The Effects of Using a Greedy Factor in Hexapod Gait Learning

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

Abstract— **Various selection schemes have been described for use in genetic algorithms. This paper investigates the effects of adding greediness to the standard roulette-wheel selection. The results of this study are tested on a Cyclic Genetic Algorithm (CGA) used for learning gaits for a hexapod servo-robot. The effectiveness of CGA in learning optimal gaits with selection based on roulette-wheel selection with and without greediness is compared. The results were analyzed based on fitness of the individual gaits, convergence time of the evolution process, and the fitness of the entire population evolved. Results demonstrate that selection with too much greediness tends to prematurely converge with a sub-optimal solution, which results in poorer performance compared to the standard roulette-wheel selection. On the other hand, roulette-wheel selection with very low greediness evolves more diverse and fitter populations with individuals that result in the desired optimal gaits.**

Keywords- Genetic Algorithm; Cyclic Control; Hexapod; Greedy Selection; Gait; Evolutionary Robotics; Learning Control; Cyclic Genetic Algorithm

I. INTRODUCTION

Gait generation was selected for testing the greedy factor because it is a real problem that we are familiar with, and in previous research, it was determined that some greedy aspects to the search of the solution space could speed up learning. Gait control for legged robots involves the repetition of unique steps in the correct sequence with smooth transitions from one step to the next. This problem increases in complexity when more legs are involved with unique features and the goal is to generate adaptive (near) optimal gaits. Many algorithms have been developed to learn gaits including evolutionary computation.

Daoxing Gong et al [1], in a recent article, reviewed the suitability of evolutionary computation techniques in gait optimization for mobile legged robots. In a previous work [2], Parker et al. used Cyclic Genetic Algorithms (CGAs) to generate gaits for legged robots using minimal *a priori* knowledge. Variations of these CGAs have been successfully used in generating primitive gait instructions that have been directly applied to actual quadruped and hexapod robots. In this paper, we use CGA using roulette-wheel selection with and without greediness in a study to compare their effectiveness of a greedy factor in learning gaits for a hexapod robot. Our analysis is based on the efficiency of the gaits generated, the convergence time to the gaits learned, and the overall performance of entire population.

Selection is a significant part of genetic algorithms. During production of new generations two individuals are selected to parent one of the new individuals in the next generation. Even though we used a CGA that learns gaits for a hexapod robot, our study of greediness could as well be carried out with any other evolutionary method provided that selection is used. The first selection method used in this paper is the standard *proportionate reproduction* (roulette wheel selection) (Goldberg, 1989; Davis, 1991) where individuals are chosen for parenting a new offspring according to their objective fitness. Viewed as a wheel, in a roulette wheel each chromosome is given a slice of a circular wheel with an area proportional to its fitness ratio in the population. In this scheme, more fit individuals have more chances of being randomly picked out in the wheel for mating.

In this paper, we developed a second selection method by adding greediness to the roulette-wheel selection which can further bias selection towards or away from the individuals with best fitness in the population. In this selection method, each individual's fitness in a population is scaled with a greedy factor. This approach is aimed at emphasizing the individuals in a population according to their individual fitness (both absolute and ranking) in such a way that the desired traits in population are recognized and picked out in the evolution process for faster convergence.

Other conventional selection techniques have been developed and are mainly biased toward individuals with better fitness. Ranking selection [3], for instance, assigns to every individual a numerical rank based on fitness and selection is based on this ranking rather that the absolute differences in fitness. The advantage of this approach is in maintaining the population's diversity by preventing very fit individuals from gaining early dominance at the expense of the less fit ones. This on the other hand poses disadvantages by hindering and/or slowing the attempts to quickly reach the acceptable solutions. Another popular scheme is Tournament selection [3] where subgroups of individuals are randomly

created from the main population, and after the comparison of fitness, winners from these subgroups are selected for reproduction. Unlike roulette wheel and ranking selections, tournament selection does not include all members of the population. Other more greedy-like algorithms appear in a work by Ravindra K. Ahuja et al[4] where they developed *a greedy genetic algorithm* that incorporates many greedy principles in its design, but not in selection. Breeder Genetic Algorithm (BGA) developed by Heinz Muhlanbein and Dirk Schlierkamp- Voosen [5] is based on artificial selection of best parents for a model of the GA, which is a technique similar to that used by human breeders. This is a form of greedy selection, but excludes some members of the population. The selection method that we present involves the entire population, but gives even more priority to the fitter individuals than in roulette wheel selection.

In this paper, we use the CGA to develop usable gaits for an 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 gaits that were specific to the robot's capabilities. Tests are done using the two selection methods: standard proportional (roulette wheel) selection and roulette wheel selection with varying greediness.

II. The Hexapod Servo-Robot

This project was based on tests on a simulation model of an actual hexapod robot; therefore the approach was to develop a simulation model of the robot with data from its physical features and movement capabilities. This model (Table I), with information for each leg, represents the movement state of the robot at any instance in time. The cyclic genetic algorithm used this model to evolve a walk, emphasizing forward movement produced and stability of the robot. The activations (signal commands to the servo motors) learned by the CGA were used with the movement capabilities in the model to determine how much to change the movement state of the robot in the model. Actual physical features were measured and the movement rates were calculated by dividing the maximum throw distance by the minimum number of activations required to attain it.

TABLE I. INFORMATION USED IN THE MODEL OF THE ROBOT.

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.

The robot used was a six-legged servo-robot, which was developed for legged robot experimentation in our robotics lab. The hexapod robot has two degrees of freedom per leg; each leg can move up/down and forward/back. Twelve servos, two per leg, provide thrust and vertical movement. A learned control sequence of primitive instructions can be transmitted to the controller on robot and this learned gait will directly control the movement of the robot.

An input to the servo-robot, which we call an activation, 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 Figure 1, 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.

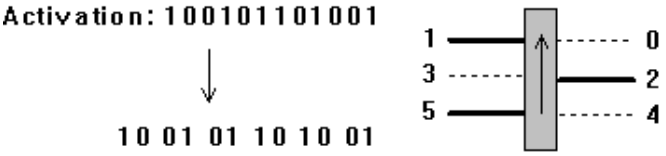

Figure 1: Results of a Robot Activation

III. Cyclic Genetic Algorithm

The Cyclic Genetic Algorithm is a type of Genetic Algorithm [3,6] that can be used to learn cyclic control programs [7]. Like standard genetic algorithms, CGAs consists of the three genetic operators (selection, crossover, and mutation) that are used to evolve a randomly generated initial population into more fit successive generations. The CGA was developed to be a more suitable method for learning gaits for legged robots by representation of a cycle of actions in its chromosomes. 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 (Figure 2). For gait generation, these tasks are activations that need to be sent to the servo-motors every 20ms. The CGA is, in effect, a method for directly evolving machine/assembly language 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. 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.

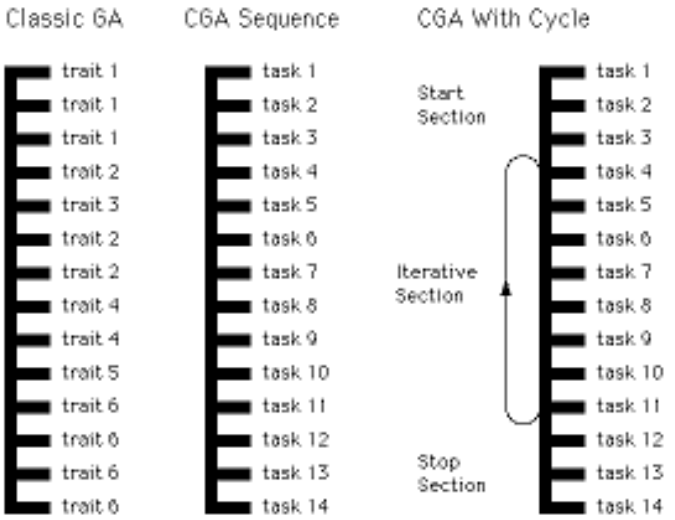

Figure 2: A comparison of chromosome structure between the classic GA and CGA.

IV. CGA CHROMOSOME

The chromosome (Figure 3) for this problem was made up of 4 parts: coordinator, inhibitor, start section, and cyclic section.

[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.

Coordinators and Inhibitors are part of robot's coordination mechanism, which evolves 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 the cyclic section before the control program was run in the robot (simulated or actual). Coordinators coordinate the three movements on each leg. For example, one coordinator makes sure that if a leg is going back it is either already down or moving in that direction. Another coordinator ensures that if a leg is going up it is either already forward or moving in that direction. The coordinator for all legs is a 12-bit binary. Inhibitors affect pairs or triples of legs. They prevent certain legs from moving in the same direction at the same time. For example, the inhibitor for legs 2&3 prevents both legs 2 and 3 from going back or forward at the same time. It allows 2 to move back, but inhibits 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. Repetitions represent the number of times to repeat the activations which are concurrent signals sent to the servos. Activations represent the signal sequence sent to the 12 servo-motors. Each activation signal is a 12-bit binary number, with 2 bits dedicated to each leg, one bit for up/down movement and the other bit is for the forward/back movement. 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 or forward. A signal of 1 would command any of the legs to go up or back.

V. GREEDINESS

The CGA in this project used two selection techniques. The first selection method was the standard roulette wheel selection method. Each chromosome is given a slice of the circular roulette wheel where the area of the slice is proportional to the chromosome's fitness ration in the population. To select a chromosome for mating, a random number is generated in the interval [0,100], and the chromosome whose segment spans the random number is selected. This process is like spinning a roulette wheel where each chromosome has a segment on the wheel proportional in size to its fitness.

The second selection method was to use a greedy factor, which adds varying degrees of greediness to the standard roulette wheel selection. This option involves scaling the entire fitness list in the population by a set of greedy factors. Each chromosome's fitness is multiplied by a particular greedy factor corresponding to its fitness relative to the rest of the chromosomes in the population. The first step involves sorting chromosomes in a population in descending order based on fitness. Then a list of greedy factors, of an equal size to the population, is created by using a particular function of choice that would determine the degree of greediness in the selection. In this algorithm, populations of size 64 were used in the training. Formula (1) was used in generating the lists of greedy factors; Formula (2) calculates a greedy fitness from an actual fitness value and its corresponding greedy factor.

$$f_g(i) = i^a \quad (1)$$

$$F_g(i) = F(i) * f_g(i) \quad (2)$$

In the formula fg(i) is the greedy factor corresponding to the position i in the fitness list, Fg(i) is greedy fitness, F(i) is an actual fitness score, *i* is an integer 1 to 64 representing fitness position in the population of size 64, and *a* is a greedy

constant – a real number. For the test reported in this paper, five greedy constants (a) were used, 0 (equivalent to the standard roulette wheel), 0.2, 0.5, 1 and 1.5 were used in generating the list of greedy factors depending on the value of i. In our tests, we wanted to increase the greediness of the selection so positive real numbers were used. In the case where one wanted to reduce the greediness of a typical GA, negative real numbers could be used.

VI. Training

During the learning stage, an initial population of 64 individuals was created by randomly assigning numbers to all the parts of the chromosomes. Three separate tests were conducted with the CGA learning on three random stating populations. In each of the three tests, roulette wheel selection was used with the five different greedy constants (0, 0.2, 0.5, 1, and 1.5). In order to accurately compare the performance of the five greedy constants, each set of five runs with different greediness was run on the same initial populations for each of the three starting populations. The CGA evaluated each chromosome using the simulation model and assigned each a fitness score.

Fitness was determined using a total of 400 activations for each individual; this assured each individual was tested a long enough time to ensure more than one full step was taken, requiring more than one cycle. Fitness was computed by summing the fitness of the individual activations after a total of 400. 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); and
- deducting fitness for lack of balance and/or asymmetry of movement;
- and repeating the process using 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 a load, adjustments were required in determining the back/forward movements when direction changes were first applied. Each forward/back movement in a new direction was set to produce ⅛ of the movement in the 1st pulse, ¼ in the 2nd pulse, ½ in the 3rd pulse, and full movement from the 4th pulse and on.

Subsequent generations of chromosomes were created by performing one of the five selection options, a stochastic (roulette wheel) or one of the four greedy selection choices to select two parent chromosomes.

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 up to generation 5000.

VII. Results

Three trials, with three randomly generated initial populations, were completed for each of the five greedy constants for selection method. From each trial, the average fitness per population was stored from selected generations. From these, we calculated the average of the average fitness per population for the three trials on each of the five greedy constants. We then plotted these results in graphs of average fitness per population versus the appropriate generation number. Figure 4 shows this graph for populations collected throughout the 5000 generations. Figure 5 shows the same graph focused on the pattern observed from the first 1500 generations. Runs with each of the five greedy constants are represented with 0G, 0.2G, 0.5G, 1G and 1.5G.

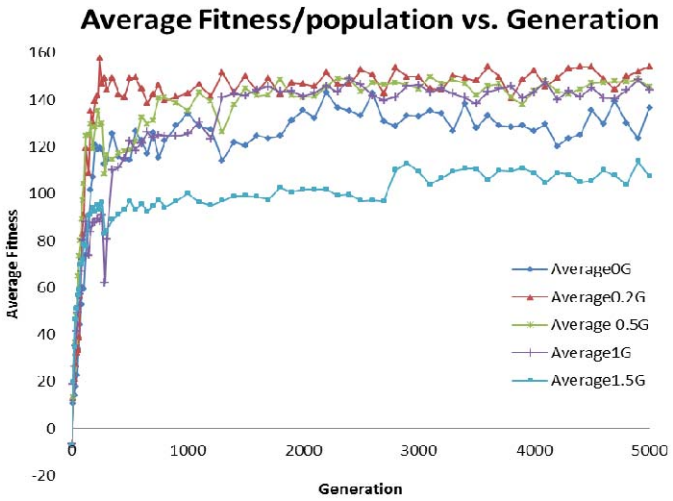

Figure 4: A plot of average fitness per population from populations selected from generations throughout the 5000 generations.

As can be seen on these two graphs, the learning pattern of the CGA can be observed. General observation shows that the evolution shows a rapid increase in average fitness in the first generations which then becomes steadier after about 2000 generations. All the three selection methods were successful at evolving more fit generations in successive generations, and continued to slowly improve the control program until somewhere in the 2000 to 4000 generation range. A closer look at the plot in Figure 5 reveals that selection with higher greediness (1G and 1.5G) evolve more rapidly in the first 100 generations compared to the lower greedy constants and the standard roulette wheel selection 0G). Looking at the entire

length of 5000 generations in Figure 4, selection with greedy constants 0.2, 0.5 and 1 evolves more fit populations than with standard roulette wheel selection which in turn evolves more fit populations than with selection with greedy constant of 1.5. This observation implies that very high greediness is not effective, and best entire population performance (which is better than standard roulette wheel selection) is achieved with low greediness, with greedy constants in the range 0<a<=1.

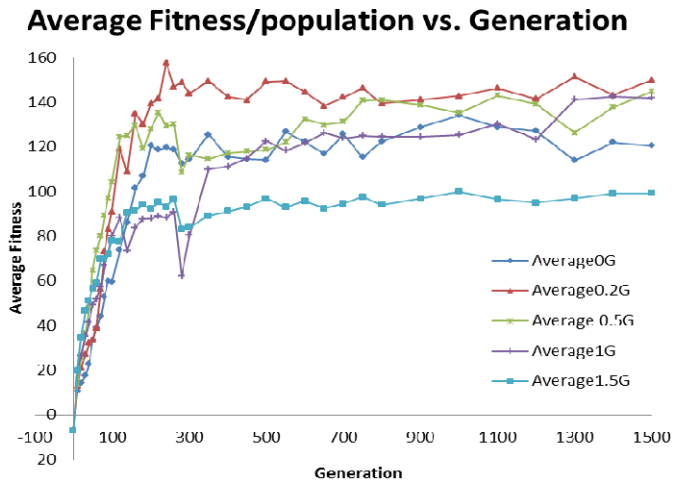

Figure 5: A plot of average fitness against generation number focused on the first 1500 generations.

From each trial, the best (most fit) individuals were stored from populations taken from selected generations throughout the 5000 generations. From this data, we calculated the fitness of each best individual and plotted two graphs of best individual fitness vs. generation number. The two graphs are shown in Figures 6 and 7.

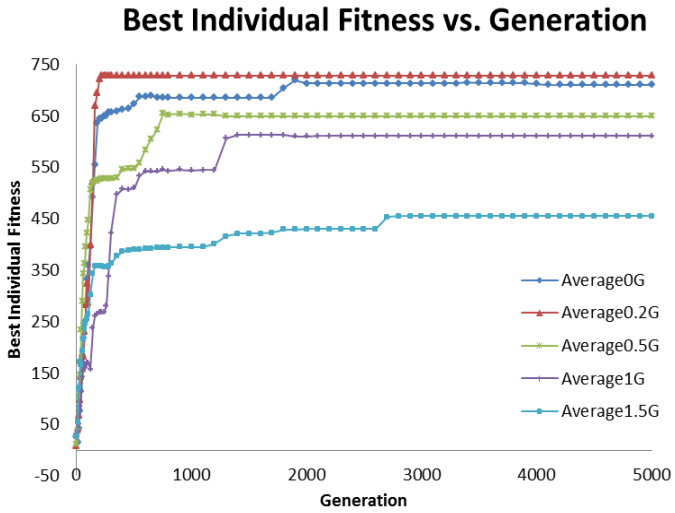

Figure 6: A plot of best individual fitness from populations selected from generations throughout the 5000 generations against generation number.

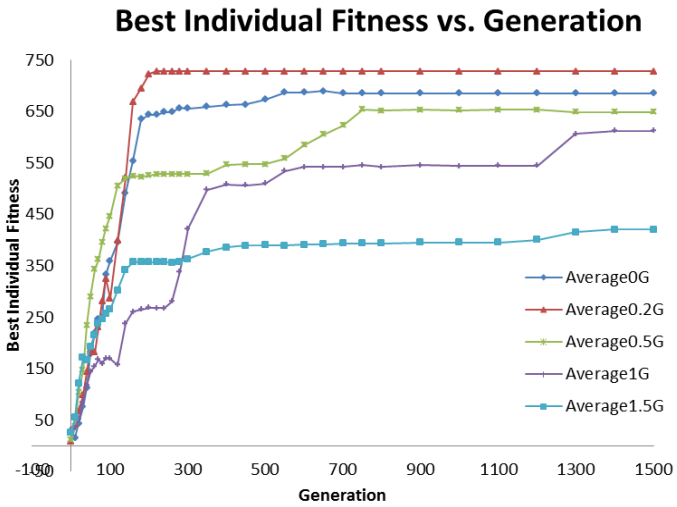

Figure 7: A plot of best individual fitness against generation number focused on the first 500 generations.

From Figures 6 and 7, we can make the observation that only selection with greedy constant 0.2 evolves individual gaits that are better than the gaits generated by the standard roulette wheel selection, which in turn evolved more fit best individuals than selections with greedy constants of 0.5, 1 and 1.5. These observations imply that, unless the greedy constant is very low (close to 0) gait learning with greedy selection tends to attain a premature convergence resulting in sub-optimal gaits. Learning without greediness, as with roulette wheel selection, results in evolving more diverse populations with gaits ranging from the best optimal gaits to gaits that are worse than the sub-optimal ones. Using a simulation, we have the capability of being able to display the move-by-move performance of any learned gaits; this display resembles the actual step-by-step movement pattern displayed by legs of an actual robot. Displays of the gaits represented by the best individuals from trials with selection with greedy constants of 0, 0.2 and 0.5 look similar to the optimal tripod gaits. In these gaits, legs 1,2,5 provide thrust, while legs 0,3,4 reposition. And when legs 0,3,4 are back in position, they provided thrust, while legs 1,2,5 reposition. On the other hand, the trials with greedy constants of 1 and 1.5 didn't evolve gaits that resembled the optimal tripod gaits most of the time. One of these resultant gaits with greedy factor 1 (1G in the graphs), was tripod and the remainder were such that legs 1,3,4 provided thrust, while legs 0,2,5 repositioned. The resultant 1.5G gaits varied with this sub-optimal gait the best produced.

VIII. Conclusions

All the five degrees of greediness demonstrated success in learning and generating gaits for the simulation model of the hexapod servo-robot. Our results displayed very interesting effects posed by greediness in selection operator of the CGA. Gait learning with very low greediness (greedy constant between 0 and 0.5) tends to evolve better than the standard roulette wheel selection (zero greediness), and results in both more fit general populations and a few best individuals with the most desired traits. On the other hand, too much greediness

(with greedy constants >0.5) tends to attain premature convergence with populations filled with individuals with sub-optimal fitness. Stochastic selection, which is solely based on proportional absolute fitness, as demonstrated by trials with the standard roulette wheel selection, generates more diverse populations with the best optimal individuals along with less fit ones. For the gait generation problem, selection with very low greediness turns out to be a more effective approach.

Even though the greedy constants of >=1 were not the best selection methods for gait generation, they may be better for other applications, which will be investigated in future work. A full range of greedy factors is possible. Other levels of greediness, such as greedy constants of <0 may also be explored for the gait generation problem or other problems. In addition, using this greedy constant allows the researcher to vary the greediness as evolution progresses, in an effort to apply selection pressure while maintaining diversity in the population.

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] 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).

[3] David E. Goldberg, and Kalyanmoy Deb., *"A Comparative Analysis of Selection Schemes Used in Genetic Algorithms,"* In Foundations of Genetic Algorithms. Edited by Gregory J.E. Rawlins.

[4] Ravindra K. Ahuja, James B. Orlin, and Ashish Tiwari., "A greedy genetic algorithm for the quadratic assignment problem," in *Computer & Operations Research 27 (2000) 917-934.* 1January 1999.

[5] Heinz Muhlenbein and Dirk Schlierkamp- Voosen, "Predictive Models for the Breeder Genetic Algorithm," Continuous Parameter Optimization. GMD P.O. 1316. D-5205 Sankt Augustin 1, Germany

[6] Holland, J., "*Adaptation in Natural and Artificial System*s," 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).