EVOLVING HEXAPOD GAITS
USING A CYCLIC GENETIC ALGORITHM

GARY B. PARKER, DAVID W. BRAUN, AND INGO CYLIAX
Department of Computer Science
Indiana University
Bloomington, IN 47405

Abstract

Gait generation is an integral part of a legged robot control. Hexapod gaits require the coordination of the simultaneous movement of all six legs in a cycle of activations. The problem is compounded when working with actual robots due to the variability in their capabilities. A learning algorithm that can optimize with the differences between and within robots would greatly reduce engineering calculations and increase robot adaptability. To be effective for the simplest of robot controllers, these algorithms should produce sequences of activations that can be used directly on the robot. Evolutionary computation would be ideal for this learning algorithm although few of its forms have a structure that conforms naturally to the cyclic nature of gaits. In previous work, we developed Cyclic Genetic Algorithms (CGAs), which are a variation on the basis genetic algorithm. Tests on robot simulations showed that these algorithms could rapidly converge to evolve the optimal tripod gait. In this paper, we successfully apply CGAs to an actual robot that is not only more complicated then the original simulation, but also has the increased variability innate in actual robots.

KEYWORDS: genetic, cyclic, robot, hexapod, control.

INTRODUCTION

Autonomous hexapod robots can be useful in situations where the environment is abnormally hostile: exploration of volcanoes, ocean depths, disaster rubble and extra-terrestrial bodies; military duties such as enemy search and mine field clearing. An important factor in attaining full autonomy is the robot's ability to adapt to changes in its own capabilities and the environment. This motivates designers to have the control system be learned as much as possible. In addition, learning reduces the human engineering required to develop the intricacies of the system.

Gait generation is an integral part of a legged robot control. Hexapod gaits require the coordination of the simultaneous movement of all six legs. In addition, sustained forward movement requires that the sequence of individual actions that propel the robot be continually repeated and that each state of the robot, throughout the cycle, can be easily transitioned to the next. The problem is compounded by the variances between robots plus the variances within a robot, such as differing leg capabilities. Evolutionary computation is particularly well suited for adapting a solution to the peculiarities of the problem. The difficulty comes in that most forms of evolutionary computation are not naturally equipped to handle the cyclic nature of gaits. One exception is with genetic programming, which can be used to evolve programs and programs can have loops. Graham Spencer [1] had some success in generating programs for hexapod gaits using genetic programming. His programs, tested only on robot simulations, resulted in gaits that maintained sustained forward movement but could not obtain the optimal tripod gait.

Another important facet of robot control development is testing on actual robots. Simulations alone cannot adequately predict the behavior of actual physical agents. Randall Beer and John Gallagher [2] used genetic algorithms to develop neural network controllers for a simulated hexapod robot. In later work, they tested these controllers on an actual robot [3]. Tripod gaits resulted in both the simulated and actual robots. Although promising for robots with complex controllers that have some *a priori* structure conducive to tripod gaits, this technique is inappropriate for small inexpensive robots that require a repeatable sequence of activations for control.

In previous work [4], we developed Cyclic Genetic Algorithms (CGAs) to solve these cyclic sequence problems by generating the proper sequence of primitive instructions that can be continually repeated to produce a gait. Tests showed that these quickly converging algorithms could produce optimal gaits on robot simulations where all robot leg capabilities were the same. In this paper we use the CGA to develop a usable gait for an actual autonomous hexapod robot. This was accomplished by creating a model with specific information taken from an individual robot. The CGA used this model to develop an optimal gait that was specific to the robot's capabilities. This gait was subsequently downloaded into the actual robot where its performance was confirmed to correspond to the performance of the model.

CYCLIC GENETIC ALGORITHMS

Genetic Algorithms, introduced by John Holland [5], are based on the laws of natural selection and survival of the fittest. The basic algorithm consists of three genetic operators (selection, crossover, and mutation) that are used to transform a randomly generated population into an optimal (or near optimal) one. The population is made up of individuals that are possible solutions to the problem and are usually represented as a bit string of fixed length called a chromosome. The survivability of the individual is dependent of how its solution compares (its fitness) with the other individuals in the population.

Cyclic Genetic Algorithms were developed to allow for the representation of a cycle of actions in the chromosome. They differ from the standard GA in that the chromosome is in the form of a circle with two tails. The tails of the CGA chromosome are provided to allow for pre and post-cycle procedures. They provide a means for completing tasks before and after entering the cycle. For gait sequence generation, the pre-cycle can position the legs in a ready to walk posture and the post-cycle can return the robot to a stable at rest posture. In our application, we used only the pre-cycle tail. The CGA genes can be one of several possibilities. They can be as simple as normal genes that represent traits of the individual or they can be as complicated as cyclic sub-chromosomes that can be trained separately by a CGA. For our purposes, the genes represent tasks that are to be completed in a set amount of time. The trained chromosome will contain the cycle of primitive instructions that will be continually repeated by our robot's simple controller to produce a gait.

CGAs can have both fixed and variable length chromosomes. In either case, the system must be able to allot the proper number of tasks to each phase and be flexible enough to allow the CGA to form a complete cycle. When fixed length are used, the tasks at each gene can be repeated. The number of repetitions is encoded in the gene. In this way, fixed length chromosomes can take on the desirable characteristics of variable yet maintain the increased control of training fixed.

SERVOBOT

The robot used was the ServoBot, which we developed for legged robot experimentation. It is an inexpensive hexapod robot that has two degrees of freedom per leg. Twelve servos, two per leg, provide thrust and vertical movement. 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 line can be disconnected. The FPGA can store and execute (repeated the designated section) the sequence of primitive instructions downloaded. Each instruction corresponds directly to an activation that is activated for 100ms, then the next instruction in the sequence is activated.

An input to the ServoBot is a 12 bit number where each bit represents a servo. A signal of 1 moves the leg back if it is a horizontal servo and up if it is a vertical servo. A signal of 0 moves it in the opposite direction. Figure 1 shows an example of an activation and its result on the robot. The activation can be thought of as 6 pairs of actuations. Each pair is for a single leg with the first bit of the pair being that leg's vertical activation and the second being that leg's horizontal activation. The legs are numbered 0 to 5 with 0,2,4 being on the right from front to back and 1,3,5 being the left legs from front to back. The activation 100101101001 results, as shown, in one phase of the classic tripod gait, which is considered to be the optimal gait for speed in this simple rigid robot when all its actuators are fully functioning.

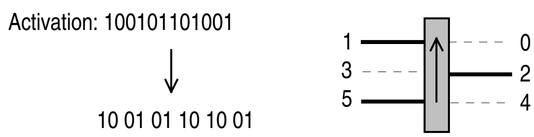

Figure 1: Results of a Robot Activation

In order to allow the CGA to generate a gait, the ServoBot had to be simulated by a model. The model was a simple data structure that held each leg's capabilities and current state (Figure 2). Input activations were used with

Fields specific for each leg:
current up -- current vertical position of the leg.
max up -- position off the ground when completely up.
max down -- position off the ground when completely down.
current back -- current horizontal position of the leg.
max back -- posit relative to completely forward when completely back.
Fields applicable to all legs:
rate up/down -- rate of vertical movement when servo activated.
rate back/forward -- rate of horizontal movement when servo activated.

Figure 2: Model Data Structure

these capabilities to determine how much to change the current state. Measurements to fill the position fields were taken by activating each control while recording the leg's maximum throw. An average rate per activation was calculated for horizontal and vertical movement by dividing the maximum throw by the minimum number of activations required to attain it.

TRAINING

Development of gaits was accomplished by running a CGA for 2000 generations on five distinct starting populations of 64 individuals. All sections of the chromosome were initialized with a random number within the appropriate range. Intermediate populations at 100, 200, 500, and 1000 generations were stored for subsequent observation of gait development.

The CGA chromosome structure used for this problem was the fixed length chromosome with a pre-cycle tail of length one and cyclic section of 12 genes. Each gene was a list that had two parts: an *activation* and a *repetition*. The *repetition* designated how many times to repeat the specified *activation*. Also included in each chromosome were two global activation affecters: the coordinators and the inhibitors. They were initiated as random numbers and were evolved during training.

Coordinators were 12 bit numbers that directed the coordination of individual leg movement. These numbers could be looked at as six pairs of bits, one pair for each leg. The first being the back-down coordinator which, if activated, ensured that the leg would be down or moving in that direction if it was moving back. The second bit was the forward-up coordinator, which ensured that the leg would be moving forward if it was up.

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). This 15 bit number could be thought of as 5 groups. The first group made up of 5 bits indicated which legs would be inhibited from moving in the same direction as leg 0. Five bits were required to cover the remaining legs 1 through 5. The second group was made up of 4 bits showing what legs would be inhibited from moving in the same direction as leg 1. Since leg 0 had already been matched with all legs in the first group it does not appear in the second. This continues until all possible leg matchings have been addressed.

The fixed length chromosomes used for training had to be converted to an activation per gene form before they could be used by the robot or its model. First the inhibitors then the coordinators were applied to the *activation* in each gene. Then the *activation*s were duplicated *repetition*s number of times. A -1 was added to make the separation between the pre-tail and cyclic sections. The result was a new CGA chromosome that was a sequence of activations.

Fitness was computed in each chromosome by summing the fitness of individual genes as each activation was applied to the current state of the simulation (Figure 3). This was done once for each activation in the pre-cycle tail of the chromosome and repeated in the cyclic section until a total of 100 activations was reached. This fitness was computed for each individual in the population and used to stochastically choose the individuals that would parent the next generation.

For each gene do
1) apply the activation’s vertical movement to the current state of legs
2) calculate the balance and probable legs on the ground from the model’s
current vertical position of each leg
3) apply the activation’s horizontal movement to the current state of legs
4) compute movement (fitness) from the horizontal movement of the legs
on the ground
5) reduce the forward movement (fitness) if the positioning of the legs on
the ground indicate that the robot would be out of balance
6) reduce also if the forward movement is asymmetrical (one side
producing more trust than the other)

Figure 3: Fitness Computation at Each Gene

Crossover was accomplished by randomly picking corresponding spots in the two selected parents. In the pre-cycle tail, a single point in both chromosomes was picked. In the cyclic section, since it could be considered a circle, crossover was performed at two points. The effect was to swap sections within the circle. An alternate type of crossover was a gene-by-gene crossover that performs 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. There were two types of mutation used and selected randomly after each recombination. One in which each gene had a random chance of being replaced by a new completely random gene. The other was one were each part of the gene had a random chance of having one of its bits flipped.

Gene-by-Gene Evaluation, a genetic operator peculiar to CGAs was used to clean up the chromosome by randomly picking one or two individuals from the population on each set of trails and examining each gene one at a time. Genes were evaluated move-by-move by comparing the previous move fitness to the present. Genes that performed poorly in their current position were eliminated. Genes that were good in the execution of their early repetitions and subsequently dropped 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 cyclic section.

RESULTS

Performance tests on simulations were done to determine the fitness of the resulting populations from each of the five starting populations. The simulated forward movement produced by the populations best individual (activation sequence producing a gait) for each of the intermediate generations was computed using the robot model. These same gaits were also used to test their usability on the actual robot. The distance traveled by each robot during physical tests was recorded.

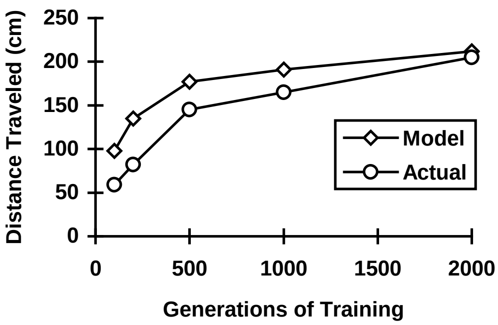

Figure 4: Average Performance of Five Populations.

Figure 4 shows the models' computations and the actual robots' measured distances traveled for each generation averaged over the five starting populations. In both cases the horizontal axis shows the number of generations completed before sampling the gait's performance and the vertical axis represents the centimeters moved in 300 activations. The CGA produced gaits with continuously increasing performance for both the model and the actual robot. All five of the resultant 2000 generation gaits were tripod in nature. Two were judged to be fully optimal and superior to gaits developed by the robotics hardware technician.

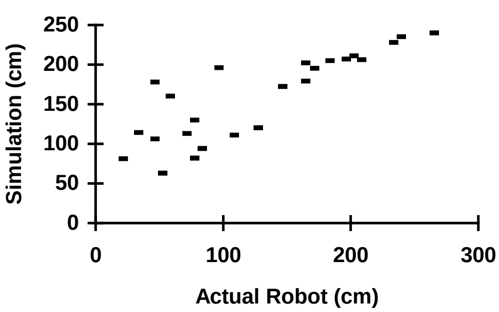

Figure 5: Distance Traveled; Simulation vs Actual Robot

A scatter plot of all the simulation computations verses the actual robot measurements is shown in Figure 5. As can be seen, the model was a satisfactory means of predicting the trend of the actual robot's performance as the training progressed (correlation = 0.86).

CONCLUSIONS

From these results we conclude that CGAs can be successfully used to generate gaits for actual hexapod robots. The gait generator was capable of producing useful gaits in simulation that were transferable to the actual robot and comparable to those developed by the robotics hardware technician. The algorithm was quickly converging despite a lack of *a priori* knowledge and the cyclic chromosome structure with tasks for genes worked naturally to produced the primitives required for direct manipulation of the robot actuators. Leg coordination, inhibition, and the proper sequence and number of activations for the specific robot were all learned with progressively better gaits being constantly evolved. The CGA should be applicable to any robot requiring a repetition of a sequence of primitives for control; no other gait control is required.

Future research will include developing and testing gaits for ServoBots with reduced capabilities including severe leg deprivation. Having adjustable servo speeds we can also use the ServoBot to test CGA generation of other gaits such as a slower, more stable metachronal wave.

Acknowledgments

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

References

[1] G. Spencer, Automatic generation of programs for crawling and walking, *Advances in Genetic Programming*, pp. 335-353, K. Kinnear, Jr. (ed.), (Cambridge, Ma: MIT Press, 1994).

[2] R. D. Beer and J. C. Gallagher, Evolving dynamical neural networks for adaptive behavior, *Adaptive Behavior*, 1, 1992, 91-122.

[3] J. C. Gallagher and R. D. Beer, Application of evolved locomotion controllers to a hexapod robot, Technical Report CES-94-7, Department of Computer Engineering and Science, Case Western Reserve University, 1994.

[4] G. Parker and G. Rawlins, 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.

[5] J. H. Holland, *Adaptation in Natural and Artificial Systems*, (Ann Arbor, Mi: The University of Michigan Press, 1975).