Using Cyclic Genetic Algorithms to Learn Gaits for an Actual Quadruped Robot

Gary B. Parker and William T. Tarimo
Department of Computer Science
Connecticut College
New London, CT, USA
parker@conncoll.edu, wtarimo@conncoll.edu

***Abstract*—It is a difficult task to generate optimal walking gaits for mobile legged robots. Generating and coordinating an optimal gait involves continually repeating a series of actions in order to create a sustained movement. In this work, we present the use of a Cyclic Genetic Algorithm (CGA) to learn near optimal gaits for an actual quadruped servo-robot with three degrees of movement per leg. This robot was used to create a simulation model of the movement and states of the robot which included the robot's unique features and capabilities. The CGA used this model to learn gaits that were optimized for this particular robot. Tests done in simulation show the success of the CGA in evolving gait control programs and tests on robot show that these control programs produce reasonable gaits.**

Keywords -Genetic Algorithm; Cyclic Control; Quadruped; Gait; Evolutionary Robotics; Learning Control; Cyclic Genetic Algorithm

I. INTRODUCTION

The use of mobile legged robots necessitates the development of optimal walking gaits, which is typically not an easy task. Multiple legs and degrees of movement per leg offer the robot increased walking capabilities, but it is a challenge to learn and coordinate many such separate movements into a stable gait. The quadruped servo-robot with three degrees of movement per leg poses a stability challenge compared to hexapod and octopod robots. For the quadruped robot to achieve and sustain fast forward movement, it is necessary for it to maintain dynamic stability, whereas hexapod robots can reach a maximum speed while maintaining static stability. For a reasonable gait, each leg needs to repeat a series of movements in synchronization with the other legs. In this paper, we use a Cyclic Genetic Algorithm (CGA) operating on a movement model (simulation) of an actual quadruped robot to generate near optimal gaits. The gaits are then tested on the actual robot to verify their viability.

Learning using evolutionary computation involves evolving many separate traits of a solution to a problem; this is common to learning gaits where many separate movements per leg are to be learning in synchronization. In a recent review article, Daoxiong Gong, Jie Yan, and Guoyu Zuo presented, in much detail, the suitability of evolutionary computation techniques in gait optimization for mobile legged robots [1]. Evolutionary computation techniques and in particular, Genetic Algorithms (GA), have previously been used to develop gaits for legged robots. Graham Spencer used genetic programs in his work to learn gaits for a virtual robot using only minimal knowledge of the mechanisms of walking [2]. Sonia Chernova and Manuela Veloso used evolutionary computation to learn gaits for four-legged robots emphasizing gait optimization [3]. Susanne Still et al. presented how a chip can be used to control the leg movements of a four-legged robot by driving motors with time varying voltages from a small network of coupled oscillators [4].

CGAs were developed to generate walking gaits for a hexapod servo-robot [5]. The CGA is distinct from the other methods of gait learning in that it learns the machine/assembly code that can be directly loaded in to the controller. Compared to the current quadruped robot, each of the six legs of the hexapod robot could move vertically and horizontally and the increased number of legs made it possible to produce statically stable gaits.

In previous work [6] we tested the effectiveness of a similar CGA in evolving walking gaits for a robot with more complex legs. The servo-robot used has four legs, each of which has three degrees of freedom, in/out, up/down, and forward/back. Compared to hexapods, quadrupeds have more stability challenges since it is easier for quadrupeds to fall out of balance while moving. The CGA learned gaits that kept the robot stable while producing forward movement, but the final results were not as consistent as desired (two different general gait types developed with only one being near-optimal), and only initial tests were done on the robot.

For the research reported in this paper, we adjusted the fitness module of the model to emphasize speed over stability. In the previous work, static stability was rewarded significantly more than dynamic stability. In this work, it is still rewarded more, but not to the same degree. The result is that the gaits produced are more near-optimal. In addition, in this paper we report a full set of tests on the robot to confirm the viability of the system.

II. QUADRUPED SERVO-ROBOT

The robot used is a quadruped servo-robot (Figure 1) by Michael Cantor, and it was constructed in the Connecticut College robotics laboratory. The robot body and legs are made of masonite. It has standard hobby servo-motors for actuators and uses a Basic Stamp II for control. The robot has three degrees of movement per leg: in/out, up/down and forward/back movements. These movements are controlled by timely signals from a control program running in the Basic Stamp. The program in the Basic Stamp is made up of movement signals that are sent to all the servo-motors. These movement signals (or commands) are sent in parallel and tell each of the servo-motors on the legs to move in one direction or the other. There is no command to stop movement so this only happens when the leg goes full throw in its possible movement, or discontinues signals on that direction, or the servo-motor reaches the end of its movement. The movement signals or activations are sent to the legs every 20 milliseconds. A sequence of these activations is needed to have continuous movement of the legs and a proper sequence of activations is required for the control program to produce a proper gait. The coordination of the concurrent movement of all of the actuators along with the requirement for the previous steps to end in the proper position to start the next step is what makes this problem so challenging. Moreover, there is the problem of dynamic stability where the robot has to be dynamically stable (creating constant forward movement) at all times.

TABLE I: INFORMATION STORED IN THE SIMULATION MODEL OF THE ROBOT.

Name	Description
current-up	The current vertical height of the leg
max-up & max-down	The highest and lowest positions that the leg could reach
rate-up & rate-down	The rates of up/down movements per control pulse
on-ground	Indicates which legs are on ground, using a 0 or 1
current-back	The current horizontal position of the leg from the back most position.
max-back	The back most position a leg can go.
current-in	The current in/out distance of the foot from the side of the robot body.
max-in & min-in	The in most and out most distances the foot could reach.
rate-in & rate-out	The rates of in/out movements per control pulse
current-theta	The angle relative to the line perpendicular to the heading of the robot
max-back-theta & max-fwd-theta	The back most and forward most angles of the leg
rate-back-theta & rate-fwd-theta	The rates at which the angle changes per control pulse when the leg moves horizontally.
temp1, temp2 & temp2	Temporary variables if a need arises.

To successfully learn optimal gaits for the particular quadruped robot, the developed cyclic genetic algorithm was used to train a simulation model of the robot to walk. Throughout the training the algorithm emphasizes forward distanced covered and stability of the robot. With this in mind, the approach was to develop a movement simulation model of an actual robot with data from its physical features and movement capabilities. This model, with information for each leg, would present all the movement states of the robot at any instance in time.

The movement model of the quadruped servo-robot was created by taking accurate physical measurements of the legs and combined movement capabilities and rates for all the degrees of movement per leg. This data was stored in a lookup table that the CGA could read and write to throughout the training. The information stored is as shown in Table I. Distances are measured in millimeters and angles in radians.

The legs of the quadruped can move in/out while moving back/forward and therefore the resulting horizontal movements are not linear but radial. It was necessary to find an appropriate way to determine the horizontal position of a leg as a function of both the current angle of the leg relative to the line perpendicular to the heading of the robot and the in/out distance from the base of the leg to the foot. To calculate the relative horizontal position of the leg, the current in/out distance of the leg is multiplied by the sine of the angle.

Figure 1: Photograph of the quadruped servo-robot used.

III. CYCLIC GENETIC ALGORITHM

A Cyclic Genetic Algorithm (CGA) is a type of Genetic Algorithm (GA) [7] that was designed to evolve solutions to problems with a cyclic nature or pattern [8]. The CGA is a suitable method for learning cyclic control programs like gait control for legged robots. The main difference between a CGA and the standard genetic algorithm is in the structural makeup of the chromosomes. The genes of a standard GA chromosome represent traits of a learned solution where as in a CGA the genes represent tasks to be completed in a set amount of time (Figure 2). This, in effect, makes the CGA a method for evolving gait control programs since learning gaits involves

the timely coordination of may separate movements to achieve a sustained movement. The tasks in this case are activations that need to be sent to the servo-motors every 20 milliseconds to maintain continuous control. The chromosome of the original CGA has genes that represent three sets of tasks depending on location in the chromosome. Each of these sections can be made up of any number of tasks, each of which can be repeated multiple times in order to achieve the desired solution.

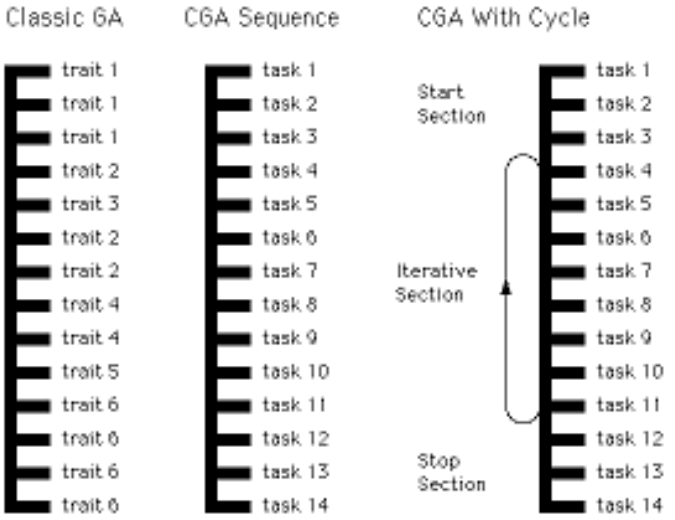

Figure 2: A comparison of chromosome structure between the classic GA and CGA.

In learning gait control signals for a legged robot, the start section should set up the robot to move into a continuous cycle (the cyclic section) where sustained fluid motion can take place. The tail section can optionally be added at the end of the cyclic section to provide a smooth translation back to the at-rest stance of the robot. CGAs use the same genetic operations, (selection, crossover and mutation), as in the standard GA's except that two point crossover is used in the cyclic section since swapping a section of the cycle is what is being accomplished.

IV. Gait Production for the Quadruped Robot

A. Chromosome Structure

The CGA chromosome structure for the problem is shown in Figure 3 with an example shown in Figure 4. The fixed length chromosome was made up of 4 parts: coordinator, inhibitor, start section, and cyclic section.

$$\mathbf{[C\ I\ \{(R\ A)\}\ \{(R\ A)_1\ (R\ A)_2\ (R\ A)_3\ \ldots\ (R\ A)_{11}\ (R\ A)_{12}\}]}$$

start section — *cyclic section*

Figure 3: A CGA chromosome structure with coordinator (C), inhibitor (I), and the start and cyclic sections which are made up of genes with repetitions (R) and activations (A) in each.

```
[8946 2432 {(32 2353)} {(35 3452) (9
2543) (32 8323) (36 564) (7 5032) (22
1321) (12 88) (7 4123) (3 1255) (0
1231) (0 6524) (0 17)}]
```

Figure 4: An example CGA chromosome structure written in the Scheme programming language with a coordinator, inhibitor, and start and cyclic sections.

Coordinators and Inhibitors are part of robot's coordination mechanism, which could evolve to increase gait control and proper movement. These two numbers were initialized as random numbers and were learned by the CGA. The coordinators and inhibitors were applied to the activations of the start and cyclic sections before the control program was run in the robot (simulation or actual).

C: Coordinators coordinated the three movements of each leg. For example, one coordinator, if activated, made sure that if that leg was going back it was either already down or moving in that direction. Another coordinator ensured that if a leg was going up it was either already forward or moving in that direction. The coordinator for all legs is a 12-bit binary.

I: Inhibitors affected pairs or triples of legs. They prevented certain legs from moving in the same direction at the same time. For example, the inhibitor for legs 2&3 prevented both legs 2 and 3 from going back or forward at the same time. It allowed 2 to move back, but inhibited 3. The inhibitors for the set of legs are represented as a 15-bit binary number. Coordinators and inhibitors are applied equally to all genes and they are listed at the start and cyclic sections of the chromosome.

The start section has one gene and the cyclic section has 12 genes. This setup was found to be sufficient to provide the maximum number of required chromosome changes during the evolution process. Each gene has two parts: repetitions and activations.

R: Repetitions represent the number of times to repeat the activations which are concurrent signals sent to the servos.

A: Activations represent the signal sequence sent to the 12 servo-motors. Each activation signal is a 12-bit binary number, with 3 bits dedicated to each leg, one bit for each of the in/out, up/down and forward/back movements. Since each bit can either be a 0 or a 1, one bit is sufficient to represent the two states of any movement. A signal of 0 on any of the movements is interpreted as a command to send the legs down, in, or forward. A signal of 1 would command any of the legs to go up, out, or back. Table II below shows the interpretation of all the 12 bits.

TABLE II: INTERPRETATION OF THE 12-BIT CONTROL SIGNAL IN THE ACTIVATIONS PART OF THE CHROMOSOME. LEG 0 IS THE RIGHT FRONT, 1 IS THE LEFT FRONT, 2 IS THE RIGHT REAR, AND 3 IS THE LEFT REAR.

Bit Index	Affected leg	Command when 0	Command when 1
0	3	Down	Up
1	3	In	Out
2	2	In	Out
3	2	Down	up
4	1	Down	Up
5	0	Down	Up
6	2	Forward	Back
7	1	Forward	Back
8	0	Forward	Back
9	3	Forward	Back
10	1	In	Out
11	0	In	Out

B. Gait Learning and Genetic Operations

The CGA used a population of 64 individuals. The initial population was created by randomly assigning numbers (according to the allowed input range at each part of the chromosome) to all the parts of the chromosomes. Five separate tests were conducted with the CGA learning on the 5 different random starting populations. During the genetic evolution process the CGA evaluated each chromosome of an individual using the movement simulation model of the robot and assigned each a fitness score.

Fitness was determined by running the simulated robot for 500 activations (control signals) of the gait/control program for each individual. This assured each individual was tested in a long enough time to ensure at least more than one whole step. The motivation for this choice was to ensure that the algorithm also learns the best and smoothest way to join different strides and steps without losing stability or forward thrust. Fitness was computed by summing the fitness from each of the 500 individual activations making up the test gait. The activation fitness was computed in terms of the forward distance and balance produced by the gene's activation signal, which is repeated the indicated number of repetitions. This was done on the model by:

- taking the current state of legs;
- applying the vertical movement;
- determining the probable legs on the ground from the model's current vertical position of each leg;
- calculating the balance of the robot from the nature and number of legs on the ground;
- applying the horizontal movement to alter the leg's state, but only counting legs on the ground in computation of the forward distance travelled;
- deducting fitness for lack of balance and/or asymmetry of movement;
- and repeating the process using the next activation and the new state.

This was sequentially done from the start to the end of the chromosome, each time repeating as many times as required in the cyclic section. Earlier observations and studies on the nature of the robot's movement and the nature of servos under pressure, as well as the presence of friction between the legs and the floor, revealed that back/forward and in/out movements are hugely affected by these resistive forces. Adjustments were required in determining the back/forward and in/out movements when direction changes were first applied. Each in/out and forward/back movement in a new direction was set to produce ⅛ of the movement in the first pulse, ¼ in the second pulse, ½ in the third pulse, and full movement from the forth pulse and on. These adjustments were made in the simulation to make it a better model of the actual robot.

At each generation, new chromosomes were created by performing a roulette wheel selection to select two parent chromosomes, with the probability for selection based on fitness. These parents produced children using crossover and mutation. Crossover was performed at randomly selected single point in the start section where as in the cyclic section two points were used. Each new offspring went through a possibility of mutation where each bit was given a 1 out of 150 chance of flipping from a 0 to a 1 or vice versa. After 64 children chromosomes were created in this way, the generation was complete. The process was repeated for a total evolution run of 5,000 generations, this number ensured that the populations can evolve to the highest possible fitness. The program stored copies of the population every 10 generations from generation 0 up to generation 100, every 20 generations up to generation 300, every 50 generations up to generation 800, and every 100 generations up to generation 5000.

V. RESULTS

From the results stored from all the five CGA runs, we calculated the average fitness per population for all the populations collected. We then plotted these results in graph of average fitness per population versus the appropriate generation number. Figure 5 shows this graph for populations collected throughout the 5000 generations. Figure 6 shows the same graph focused on the pattern observed from the first 1500 generations.

As can be seen on these two graphs, the learning pattern of the CGA can be observed. The evolution shows a rapid increase in average fitness in the first generations which then becomes steadier after about 500 generations. The CGA was successful at evolving more fit generations in successive generations, and continued to slowly improve the control program until somewhere in the 1500 to 2000 generation range. Figures 7 and 8 show fitness of the most fit individuals from populations collected throughout the 5000 generations. As expected, the CGA evolving individual gaits that get progressively more fit as the evolution goes on.

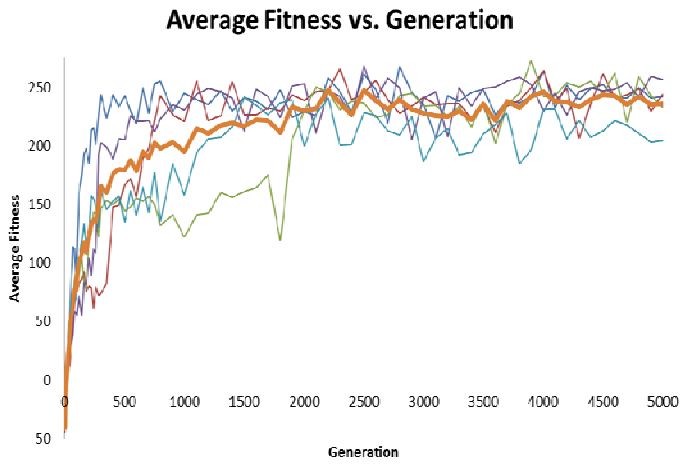

Figure 5: A plot of average fitness per population from populations selected from generation throughout 5000 generations. The average of the five trials is shown in bold.

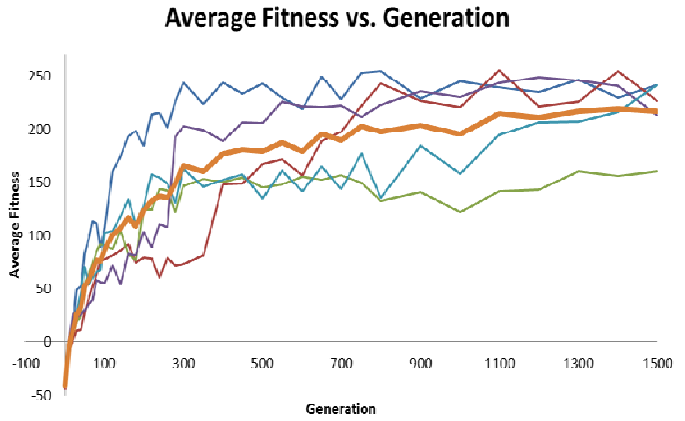

Figure 6: A plot of average fitness against generation number focused on the first 1500 generations. The average of the five trials is shown in bold.

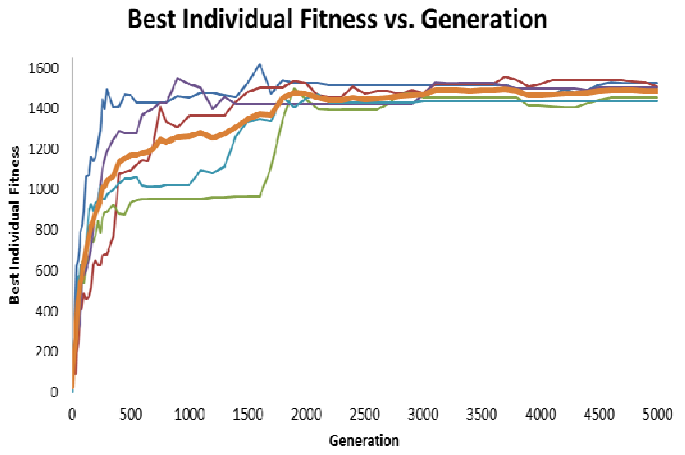

Figure 7: A plot of the best individual's fitness in a population against generation number from populations throughout the 5000 generations. The average of the five trials is shown in bold.

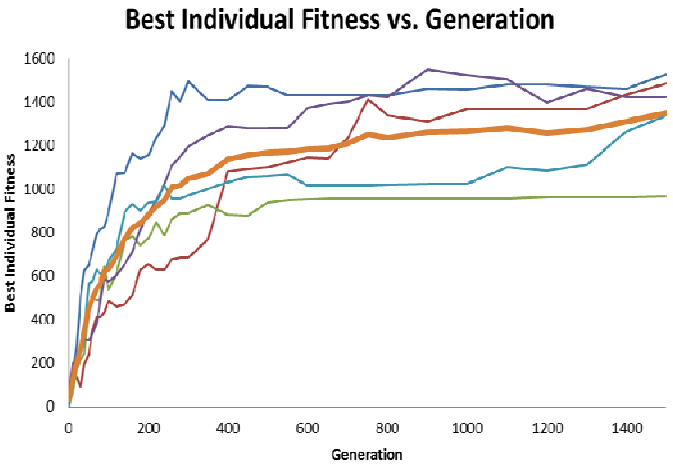

Figure 8: A plot of the best individual's fitness in a population against generation number focused on the first 1500 generations. The average of the five trials is shown in bold.

The second crucial set of results was from the tests made on the actual robot. Since all the five simulation tests maintained a similar evolution pattern and display reasonable gaits, we only saw the need to use results from one the five runs to perform tests of the robot. From this run, the best gaits from populations 0, 10, 100, 200, 500, 1000, 2000, and 5000 were processed for further tests on the robot. Each of these gaits was downloaded to the Basic Stamp and allowed to control the movement of the robot for a total of 500 activations as in the simulation. Each gait was run 3 times while collecting the forward distance (in millimeters) travelled and the final score was the average of the three distances. Figure 9 below shows a plot of the distances travelled by the gaits on the actual robot compared to the distances produced in simulation by the same individuals.

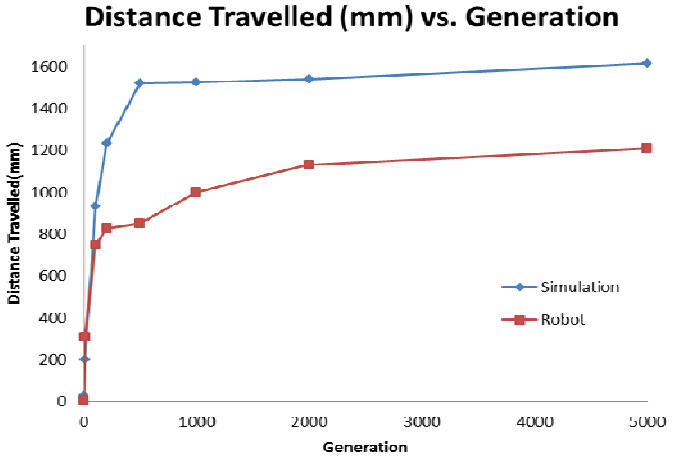

Figure 9: A comparison of distances (mm) travelled by the same individuals between the actual robot and the simulation.

As can be seen on Figure 9 above, the same gaits travelled shorter distances on the robot as compared to the simulation. What is more important as an observation from these results is the similar evolution pattern of the two sets of results. It is clear that the results from tests on robot are consistent with the

simulation results --there is almost a constant difference in distances separating the two plots, which implies that the difference is due to the experimental nature and minor errors in the robot capability tests. There are physical factors and errors in the environment that could not all be easily incorporated in the simulation, especiually friction and resistance from the floor.

All runs resulted in reasonable near optimal gaits, the same movement patterns were observed on simulation and in actual robot tests. This shows that the CGA was successive in learning near optimal gaits that could be successfully transferred to the robot. Most of the gaits evolved were functionally equivalent to the best quadruped gaits that had been manually programmed into the robot. The CGA produced gaits can be described as diagonal-paired gaits where a pair of diagonal legs moves in directions opposite to the other pair. This gait is shown in the Figure 10 below.

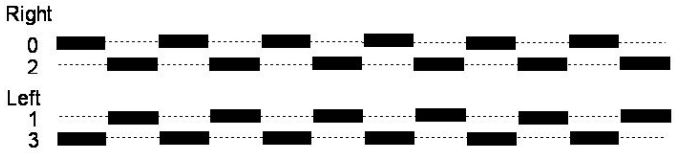

Figure 10: Movement of the robot based on the overall gait pattern learned by the CGA. The black rectangles indicate legs not on the ground. Leg 0 is the right front, 1 is the left front, 2 is the right rear, and 3 is the left rear. Reading the figure from left to right, legs 1&2 start on the ground as legs 0&3 reposition.

The black areas show legs that are off the ground. The dashed line areas are where legs are on the ground and producing thrust. With respect to up/down, forward/back and in/out; legs 1&2 start out moving back and down (providing forward thrust); legs 0&3 are moving forward and up. Legs 2 and 3 move in while moving forward and out while moving back and Legs 0 and 1 do the opposite. Half way through the full movement of the legs moving forward, they start to move down as they keep moving forward to reposition for the next step.

This evolved gait made use of alternate pairs of legs moving in a two-step cycle that involved reaching forwards with two lifting legs while the two lowering legs moved back. This gait is not statically stable. If the motion stopped the robot would tip one way or the other. However, it is dynamically stable and the force of the servos and resultant leg movement is sufficient to produce enough forward momentum that the robot has minimal rocking motion as it walks.

VI. CONCLUSIONS

The CGA was successfully used to learn near optimal gaits for the 4-legged robot with 3 degrees of freedom per leg. Learning was done on a model of an actual robot and tested on the robot. In all five trials the CGA learned an effective gait and did so within 2000 generations. Tests on the actual robot confirmed the effectiveness of the learned gait and showed that the model adequately represented the robot during gait evolution. These results are a further indication that the CGA is capable of directly learning the control programs for multi-legged robot locomotion. This algorithm can be a useful tool for learning gaits in any real world scenario; all that is needed is an accurate movement model of the robot that effectively represents its unique features and capabilities.

In future work, we will conduct more intensive studies with CGA learning applied to the actual robot locomotion, considering different terrain environments and how the CGA can adapt to the changes in the capabilities of the robot. In addition, future work will include the application of the CGA to learn gaits for six and eight legged robots with three degrees of freedom legs.

REFERENCES

[1] Daoxiong Gong, Jie Yan, and Guoyu Zuo (April 2010), "A Review of Gait Optimization Based on Evolutionary Computation," School of Electronic Information and Control Engineering, Beijing University of Technology, Beijing 100124, China.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Spencer, G. (1994), "Automatic Generation of Programs for Crawling and Walking," *Advances in Genetic Programmin*g. (pp. 335-353) K. Kinnear, Jr. (ed.), Cambridge, MA: MIT.

[3] Sonia Chernova, Manuela Veloso (September 2004). "An Evolutionary Approach To Gait Learning For Four-Legged Robots,"in Proceedings of IROS' 04, Sendai, Japan, September 2004.

[4] Susanne Still, Bernhard Schlkopt, Klaus Hepp, and Rodney J. Douglas, (2000), "Four-legged Walking Gait Control Using a Neuromorphic Chip Interfaced to a Support Vector Learning Algorithm," *NIPS2000.*

[5] Parker, G., Braun, D., and Cyliax, I.(1997), "Evolving Hexapod Gaits Using a Cyclic Genetic Algorithm," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC'97*). (pp. 141-144).

[6] Gary B. Parker, William T. Tarimo, and Michael Cantor., "Quadruped Gait Learning Using Cyclic Genetic Algorithm," *in Proceedings of 2011 IEEE Congress on Evolutionary Computation (CEC 2011).*

[7] Holland, J., "Adaptation in Natural and Artificial Systems," Ann Arbor, MI: The University of Michigan Press, 1975

[8] Parker, G., and Rawlings, G. (1996), "Cyclic Genetic Algorithms for the Locomotion of Hexapod Robots," in *Proceedings of the World Automation Congress (WAC'96), Volume 3, Robotics and Manufacturing System*s. (pp. 617-622.