into two sections: a start section and an iterative section. The start section is executed once, whereas the iterative section is repeated until expiration of some parameter or a number of iterations. This type of GA is particularly useful for training robots where their start stance is unique in that it is only used while at rest (no motion). The desired effect is that the start section should set up the robot to move into a continuous cycle where sustained fluid motion can exist. A trailing stop section can be added to effect a smooth transition back to the at-rest stance.

To optimally take advantage of the cyclic genetic algorithm, the individual must be represented by a variable length chromosome (list of genes) or a fixed length chromosome with repetitions encoded in the gene. In the later case, care must be taken to provide enough genes to allow for the maximum number of signal changes. This will allow the algorithm to adjust the number of time units in the start and iterative sections to best suit the fluidity of motion.

For our experiments, we used both fixed and variable length representations. The fixed ones were faster and easier to work with and control; in addition, they had the required flexibility, due to the designed repetitions, to gain optimum results.

Genetic Operators

Probability for selection was based on fitness, which was computed by summing the fitness of the individual genes. The gene fitness equaled the forward motion produced by the gene's signal (repeated the indicated number of repetitions if applicable). This was done on the model by:

1) taking the current state of legs
2) applying the vertical movement
3) calculating the balance and probable legs on the ground from the model's current vertical position of each leg
4) applying the horizontal movement to alter the leg's state, but only counting legs on the ground in computation of the movement (fitness)
5) taking off some deduction for lack of balance and/or asymmetry of movement
6) repeat using next gene and the new legs' state.

This was sequentially done from the start to the end of the string and then repeated as many times as required in the iterative section. Using this fitness, the best individual was preserved; the rest of the new population was formed by stochastic selection of mates with the probability of selection proportional to the fitness.

Crossover in the start section was at a single point equivalent in both chromosomes. In the iterative section, since it could be considered a circle, crossover was performed at two points; again 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. In the case where these genes were represented as lists, 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 replace -- each gene had a random chance of being replaced by a new completely random gene. 2) Gene mutate -- each part of the gene had a random chance of having one of its bits altered.

For variable length chromosome genetic operators, selection and fitness were calculated in the same way as the fixed except for the addition of shortness and longness penalties, which were found to be necessary to maintain reasonable chromosome lengths in early stages. Crossover differed from the fixed in that the points picked did not correspond to the points picked in the mate. This resulted in varying lengths of chromosomes.

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 repetitions. Genes that had zero repetitions were moved out so that only active genes were at the start of the iterative section. Following these eliminations, if the number of genes or the total number of gene repetitions fell below some threshold, additional random genes were added until the thresholds were met.

Coordinators and Inhibitors

Each leg had a possibility of two coordinators. They coordinated the forward/back motion with the up/down. One made sure that if the leg was going back it was either already down or moving in that direction. The other ensured that if a leg was going up it was either already forward or moving in that direction. The coordinators for all legs were stored in a single 12 bit number (back-down and forward-up for each leg). The inhibitors affected pairs of legs. They prevented pairs of legs from moving back at the same time. The 2,3 inhibitor prevented both legs 2 and 3 from going back at the same time. It allowed 2 to move back, but inhibited 3. The inhibitors for the set of legs were stored in a single 15 bit number (one bit per possible pair). Coordinators and inhibitors could be applicable equally to all genes in an individual; in which case they were listed first up front. Or they could be unique to each gene; in which case they were stored in the gene.

TESTS

Tests were done on both variable and fixed length chromosomes using a variety of population sizes, gene representations, genes per individual, genetic operators, and number of generations. Fitnesses were determined after 100 moves. Each move equated to an activation applied for 100 msec.

Variable Length Chromosomes, Numeric Genes -- In these tests the chromosomes were the most primitive. The start and iterative sections could both vary in size. Each gene was a 12 bit number with an up or down bit and a forward or back bit for each leg.

Variable Length Chromosomes, List Genes -- The start section was fixed at 10 (this is the max throw of any of the leg movements). The iterative section could vary in length with penalties for being too long or too short. The gene was a list of 3 integers: the activation, inhibitors, and coordinators. A pre-evaluation algorithm combined the elements of the gene to form a single 12 bit number (activation) that corresponded to the specific leg movements.

Fixed Length Chromosomes, Global Inhibitors And Coordinators -- The start section was fixed at one gene and the iterative section at 12. Ten repetitions of one movement should be sufficient to get the robot's legs in the proper position to enter the cycle. Twelve in the iterative section was judged to be enough move changes to handle every possibility (two per leg). The chromosome was a list consisting of the individual's inhibitor, coordinator, start section gene and the 12 iterative genes. The genes were each made up of two integers: the moves integer with a limit of 10 (the max moves until full throw) and the activations integer (12 bit number). The pre-evaluation algorithm applied the coordinator and inhibitor to each activation and then reproduced the result for the specified number of repetitions designated in the moves integer.

Fixed Length Chromosomes, Gene-By-Gene Evaluator -- This was essentially the same as above except the Gene-by-Gene Evaluator was added. This helped to speed the elimination of low performance genes.

RESULTS

Comparison of the different methods at 2000 generations (Figure 1) reveals improvements with each change. The graph shows each method's average population fitness and the best individual's fitness. The y axis shows average speed of advance (in arbitrary units (9.5 is maximum)), the x axis shows the differing methods.

Variable Length Chromosomes, Numeric Genes (VLC, Numeric) -- This scheme never developed the optimal tripod gait within 40,000 generation, although it did consistently attain sustained forward motion. In addition, eventual development of a system that could learn in real time required faster learning rates. It was determined that inhibitors and coordinators were required.

Variable Length Chromosomes, List Genes (VLC, List) -- The results, with the addition of the inhibitors and coordinators resulted in significant improvement. But the drastic changes in chromosome length due to random crossover points in the iterative section were mixing up the gene locations too much to take advantage of the full power of genetic algorithms. A scheme allowing fixed chromosome length yet variable repetitions was required.

Fixed-Length Chromosomes, Global Inhibitors and Coordinators (FLC, Global) -- Fixing
the length and putting the repetitions in each gene, plus reducing the variances by making the inhibitors and coordinators global resulted in some additional improvement. A tripod like gait developed but its fitness plateaued with some less than optimal moves. These would probably eventually be worked out but a faster method was desired.

Fixed-Length Chromosomes, Gene-by-Gene Evaluator (FLC, G by G) -- The Gene-by Gene evaluator in conjunction with the genetic operators converged on the optimal tripod gait within 2000 generations.

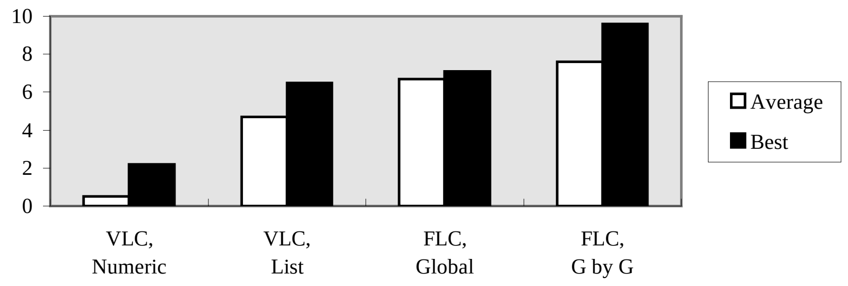

Figure 1: Method Comparison, 2000 Generations

What this comparison fails to emphasize is the rapid convergence to optimality of the last method (Fixed Length Chromosomes, Global Coordinators / Inhibitors and Gene-by-Gene Evaluator). This can be observed by the drastic improvements shown in Figure 2 (the x axis is the number of generations, the y axis is the acquired fitness). The addition of the gene-by-gene evaluator enabled the algorithm to reach optimality in less than 500 generations. This graph also emphasizes the difficulty of this problem in that the initial random population's performance was so poor. The negative figures come from movement causing backward motion and penalties for lack of balance and non-symmetric activation.

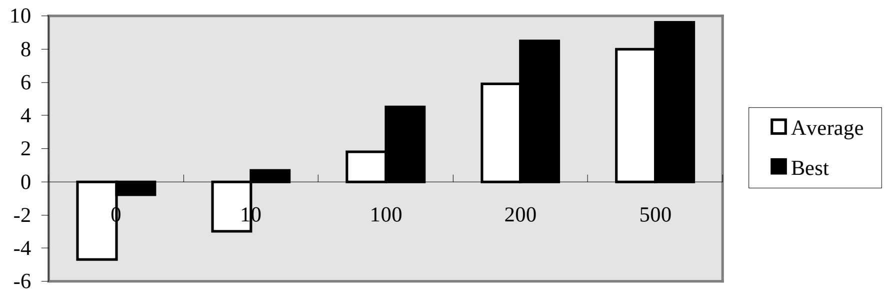

Figure 2: Fixed Length, Gene-by-Gene
fitness increase with increased generations

Important in this model is that it is adaptable to changes (sometimes drastic) in the abilities of the robot. In an attempt to test this adaptability, we disabled selected legs by restricting the horizontal movement (leaving the vertical enabled). Tests started with the healthy 500 generation population discussed in the previous paragraph and with new random populations. The results (Figure 3) confirmed the adaptability of the algorithm as in all cases the fitness improved quickly to a stable gait. The following key will assist in graph interpretation (all use the method FLC, G by G):

Start	Shows starting random population fitness and best individual after 500 generations fitness
Leg2S	Slight restricted movement of leg 2 (before & after 2000 generations)
Legs2&3T	Total disablement of legs 2 & 3 (before & after 5000 generations)
Leg2SN	Slight restricted movement of leg 2 (before & after 5000 generations) Starting with a random population
Legs2&3TN	Total disablement of legs 2 & 3 (before & after 30000 generations) Starting with a random population
All But One	All legs but one with disabilities (vertical and horizontal movement) (before & after 5000 generations)

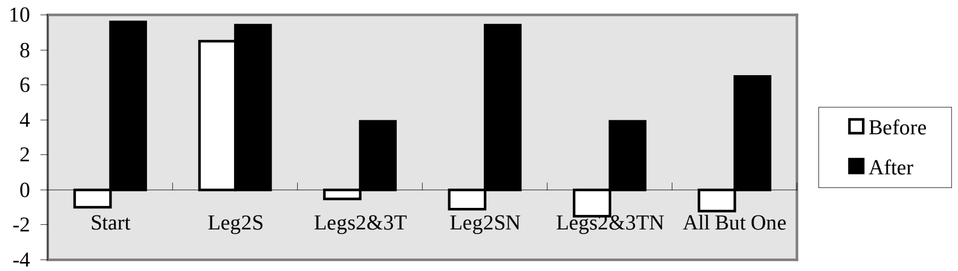

Figure 3: Fixed Length, Gene-by-Gene with Disabled Leg/Legs

CONCLUSIONS

The Cyclic Genetic Algorithm can, with only the most basic of motion primitives, produce reasonable gaits in a model of the Stiquito robot. With only slightly higher order primitives it is capable of developing the optimal tripod gait and attain maximum speed. Additionally, the learning algorithm is fast and will adapt to changes in the robot's capabilities.

ACKNOWLEDGMENTS

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

REFERENCES

1. Donner, M. D. (1986). Real-Time Control of Walking. Boston; Basel; Stuttgart: Birkhauser.
2. Brooks, R. A. (1989). "A Robot That Walks: Emergent Behaviors from a Carefully Evolved Network." Neural Computation (pp. 254-262).
3. Gallagher, J. C. and Beer, R. D. (1994). "Application of Evolved Locomotion Controllers to a Hexapod Robot." Technical Report CES-94-7, Department of Computer Engineering and Science, Case Western Reserve University. Refers to Beer, R. D., and Gallagher, J. C. (1992). "Evolving Dynamical Neural Networks for Adaptive Behavior." Adaptive Behavior, 1 (pp. 91-122). Cambridge: MIT Press.
4. 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.
5. Grefenstette, J. J. and Ramsey, C. L. (1992). "An Approach to Anytime Learning." Proceedings of the Ninth International Conference on Machine Learning, (pp. 189-195), D. Sleeman and P. Edwards (eds.), San Mateo, Ca: Morgan Kaufmann.
6. Mills, J. (1994). "Stiquito II and Tensipede: Two Easy-to-Build Nitinol-Propelled Robots." Technical Report #414, Computer Science Department, Indiana University.
7. Goldberg, David E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Reading, Ma: Addison-Wesley.