

A Hybrid Evolutionary Algorithm for Gait Generation of Sony Legged Robots

Dragos Golubovic and Huosheng Hu

Department of Computer Science, University of Essex
Colchester CO4 3SQ, UK

Email: dgolub@essex.ac.uk, hhu@essex.ac.uk

Abstract – This paper presents a hybrid evolutionary algorithm (EA) for developing locomotion gait of Sony legged robots. The selection of EA parameters such as the population size and recombination methods is made to be flexible and strive towards optimal performance autonomously. An interactive software environment with an overhead CCD camera is used to evaluate the performance of the generated gaits. The experimental results are given to show that the stable and fast gaits have been achieved

I. INTRODUCTION

It has been an ultimate long-time dream in robotics and AI fields to build robots with life-like appearance, behaviours and intelligence [2][12], reflected by many science fiction books and films. It is also an extremely challenging task. Until recently the success of the Honda Humanoid robots P2 & P3 has demonstrated its technical feasibility [1]. At the same time, Sony advanced-legged robots, AIBO, resemble the basic behaviour of dogs or cats [11], as shown in Figure 1. These robotic pets have been equipped with all the necessary hardware such as the brain, sensors and actuators. Their software enables them to have emotions, instincts, learning ability and capability to mature. Each AIBO turns out differently, as its behavioural patterns continuously change. This is because AIBO acts based upon its feeling and instincts then learns from the results of experience, until maturing. In a good mood, it may entertain you with its favourite performance such as stretching and chasing, whereas in a bad mood, it will lie on the floor and do nothing at all except sleep.

Each Sony AIBO Robot has 3 degrees of freedom (DOF) on its head and four legs. There is also 2 DOF for the tail and 1 DOF on its mouth. The body length (not including the head or tail) is approximately 18cm and the length of the leg (from shoulder to foot) is just under 12cm. On its head, there are a micro-camera, stereo microphone, infrared range sensor, and a touch sensor. There is also a touch sensor at the bottom of each foot. In addition to embedded CPU, its body houses a gyroscope and two accelerometers.

The movement of the AIBO legs is controlled by a locomotion module that controls the robot's gait with a user-specified set of real-valued parameters [9][10]. At every 20ms it updates the joint angles of the legs to the next position in their swing phase. This module also detects, and recovers from, the robot falling over.

Running a genetic algorithm entails setting a number of parameter values [7][8]. Finding settings that work well on a problem is a non-trivial task. If poor settings are used, the performance of a genetic algorithm can be severely impacted. We proposed a technique for setting the probabilities of

applying genetic operators during the course of a run. The technique involves adapting the operator probabilities based on their observed performance as the run takes place.

Two useful GA strategies were employed by researchers to find good operator probabilities. One was contained in DeJong's thesis work [3] and the other was described by John Grefenstette in [6]. The problem DeJong and Grefenstette were attacking was that of determining robust parameter settings for population size, operator probabilities, and evaluation normalisation techniques. The problem of adapting genetic algorithm parameters in general was simplified such that the only chromosomal representation technique used was the bit string, and only two operators were considered - mutation and crossover. This is a simpler problem than that of finding robust parameters for genetic algorithms across all representations, across the range of all possible operators, and so forth. Yet, even stripped down to the simple form just given, the problem is not easy to solve.

The first difficulty is that the problem is not precisely specified. We are asking for good values for the parameters, but measuring how good such values are is non-trivial since the range of potential applications of genetic algorithms using binary representations is infinite. DeJong solved this difficulty by simplifying the problem and used a test suite of function optimisation problems compiled by researchers in the field of function optimisation theory to stand in for the full range of possible domains. DeJong's choice of functions in the test suite was a good one.

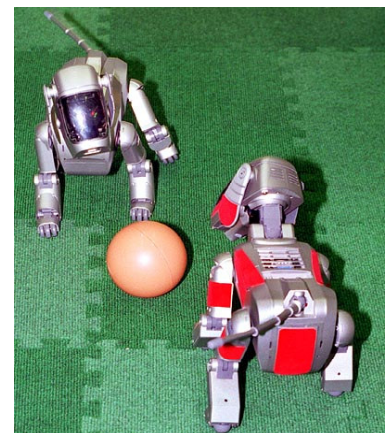


Fig. 1. Two Sony legged robots

The second difficulty in solving this optimisation problem given a test suite is that the evaluation function is noisy. Genetic algorithms are stochastic, and the same parameter settings used on the same problems by the same genetic algorithm generally yield different results. A consequence of this fact is that it can take a tremendous amount of computer time to find good parameter settings across a number of problems. DeJong's research was of immense benefit to people using genetic algorithms because it provided them with parameter values that were robust, in the sense that they had been validated on a number of different types of problems.

The rest of this document is organised as follows. In the following section we present an interactive software environment developed for our research purpose. Section 3 presents the gait model the choice of parameters for gait generation. In section 4, the evolutionary algorithm has been used to improve the performance of the gaits being generated. Experimental results are presented in section 4 to show the feasibility of the system. Finally, conclusions and future work are briefly outlined.

II. INTERACTIVE SOFTWARE ENVIRONMENT

Increased complexity and sophistication of advanced walking robots has led to continuing progress in building software environments to aid in the development of robust functionality. This is true not only because the physical construction of these robots is time consuming and expensive, but also because the evaluation and control of their gaits often requires prolonged training and frequent reconfiguration. To speed up the development cycle and decrease the design cost and the time required for gaits generation, many software environments have been developed [13][14]. The benefit of developing a suitable software environment includes the ability to record precise and voluminous data. Indeed, a flexible software environment plays an important role in many aspects of robotics research.

Since Sony AIBO robots have a large number of input and output parameters, their control design is complex. Therefore a modular approach is adopted here [4]. As shown in Figure 2, high-level control is conducted in a desktop PC (Pentium II 266) that is connected to the robot through a serial port (19200 baud). There are three main modules in it: a state reflector, a gait generator and an image interpreter. More specifically,

- **State reflector** -- An internal state reflector has been incorporated to mirror the robot's state on the host computer, which is an abstract view of the actual robot's internal state, such as sensor information from the robot and control commands from the host computer.
- **Communication routines** -- The designed controller communicates with the robot using a handshake mechanism, and sends commands to the robot.
- **Gait generator** -- The gait generator communicates with the robot by passing a sequence of arrays that are transformed into a sequence of robot movements. It creates different gaits in a form of a sequence of arrays.

On the robot side a debug box has been mounted on the robot's back and connected to the PC via a serial cable. Changes in any of these modules don't affect other modules, enabling users to split application development on several parts that can be carried out independently. More details can be found from [4].

III. GENERATION OF WHEEL-LIKE MOTION

The ability of legged machines to pass obstacles or to move on very uneven terrain in a low-invasive way is in

general better than that of wheeled robots. For example legged robots can walk up the stairs, walk across a pipeline on the ground, repair railway tracks or any post-accident relief work, walk along rocky and sandy terrain, much better than wheeled robots. However, wheeled robots can achieve much greater speed and mobility on flat grounds.

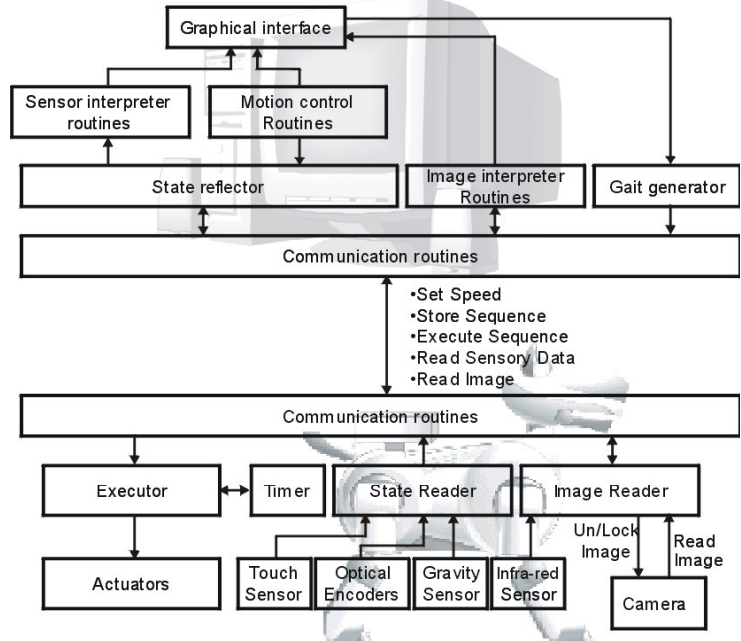


Fig. 2. Configuration of the software environment

A trajectory refers to both the path of the movement of the tip of a limb (paw), and the velocity along the path. Thus, a trajectory has both spatial and temporal aspects. The spatial aspect is the sequence of the locations of the endpoint from the start of the movement to the goal, and the temporal aspect is the time dependence along the path. The essential conditions for stable dynamic walking on irregular terrain as well as on the flat ground can be itemised with physical terms:

- the swinging legs not be prevented from moving forward during the former period of the swinging phase,
- the swinging legs must be landed reliably on the ground during the latter period of the swinging phase,
- the angular velocity of the supporting legs during their pitching motion around the contact points at the moment of landing or leaving should be kept constant, and
- the phase differences between the legs should be maintained in spite of delay of motion of a leg receiving disturbance from irregular terrain.

For the creation of wheel-like leg motion we used six parameters for front and six parameters for rear legs. These twelve parameters determine the spatial aspect of a robot trajectory. The 13th parameter is bound to temporary aspect and determines the speed of paw movement.

Six posture parameters used for designing the gait trajectory are presented in the table 1. These posture parameters are graphically shown in the figure 3. If posture

parameters for the front and rear legs are not identical, the top plane of the body will make the angle with the ground rather than horizontally.

In total, there are 13 real-valued parameters are used to determine a gait for the locomotion module. Table 1 lists some of these parameters, which are also the genes for individuals evolved by the evolutionary algorithm. These parameters specify the position and orientation of the body, the swing path and the swinging rate of the robot legs, the amplitude of oscillation of the body's location and orientation.

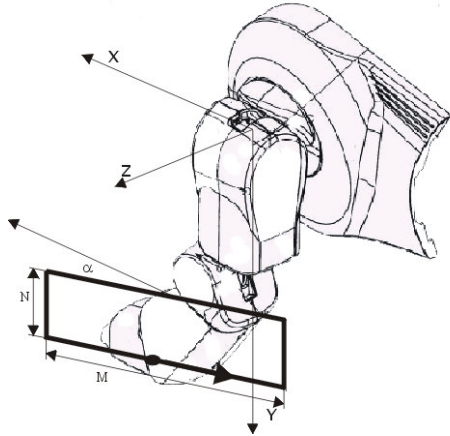


Fig. 3. Paw trajectory

TABLE I CONTROL PARAMETERS FOR GAIT GENERATION

	Parameter	Max value (mm)	Min value (mm)
Paw motion length	m	60	20
Paw motion width	n	50	10
X coordinate of COR	X _o	70	-20
Y coordinate of COR	Y _o	140	50
Z coordinate of COR	Z _o	40	-10
Paw rotation angle	α	90	-90
Paw moving speed	s	600ms/step	300ms/step

IV. EVOLUTIONARY ALGORITHMS

The basic operations first performed by the proposed algorithm are the generation and evaluation of an initial population. Subsequently, a main loop of operations is done, including selecting the best individuals, crossing them and applying mutation is done. The solution is usually the best string that is present in the final population.

A. Implementation of evolutionary algorithm

Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution. The algorithm we developed operates on a population of potential solutions applying the survival principle of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of

fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

We model natural processes, such as selection, recombination, mutation, migration, locality and neighbourhood. Figure 4 shows the structure of a genetic algorithm we implemented. Evolutionary algorithms work on populations of individuals instead of single solutions. In this way the search is performed in a parallel manner.

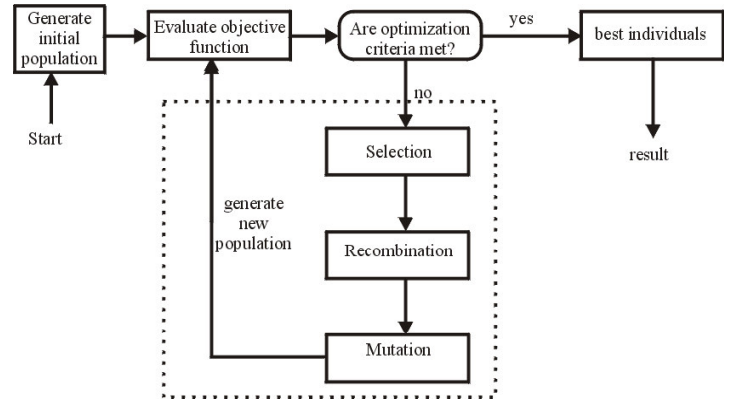


Fig. 4. Structure of a single population evolutionary algorithm

Selection -- It determines which individuals are chosen for mating (recombination) and how many offspring each selected individual produces. In our design we used two selection techniques (operators): roulette-wheel selection, also called stochastic sampling with replacement and selection of the fittest technique.

Regarding the roulette-wheel selection the individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness. A random number is generated and the individual whose segment spans the random number is selected. The process is repeated until the desired number of individuals is obtained (called mating population). This technique is analogous to a roulette wheel with each slice proportional in size to the fitness.

Recombination -- It produces new individuals in combining the information contained in the parents (parents - mating population). After recombination the new individuals of a population are created. We used four recombination techniques in parallel.

Guaranteed-Uniform-Crossover -- In this technique we created two children from two parents by deciding randomly, gene by gene, which child receives the corresponding value from which parent.

Guaranteed-Average -- We created one child from two parents by averaging their fields. We choose randomly which integer to use if the average lies between two integer values. Child has been used only if it differs from both parents.

Guaranteed-Big-Creep -- We created one child from one parent by passing down the parent's fields and, with 20%

probability, altering a field by replacing it with a value that is 1, 2 or 3 units above or below the field value. The amount and the direction of the creep are selected randomly. Child has been used only if it differs from both parents.

Guaranteed-Little-Creep -- We create one child from one parent by passing down the parent's fields and, with 10% probability, altering a field by replacing it with a value that is 1 unit above or below the field value. The direction of creep is selected randomly. We limited creep with maximum and minimal values

Mutation -- After recombination every offspring undergoes mutation. Offspring variables are mutated by small perturbations (size of the mutation step), with low probability. The probability of mutating a variable is set to be inversely proportional to the number of variables (dimensions). The more dimensions one individual has as smaller are the mutation probability. The size of the mutation step is usually difficult to choose. The optimal step size depends on the problem considered and may even vary during the optimisation process. Small steps are often successful, but sometimes the bigger steps are quicker.

- mutated variable = variable \pm range-delta; (+ or - with an equal probability)
- range = 0.5-domain of variable; (search interval),
- delta is the value between 0 and 1 with probability 1/m.

B. The Adaptation Mechanism

The idea described here is a hybrid approach that adapts the probability that genetic operators described above will be applied in reproduction based on the performance of the operator's offspring.

The first intuition underlying the adaptation mechanism is that an adaptive mechanism should alter the probability of applying an operator in proportion to the observed performance of the individuals created by that operator in the course of a run.

These intuitions were implemented in the following way. Whenever a new population is generated using selection operator *Soi* and recombination operator *Roj* a pointer is created to the class *CStat* responsible for keeping statistical data of the current population. After evaluation of all generation members and their fitness functions we evaluate the performance of whole generation and store calculated values in the list. Each operator combination *Soi* and *Roj* is associated with average fitness ΔF_{ij} , the fitness of the best member produced with these operators M_{ij} and the weight of current operators W_{ij} , which determines probability of appearance for those operators.

At this point check was made to determine whether the new members were better than the current best member of the population produced with these operators. If so M_{ij} is updated. The average fitness ΔF_{ij} is also updated. The average fitness and the best member of the current operators are then compared to average fitness and best members of all the other operator combinations. According to this comparison weights are modified according to equations (1) and (2) below.

$$W_{ij} = W_{ij}' + \frac{1}{2} \left(\Delta F_{ij} - \frac{\sum_{m \neq i} \sum_{n \neq j} \Delta F_{mn}}{s * r - 1} \right) + \frac{1}{2} \left(\Delta M_{ij} - \frac{\sum_{m \neq i} \sum_{n \neq j} \Delta M_{mn}}{s * r - 1} \right) \quad (1)$$

All other weights are modified accordingly so sum of all weights should equal 1.

$$\sum_{m \neq i} \sum_{n \neq j} (W_{mn} = W_{mn}' - \frac{W_{ij} - W_{ij}'}{s * r - 1}) \quad (2)$$

where *s* is the number of selection methods and *r* is the number of recombination methods.

The mechanism is that purely local measures of performance are not sufficient. One must reward an operator that produces a good child, but one must also reward an operator that set stage for this production. This intuition is grounded in the sort of situation that often happens toward the end of a run, when one's population is mostly converged. That is the reason we are using both average fitness and best fitness of the applied operators for modifying weight values.

C. Evaluation Mechanism

Initially we created two members of a population. The first task of the controller is to choose operators for the creation of the entire population. Operators are weighted and therefore those operators that produced better results in the past have more chance to be chosen again. After generating entire population evaluation mechanism is up and running.

The controller picks up the next member of the population and start evaluation process. Before executing the gait determined by population member genes snapshot should be taken from the CCD camera mounted 2m above the pitch. The initialisation of the robot movement is due to the communication routines and the executor. After several steps the robot notifies controller about completion of gait movements. The controller takes another snapshot of the field and analyses the change of robot position during the execution of the gait. According to extent and stability of robot movement a fitness value is produced.

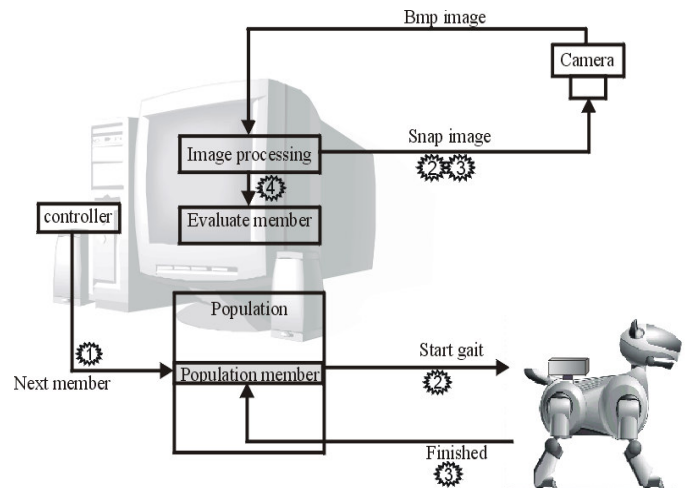


Fig. 5 Evaluation process of generation member

If the optimisation criteria are not met the creation of a new generation starts. Individuals are selected according to their fitness for the production of offspring. Parents are recombined to produce offspring. All offspring will be mutated with a certain probability. The fitness of the offspring is then computed. The offspring are inserted into the population replacing the parents, producing a new generation. This cycle is performed until the optimisation criteria are reached.

D. Fitness Function and Optimisation

The construction of an appropriate fitness function is very important for the correct work of the GA. This function represents the problem environment, that is, it decides how well the string solves the problem. There are two primary optimisation criteria for gait generation we have been concerned with. Our gait should be fast and stable at the same time. While the speed can be accurately measured stability criteria is fuzzy. We want for our robot to be stable as much as possible but this criterion is to the opposition of the speed requirements. To measure stability during population member run we used readings from gyro-sensor. At the end of run, the variance of gyro readings is calculated from equations (3) and (4) below. The minimisation of the variance and maximisation of the achieved speed are targeted.

AIBO robot's gyro sensor returns 3 double values ranging from -9 to 9. During execution of the gait created with member genes we sample readings from the gyro sensor. Our aim is to get the gait with minimal oscillations but it also should be a fast gait. Stability is important in our case because robot should be able to track the ball in order to play football well even when he is moving. The number of samples taken from the gyro during the robots run with one gait varies between 20 and 30.

$$EX = \frac{\sum_{i=1}^s X_i}{s}, \quad EY = \frac{\sum_{i=1}^s Y_i}{s}, \quad EZ = \frac{\sum_{i=1}^s Z_i}{s} \quad (3)$$

$$G = \sqrt{\frac{\sum_{i=1}^s (X_i - EX)^2 + \sum_{i=1}^s (Y_i - EY)^2 + \sum_{i=1}^s (Z_i - EZ)^2}{s}} \quad (4)$$

where s is the number of samples from the gyro.

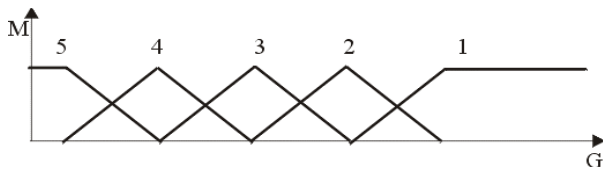


Fig. 6 The Fuzzy membership functions

Because stability requirements were fuzzy we decided to use fuzzy logic for its evaluation. On the basis of variance of gyro readings we calculate multiplication factor that ranges from 5 to 1, as shown in Figure 6.

Fuzzy functions are calculated by multiplying the speed of the robot with the multiplication factor. In case the robot falls over, a multiplication factor 1 will be assigned regardless to the gyro readings.

V. EXPERIMENTAL RESULTS

The evaluation takes place inside of AIBO robot's football pitch. Each generation member produces a gait that runs for 5 steps. Time necessary to finish one evaluation varies because these 5 steps won't be executed always for the same time. The 13th gene specifies the speed. If the robot falls over, it is fully capable of getting up and continuing its execution. After executing one member evaluation takes place and the speed and stability parameters are produced which qualitatively determines fitness values. The overhead camera records offset from the starting position and the changes when the robot to return to the starting position. In order to do that, an appropriate number of steps, e.g. forwards and backwards, are executed by the robot. When We used a population size of 20 and run it for 50 generations.

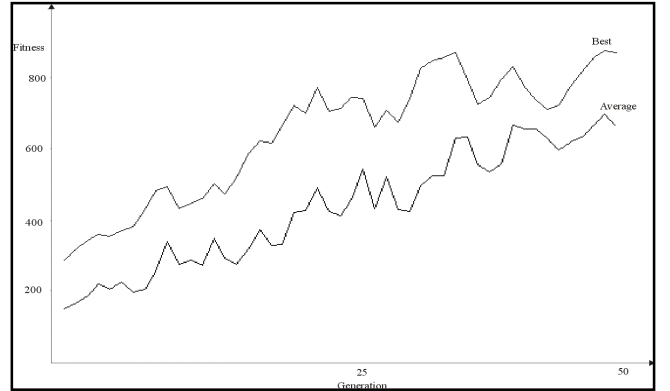


Fig. 7 The results of gait generation based on GA

In the beginning, most of the population members didn't perform very well. Some of them didn't even walk in the straight line or they even walked backward. Some of them were also causing the robot to fall. By the end of the experiment, the performance of the members significantly improved. The best member manages to walk in the straight line with good stability with the speed of 7m per minute.

We already mentioned that we are dealing with contradictory demands in terms of speed and stability. It is interesting to see the improvement of speed and stability separately over a sequence of generations and the affects it has on the overall fitness function. In this case we are observing speed and stability results of the member of population with the greatest fitness value.

In the first 20 generations stability measure tend to have constant growth but it has negative affect on the speed increase. Still at that time speed has more than doubled its value from the start of the experiment. In the 24th generation stability reaches its maximum showing no increase and even tending to fall towards the end of the experiment. On the other hand small decrease in stability values has positive effect on the increase of speed that manages in the next 20 generations to double its value.

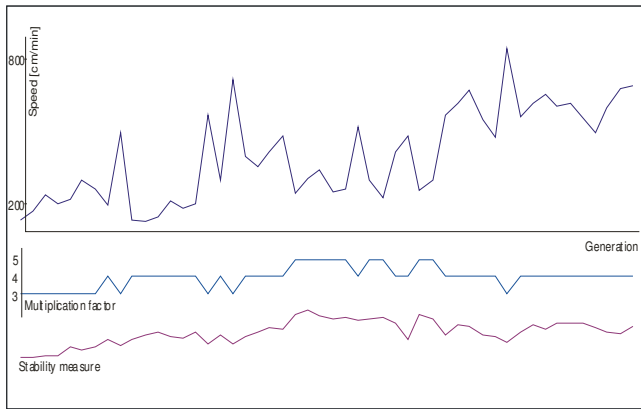


Fig. 8 The speed and stability results (separately)

Periodical falls in average fitness from one generation to the next are caused mostly because of some generation members whose execution was followed by losing balance of the robot decreased average fitness of whole generation.

VI. CONCLUSIONS AND FUTURE WORK

Based on this hybrid approach, Sony robots can learn good walking behaviors with little or no interaction with the designers. Once the learning method is put into place, the module can learn through its interaction with the world. The mutating and combination behaviors of the Genetic Algorithms allow the process to develop to a useful behaviour over time. There are however some problems with this approach. First it is very time consuming to have the physical robot performing a task hundreds of times in order to perform well. It would be much more beneficial to be able to perform this training in a simulated environment. However, such an environment normally misses some key elements of the behaviour of the physical world and such test results were found to be less optimal. Another problem with this approach is that it tends to discover local optima. As stated from the experiments, it required physical interference by the environment to force the AIBO to raise its feet higher during training. The resulting gait from this training proved to be a better solution than the non-interference training for movement over all types of surfaces, pointing to a local optima being discovered in the non-environmental interference situation.

EAs are heuristics and thus they do not ensure an optimal solution. The behaviour of these algorithms is stochastic so they may potentially present different solutions in different runs of the same algorithm. The mechanism described here has several features that should be noted. It allows rapid parameterisation of operator probabilities across the range of potential genetic algorithms and operator set. As described here, it is tailored to a steady state reproduction scheme. It would not be literally applicable to problems with noisy evaluation functions. The most significant parameter impacting solution of the test problem was not the relative probabilities of the operators. Rather, it was a parameter having to do with the selection of parents. This parameter, too, can be adapted usefully over the course of a genetic

algorithm run. We will address these issues in the next stage of research.

VII. ACKNOWLEDGMENT

The authors gratefully acknowledge the contributions of Dongbing Gu and Bo Li.

VIII. REFERENCES

- [1] M. Asada and H. I. Christensen, "Robotics in the home, office and playing field", in Proceedings of IJCAI, Stockholm, 30 July - 5 August 1999, pp. 1385-1392.
- [2] M. Brady and H. Hu, The Mind of a Robot, Philosophical Transactions: Physical Sciences and Engineering, the Royal Society, Vol. 349, No. 1691, London, 1994, pp. 15-28.
- [3] P. Doerchuk, W. Simon, V. Ngyyen, "A Modular Approach to Intelligent Control of Simulated Joint Leg", IEEE Robotics & Automation Magazine, June 1998, pp. 12-20.
- [4] D. Golubovic and H. Hu, An interactive software environment for gait generation and control design of Sony legged robots, in Proceedings of International Symposium on RoboCup, Fukuoka, Japan, June 2002.
- [5] D. Gu and H. Hu, Evolving Fuzzy Logic Controllers for Sony Legged Robots, Proceedings of the RoboCup 2001 Int. Symposium, Seattle, Washington, 4-10 August 2001.
- [6] D. Gu and H. Hu, Fussy Behaviour Learning for Sony Legged Robots, EUSFLAT 2001 - European Society for Fuzzy Logic and Technology Conference, De Montfort University, Leicester, UK, September 5-7, 2001.
- [7] H. Hu and D. Gu, Reactive Behaviours and Agent Architecture for Sony Legged Robots to Play Football, International Journal of Industrial Robot, Vol. 28, No. 1, ISSN 0143-991X, January 2001, pp. 45-53.
- [8] H. Hu and D. Gu, A Multi-Agent System for Cooperative Quadruped Walking Robots, IASTED International Conference Robotics and Applications (RA 2000), Honolulu, Hawaii, USA, August 14-16, 2000.
- [9] H. Kitano, M. Fujita, S. Zrehen, K. Kageyama, "Sony Legged Robot for RoboCup Challenge", in Proceedings of IEEE International Conference on Robotics and Automation, Leuven, Belgium, 1998, pp. 2605-2612.
- [10] P. Maes, Artificial life meets Entertainment: Lifelike Autonomous Agents, Communication of ACM: Special Issue on New Horizons of Commercial and Industrial AI, 1995, pp. 108-114.
- [11] M. Maza, J. Fontaine, M. Armada, P. Gonzalez, "Wheel+Legs- A New Solution For Traction Enhancement Without Additive Soil Compaction", IEEE Robotics and Automation Magazine, June 1998, pp. 26-32.
- [12] J. Reichler and F. Delcomyn, "Dynamics Simulation and Controller Interfacing for Legged Robots", International Journal of Robotics Research, Vol.19, No. 1, 2000, pp. 42-58.
- [13] M. Veloso, et al., Vision-served localisation and behaviour-based planning for an autonomous quadruped legged robot, AI Magazine, The AAAI Press/MIT Press, Spring 2000, pp. 387-394.
- [14] K. DeJong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", PhD Dissertation, University of Michigan, 1975.
- [15] J.J. Grefenstette, "Optimization of Control Parameters for Genetic Algorithms", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-16, No. 1, January/February, 1986.