6-2011

Quadruped Gait Learning Using Cyclic Genetic Algorithms

Gary Parker
Connecticut College, parker@conncoll.edu

William T. Tarimo

Michael Cantor

Recommended Citation
Parker, G.B.; Tarimo, W.T.; Cantor, M., "Quadruped gait learning using cyclic genetic algorithms," Evolutionary Computation (CEC), 2011 IEEE Congress on , vol., no., pp.1529,1534, 5-8 June 2011 doi: 10.1109/CEC.2011.5949797

Quadruped Gait Learning Using Cyclic Genetic Algorithms

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

DOI:10.1109/CEC.2011.5949797

Quadruped Gait Learning Using Cyclic Genetic Algorithms

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

***Abstract*— Generating walking gaits for legged robots is a challenging task. Gait generation with proper leg coordination involves a series of actions that are continually repeated to create sustained movement. In this paper we present the use of a Cyclic Genetic Algorithm (CGA) to learn gaits for a quadruped servo-robot with three degrees of movement per leg. An actual robot was used to generate a simulation model of the movement and states of the robot. The CGA used the robot's unique features and capabilities to develop gaits specific for that particular robot. Tests done in simulation show the success of the CGA in evolving a reasonable control program and preliminary tests on the robot show that the resultant control program produces a suitable gait.**

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

I. INTRODUCTION

The development of walking gaits is an important part in the designing and studies of legged robots. While multiple legs and multiple degrees of freedom can give the robot increased capabilities, the task of coordinating many separate movements into a stable gait can make learning gaits difficult. The quadruped servo-robot with three degrees of movement per leg poses a stability challenge. In order for a quadruped robot to achieve and maintain forward movement, it is necessary for each leg to repeat a series of movements in synchronization with the other legs. In this paper we used a Cyclic Genetic Algorithm (CGA) operating on a model of an actual robot to generate gaits.

Evolutionary computation techniques and in particular, Genetic Algorithms, have previously been used to develop gaits for legged (primarily hexapod) robots with two degrees of movement per leg. 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]. 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].

In a previous work Parker made use of cyclic genetic algorithms to develop walking gaits for a hexapod robot [5]. Each of the six legs of this hexapod robot could only move vertically and horizontally and the number of legs made it possible to produce statically stable gaits. One of the main differences of the CGA in comparison to other evolutionary computation methods is that the CGA learns the actual sequence of activations needed to be continually repeated to produce a gait. It is, in a sense, an automated code generator of machine level or assembly language code that can be directly loaded into the control chip.

In this paper the ability of a CGA to evolve effective walking gaits for a robot with more complex legs is tested. 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. In addition, although control for the same number of servos was being learned as in the hexapod, the three degrees of freedom legs posed additional complexity since the in/out movement required the simulation to deal with radial, in addition to linear movement, plus there were more alternatives in attaining the same general movement. Nevertheless, the CGA produced gaits that kept the robot stable with sustained forward movement.

II. QUADRUPED SERVO-ROBOT

The approach was to develop a 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 cyclic genetic algorithm was used to train this model to walk, emphasizing forward movement and the stability of the robot.

The robot used in creating the model was a quadruped servo-robot (Figure 1) developed by Michael Cantor in the Connecticut College robotics laboratory. The robot body and legs are made of masonite. It has standard hobby servomotors for actuators and uses a Basic Stamp II (from Parallax, Inc) for control.

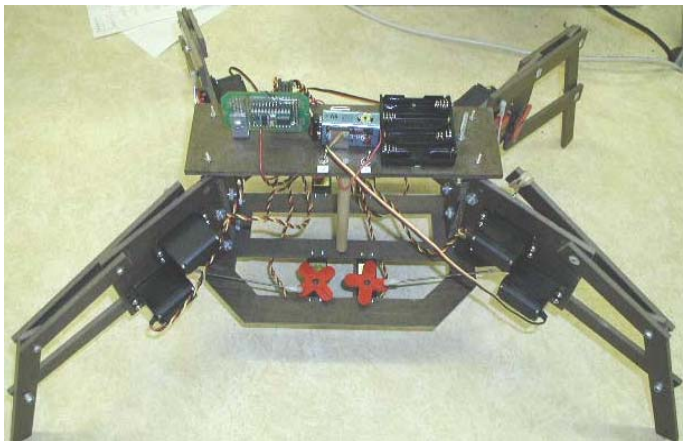

Figure 1. Photograph of the quadruped servo-robot used.

The robot has three degrees of movement per leg; in/out, up/down and forward/back movements. These movements are controlled by a control program running in the Basic Stamp. The program is made up of movement signals that are sent to all the servomotors. These signals are sent in parallel and tell each of 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. The movement signals or activations (Table I) are sent to the legs every 20 milliseconds.

TABLE I. INTERPRETATION OF THE 12-BIT ACTIVATION, WHICH IS MADE UP OF THE CONTROL SIGNALS THAT ARE SIMULTANEOUSLY TRANSMITTED TO EACH OF THE 12 SERVOS. 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

Activations are 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.

A sequence of these activations is needed to have continuous movement of the legs and a proper cycle 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.

The model of the quadruped servo-robot was created by taking accurate movements of its capabilities and storing the information in a lookup table. The information stored included separate measurements taken from each leg as shown in Table II. Distances are measured in millimeters and angles in radians.

TABLE II. 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.

Since the legs of the quadruped can move in/out while moving back/forward, the resulting 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. The resulting formula is shown in Equation 1, where $\emptyset$ is the angle, i is the current in/out distance of the leg, and d is the relative horizontal position of the leg.

$$d = i \sin \emptyset \quad (1)$$

III. CYCLIC GENETIC ALGORITHM

The Cyclic Genetic Algorithm is a type of Genetic Algorithm [6] that can be used to learn cyclic control programs [7]. It was developed to be a more suitable method for learning gaits for legged robots. The CGA differs from the standard GA mainly in the makeup of the chromosome. In a CGA the genes of the chromosome do not represent traits of the solution but represent tasks to be completed in a set amount of time. This, in effect, makes the CGA a method for directly evolving control programs. The chromosome of the original CGA had genes that represent three sets of tasks depending on location in the chromosome. A start section, a cyclic section, and a tail section are shown in Figure 2.

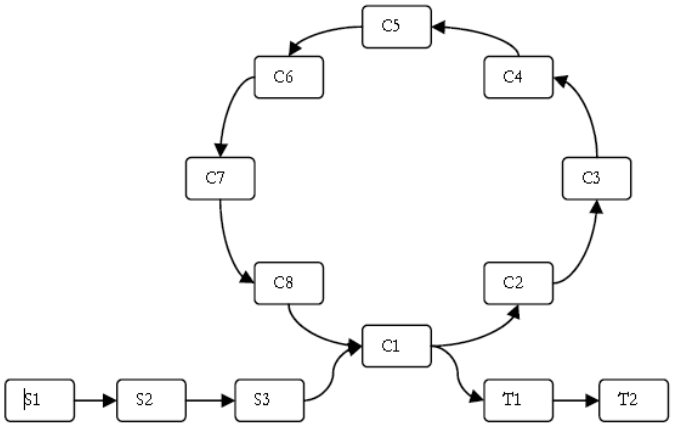

Figure 2. A CGA chromosome with start section, cyclic section, and tail section.

In the context of this research, 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 to provide a smooth translation back to the at-rest stance of the robot. The genetic operations of the CGA are typically the same as that of the GA except that crossover in the cyclic section is done at two points since it is, in effect, swapping a section of the cycle.

IV. CGA QUADRUPED GAIT PRODUCTION

A. Chromosome Structure

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

[C I {(R A)} {(R A)$_1$ (R A)$_2$ (R A)$_3$ … (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.

```
[3580 2882 {(40 1060)} {(43 1710) (4
1598) (20 1891) (19 259) (1 3097) (41
1052) (3 57) (0 3096) (0 1074) (0
1070) (0 1045) (0 17)}]
```

Figure 4. An example CGA chromosome structure written in Scheme 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 initiated 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 (simulated or actual).

C: Coordinators coordinated the three movements on each leg. For example, one coordinator made sure that if a 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: Twelve bit activation as described in Section II.

The sequence of activations in this chromosome are put directly into the Basic Stamp controller to generate a gait.

B. Gait Learning and Genetic Operations

During the learning stage, populations of 64 individuals were used. The initial population was created by randomly assigning numbers to all the parts of the chromosomes. Five separate tests were conducted with the CGA learning on random starting populations. The CGA evaluated each chromosome using the simulation model and assigned each a fitness score.

Fitness was calculated by executing 200 activations for each individual and determining its forward movement in this time. This number of activations assured that each individual was tested in a long enough time to ensure at least more than one whole step, meaning at least one cycle of the control

sequence was required. Fitness was computed by summing the fitness of the individual activations. The activation fitness equaled the forward motion 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; calculating the balance and probable legs on the ground from the model's current vertical position of each leg; applying the horizontal movement to alter the leg's state, but only counting legs on the ground in computation of the movement (fitness); 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 and then repeated as many times as required in the cyclic section.

Due to the nature of servos under pressure, they do not attain full movement when the activation is first applied. A single activation would just result in a twitch since one signal to a servo would produce minimal movement. It typically takes a sequence of four or more signals before each produces full movement. To simulate this, 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 after.

Subsequent generations of chromosomes were created by performing a stochastic (roulette wheel) selection to select two parent chromosomes, with the probability for selection based on fitness.

Crossover was performed at randomly selected points in the chromosome. The resultant offspring was subjected to 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. 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 for the remaining 4200 generations.

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.

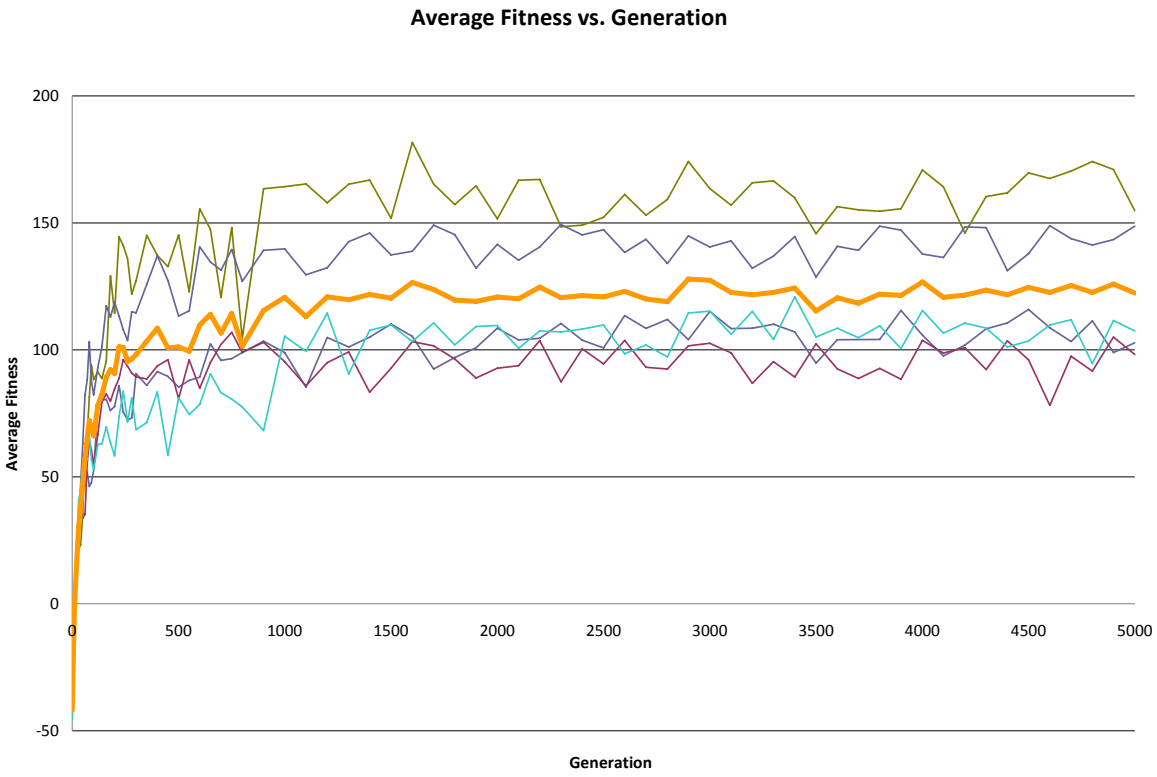

Figure 5. A plot of average fitness per population from populations selected from generation throughout 5000 generations. The average of the five trials is shown in bold.

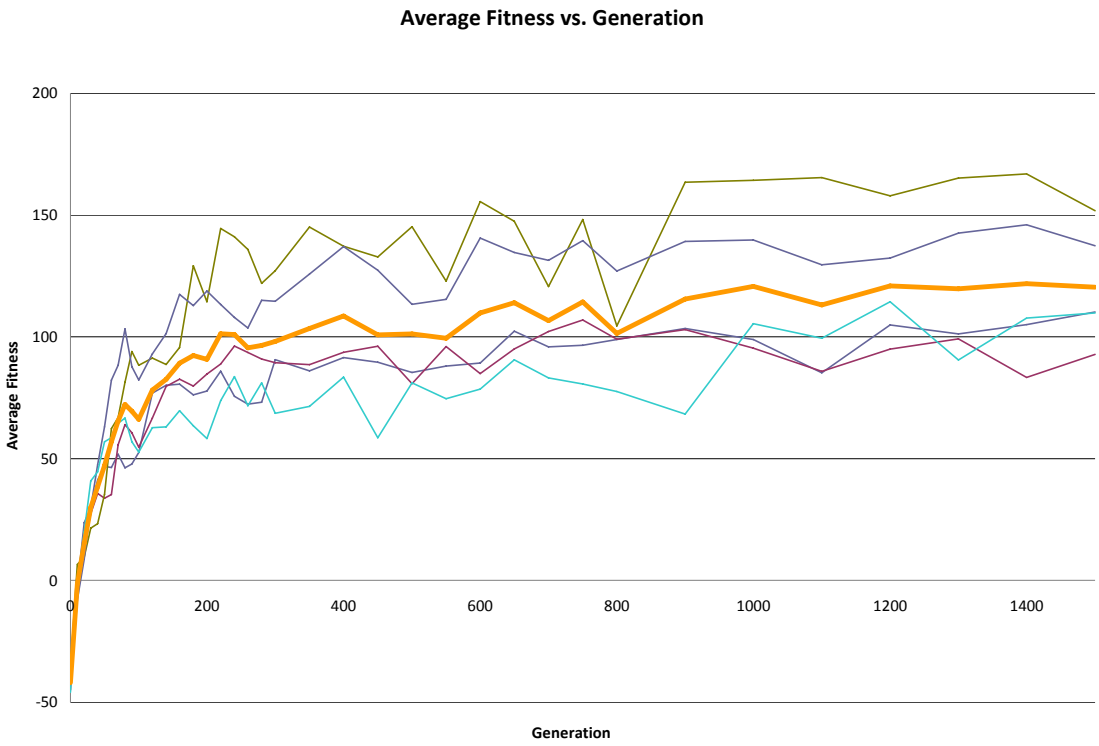

Figure 6. A plot of average fitness against generation number focused on the first 1500 generations. The average of the five trials is shown in bold.

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 successive at evolving more fit gaits in successive generations, and continued to slowly improve the control program until somewhere in the 1500 to 2000 generation range.

The best solutions from the final populations were downloaded to the Basic Stamp and allowed to control the movement of the robot. Although all runs resulted in reasonable gaits, there were two distinct walking methods that developed. One was to maintain static stability (three legs on the ground) as much as possible, with a slowdown in locomotion whenever it was time to reposition for another step. The other gait evolved was similar in nature to a slow trot or 2 beat gait of a horse. This CGA produced gait can be described as a diagonal-paired gait where a pair of diagonal legs moves in directions opposite to the other pair. This gait is shown in Figure 7.

With respect to up/down, forward/back and in/out; legs in black are moving back and down (providing forward thrust); legs in white are moving forward and up; and legs 2 and 3 move in when moving forward and out when moving back and Legs 0 and 1 do the opposite. Half way through the full movement of the legs in white, they start to move down as they keep moving forward to reposition for the next step.

This evolved gait, which was more effective than the other solution found by the CGA, 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.

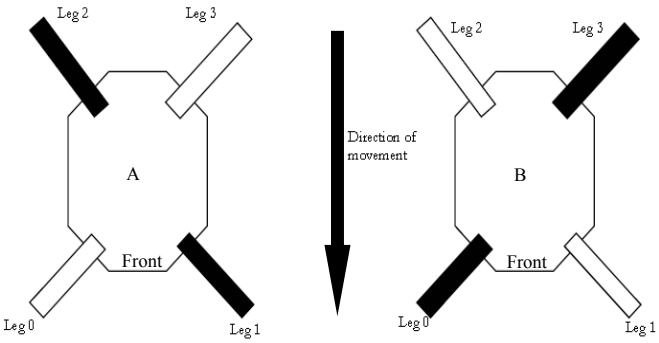

Figure 7. Movement of the robot based on the overall gait pattern learned by the CGA.

VI. CONCLUSIONS

The CGA can be used to learn gaits for the 4-legged robot with 3 degrees of freedom per leg. Learning done on a model of an actual robot produced a reasonable gait for that robot. This is the first use of CGA to learn gaits for a quadruped robot and the first time it has been used on robots with three degrees of freedom per leg. The CGA is capable of directly learning the control programs for multi-legged robot locomotion.

Although successful in this initial attempt at using a CGA to produce the quad gaits, further adjustments are needed so that the resultant gaits are more consistently near-optimal. In addition, future work will involve more extensive tests on the actual robot, plus different terrain environments and how the CGA can adapt to the changes in the capabilities of the robot will be considered.

Future work will continue to investigate the viability of the CGA for directly generating the control code for legged locomotion. Work will include the application of the CGA to learn gaits for six and eight legged robots with three degrees of freedom legs and reconfiguring the robot to have a Basic Stamp control each individual leg to allow for more complicated gait patterns.

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

[7] 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).