Parameter Optimisation of an Evolutionary Algorithm for On-line Gait Generation of Quadruped Robots

Dragos Golubovic and Huosheng Hu

Department of Computer Science, University of Essex
Colchester CO4 3SQ, United Kingdom
Email: dgolub@essex.ac.uk, hhu@essex.ac.uk

Abstract– This paper presents a hybrid evolutionary algorithm (EA) for developing locomotion gaits of Sony legged robots. An online training algorithm is used for generating gaits for quadruped walking robots based on a hybrid approach that changes the probability of genetic operators in respect to the performance of the operator's offspring. The probability of applying an operator changes in proportion to the observed performance of the individuals created by that operator in the course of a run. The selection of EA parameters such as the population size and recombination methods and mutation parameters are made to be flexible and strive towards optimal performance autonomously. An overhead CCD camera is used to evaluate the performance of the generated gaits on-line while the robot is playing a football game. Robot is learning to walk on its own without any human interference.

1 Introduction

Legged robots show significant advantages over wheeled robots since they can traverse in uneven and unstructured environments. This high mobility stems from the fact that in contrast to wheeled or tracked robots legged robots discretely contact with their environments via their legs. This, however, inevitably causes highly complicated dynamics between the robots and their environments. On the other hand wheeled robots can achieve much greater speed and mobility on flat grounds [6].

We took advantage of this observed property of wheeled robots and we generated walking motions for the robot that mimic the movements of wheels but still we tried to preserve flexibility and adaptability of legged robots (fig. 1). By placing some minor restrictions on how the leg motions are generated, we develop an easily computable classification of the best gait patterns to be used for any given motion (forward, backward, sideways or rotational) of Sony AIBO robot.

The objective of the present work is to make efficient evolutionary algorithms (EA) easier in order to solve large instances of difficult combinatorial optimization problems within an acceptable amount of time. No known technique allows one to exactly solve within an acceptable amount of time, such difficult combinatorial optimization problems. Moreover, traditional approaches that are used to find sub-optimal solutions are not always satisfactory since they are easily attracted by local optima. Evolutionary algorithms (EA) that are inspired by natural evolution mechanisms explore different regions of the search space concurrently. They are thus often trapped in a local optimum and are not well suited to treat difficult combinatorial optimization problems. They are also greedy in computation power and memory space.

Thus, it is in general extremely difficult to design gait controllers for legged robots. So far various methods have been proposed to construct legged-robot gait controllers, whilst very few studies have investigated the design of controllers considering the interaction dynamics with the environments [1][2][5]. However, since the above approaches are based on a hand-crafted manner, it is questionable whether or not these approaches will be still feasible (i.e. easily implemented) as the complexity of the desired task and the interaction dynamics increases. On the other hand, recently the Evolutionary Robotics (ER) approach has been attracting a lot of concern in the field of robotics and artificial life [3][4]. In contrast to the conventional approaches where designers have to construct controllers in a top-down manner, the methods in the ER approach have significant advantages since they can autonomously and efficiently construct controllers by taking embodiment (e.g. physical size and shape of robots, sensor/motor properties and disposition etc.) and the interaction dynamics between the robot and its environment into account.

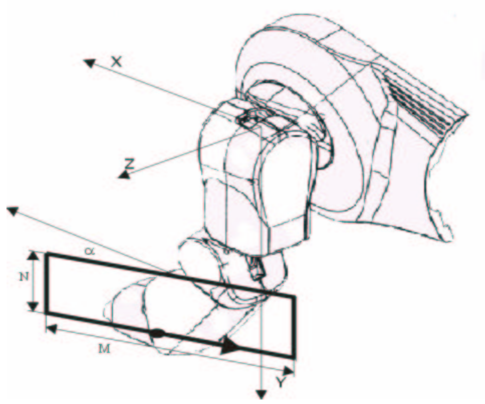


Fig. 1 Paw trajectory

Running a genetic algorithm entails setting a number of parameter values. 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 [9] and the other was described by Grefenstette in [10]. 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, in that the only chromosomal representation technique used was the bit string, and only two operators were considered - mutation and crossover.

The rest of this paper is organised as follows. Section 2 describes the fundamental issues for a walking robot that is statically and dynamically stable. Section 3 presents the genetic modelling of wheeled motion for Sony AIBO robots. The evolution mechanism is addressed in section 4. Then, section 5 investigates the issue of multi-objective optimisation algorithms. Experimental results are presented in section 6 to demonstrate the feasibility. Finally, a brief conclusion and future work are given in section 7.

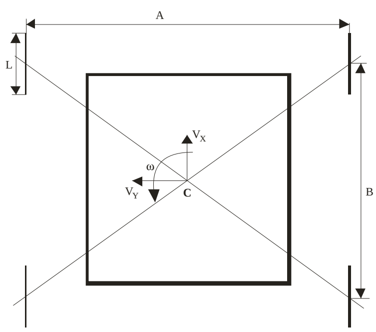
2 Statically and Dynamically Stable Walking Gait

A gait is a cyclic, periodic motion of the joints of a walking robot. For quadrupeds, statically stable walking gaits have at least 3, and dynamically stable gaits at least 2 legs, in contact with the ground at all times [8]. The legs of the robot cycle through the gait with a particular phasing. We present a method to choose the phasing according to the direction of motion of the centre of mass of the robot.

A set of requirements were drawn up that the walking needed to satisfy. These included:

§ Omni-directional motion - The robot should be able to walk in all directions.

§ The gait should move the robot as fast as possible. Since the robot would be moving forward most of the time, it was decided that the robot should be able to move forward at the maximum speed possible.



V_x and V_y - are translation velocities
ω - rotational velocity
c – center of gravity
L – step length

Fig. 2 Robot Top View

§ Stability - It was important that the robot should remain stable at all times.

The method should be computationally fast and implementable in real-time. Computational efficiency was important because of limitations of the speed of the onboard processor and available memory.

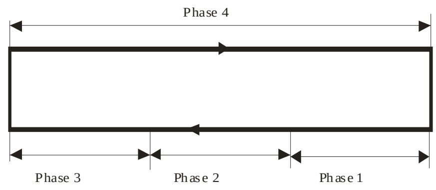


Fig. 3 Foot placement curve

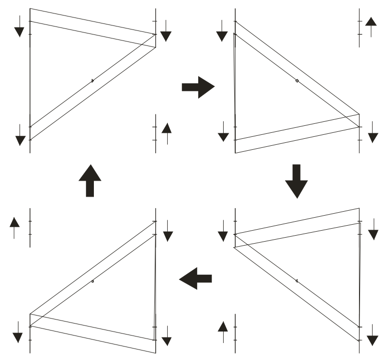


Fig. 4 Walk forward

Statically stable ¾ duty cycle provides a stable gait for at least one type of desired motion. Only in phase 4 a leg is not in contact with the ground and we are trying to make statically stable gait we concluded that each of the four legs should be in the different phase. Leg has to follow the sequence of phases 1-2-3-4 (fig. 3). That leaves us with 6 possible phasings of the legs of quadruped walking using ¾ duty cycle.

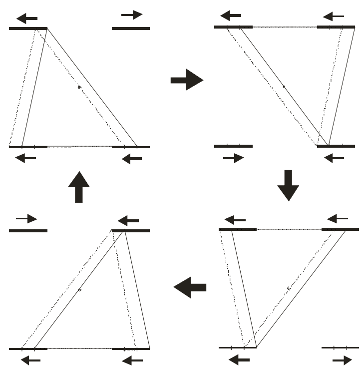


Fig. 5 Side walk

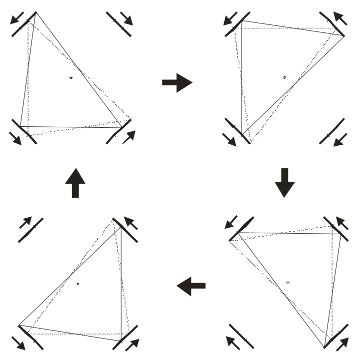


Fig. 6 Rotation to the right

Note that (V_x, V_y, ω) describe movement of the robot body. To simplify problem lets assume that only one of these velocities has positive or negative value while the other two are zero. In that case we can find at least one statically stable phasing (among 6 possible) for forward, backward, left, right and rotational movement of the robot.

The choice of phasing that maximize stability margin is given in the following table:

Leg	Front-left	Hind-left	Hind-right	Front-right
Forward	1-2-3-4 phas.	2-3-4-1	4-1-2-3	3-4-1-2
Side walk	1-2-3-4	3-4-1-2	2-3-4-1	4-1-2-3
Rotation	1-2-3-4	2-3-4-1	3-4-1-2	4-1-2-3

Tab.1 Leg Phasings

Each phase is done in ¼ of a step cycle T. According to this phasing we can find out maximum translational and angular velocities that can be achieved (equations 1-4)

$$V_x = \frac{4}{3}\frac{L}{T}, \text{ for } V_y = 0 \text{ and } \omega = 0 \quad (1)$$

$$V_y = \frac{4}{3}\frac{L}{T}, \text{ for } V_x = 0 \text{ and } \omega = 0 \quad (2)$$

$$\omega \approx \frac{4}{3}\frac{arctg(\frac{L}{\sqrt{a^2+b^2}})}{T}, \text{ for } V_x = 0 \text{ and } V_y = 0 \quad (3)$$

$$V_x = \frac{4}{3}\frac{L}{T}\cos\alpha,\ V_y = \frac{4}{3}\frac{L}{T}\sin\alpha, \text{ for } \omega = 0 \quad (4)$$

Dynamic walking gaits are faster than statically stable gaits. However, they are harder to control and implement. We present here details of the experimental implementation of dynamic walking for a quadruped. Dynamic walking is similar to that described earlier for static walking. It involves generation of foot placement curves that are a function of the direction of motion of the robot. Inverse kinematics is then used to calculate the joint motions required to generate the foot placement curves for the particular gait.

Maximum duty cycle that can be achieved through dynamic walk is ½.

dc – duty cycle (¾ , ½)

$T_{P2} = T_{P4} = (1-dc)T$, $T_{P1} = T_{P3} = (dc-½)T$, for $dc \in$ (¾ , ½)

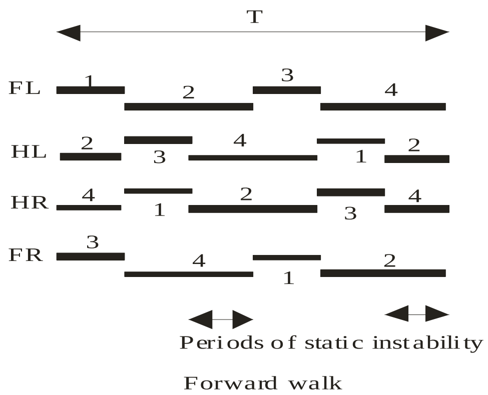


Fig. 7 Period of static instability

Equation 5 presents period of time in which the robot is statically instable regarding duty factor for forward, side an d rotational walks (fig. 7).

$$IS_{fwd} = IS_{sid} = IS_{rot} = (4dc-3)T \quad (5)$$

An important part of omni-directional walking is the ability to transition between two gaits. We have found that changing phasings of the legs can be solved by using a transitional stance, namely a stance at which the body momentarily comes to rest at a reset position. We have chosen for these purposes an upright stance that position the feet at each of the foot placement curve's center points. Although this results in some small shift in body position, the effect is generally negligible.

3 Genetic Modelling of Wheeled Motion

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 and the 14^{th} represents duty factor. Duty factor can take values between ¾ and ½.

Six posture parameters used for designing the gait trajectory are presented in the table 2. These posture parameters are graphically shown in the figure 1. 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 14 real-valued parameters are used to determine a gait for the locomotion module. Table 2 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.

	Paramet er	Max value (mm)	Min value (mm)
Paw motion length	m	60	20
Paw motion width	n	50	10
X coordinate of COR	Xo	70	-20
Y coordinate of COR	Yo	140	50
Z coordinate of COR	Zo	40	-10
Paw rotation angle	α	90	-90
Paw moving speed	s	600ms/step	300ms/step
Duty factor	dc	¾	½

Tab. 2 – Control Parameters for Gait Generation

Selection 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.

Recombination 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, Guaranteed-Average, Guaranteed-Big-Creep, Guaranteed-Little-Creep).

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.

4 Evolution Mechanisms

After generating entire population the controller picks up the next member of the population and start evaluation process by passing the genes of the selected member to the robot. On the basis of those genes the robot will generate all its gaits. Before executing the gait determined by population member genes snapshot should be taken from the CCD camera mounted 2m above the pitch. After each steps (rotational or transitional) the robot notifies controller about completion of step movement and about the type of movement that has been undertaken (rotational or translator with appropriate angles). The controller takes another snapshot of the field and analyses the change of robot position during the execution of the gait. After that process has been repeated several times according to extent and stability of robot movement a fitness value is produced.

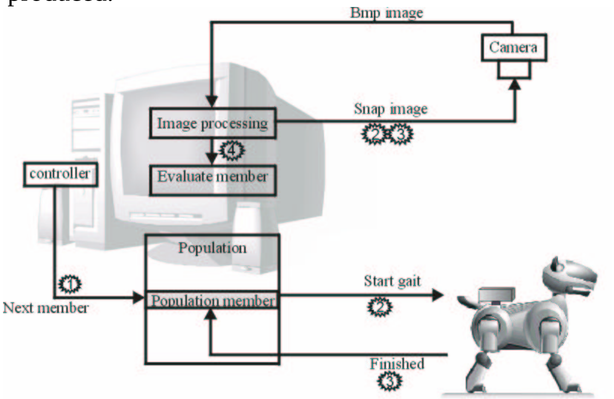


Fig. 8 Evaluation process of generation member

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.

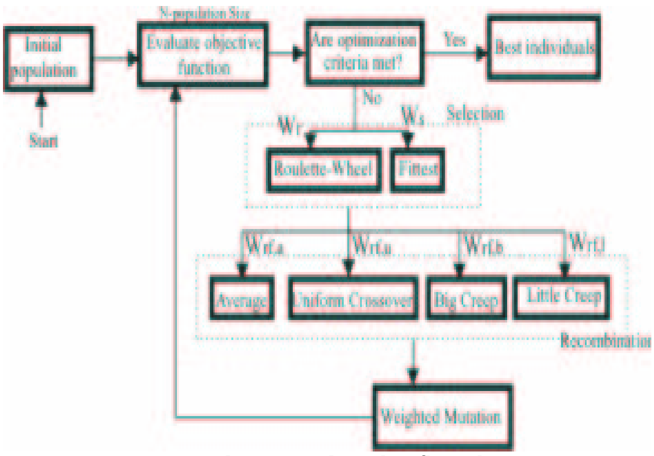


Fig. 9 Evolutionary algorithm flow chart

A hybrid approach that adapts the probability of genetic operators described above is adopted in reproduction based on the performance of the operator's offspring. The first intuition underlying the adaptation mechanism is that it 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.

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 (6) and (7).

$$W_{ij} = W_{ij}' + \frac{1}{2}\left(\Delta F_{ij} - \frac{\sum_{m\neq i}^{s}\sum_{n\neq j}^{r}\Delta F_{mn}}{s*r-1}\right) + \frac{1}{2}\left(\Delta M_{ij} - \frac{\sum_{m\neq i}^{s}\sum_{n\neq j}^{r}\Delta M_{mn}}{s*r-1}\right) \quad (6)$$

$$\sum_{m\neq i}^{s}\sum_{n\neq j}^{r}\left(W_{mn} = W_{mn}' - \frac{W_{ij} - W_{ij}'}{s*r-1}\right) \quad (7)$$

- $W_{i,j}$–Weight of i-th selection and j-th recombination operator
- $W_{i,j}$'–weight of i-th selection and j-th recombination operator from the previous generation
- ΔF_{ij}–Average fitness of i-th selection and j-th recombination operator
- s – number of selection operators (2)
- r – number of recombination operators (4)
- $M_{i,j}$ - fitness of the best member produced with i-th selection and j-th recombination operator

5 Multi-Objective Optimization

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. Thus, the complexity of the problem increases as the number of objectives increases because the objectives considered are often contradictory to one another (for example speed and stability). Such complex optimization problems have a lot of feasible solutions. However, only a few solutions among them are desirable.

Multi-objective evolutionary algorithms usually try to find all the non-dominated solutions of an optimization problem with multiple objectives. Let us consider the following multi-objective optimization problem with 4 objectives: Maximize ($f_1(x)$, $f_2(x)$, $f_3(x)$, $f_4(x)$) where x is a vector of 14 genes we use to create gait and $f_1(x)$, $f_2(x)$, $f_3(x)$, $f_4(x)$ are fitness functions to be maximized (forward-backward walk, side walk, rotation walk and stability). Separation of these walk types is reasonable because of different phasing used in its implementation. If a feasible solution is not dominated by any other feasible solutions of the multi-objective optimization problem, that solution is said to be a non-dominated solution. When the following inequalities hold between two solutions x and y, it is said that the solution x is dominated by the solution y:

$$\forall i:\ f_i(x) <= f_i(y) \ \text{ and } \ \exists j:\ f_j(x) <= f_j(y)$$

Example of non-dominated solutions for two-objective optimization problem is shown in Fig. 10. Multi-objective optimization problems usually have several dominated solutions (filled circles).

After entire population has been tested and partial fitness functions evaluated we pre-select only non-dominated solutions and discard dominated-members. Evolution operators for selection, recombination and mutation are then applied only on non-dominated members.

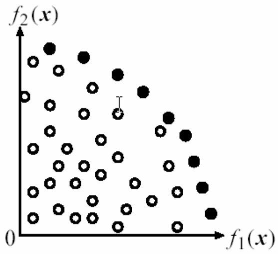


Fig. 10 Non-dominated solutions (closed circles)

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. In order to transform the values of partial fitness functions to the single fitness value of a population member we combine weighted values of partial fitness functions as shown in the following equation:

$$f(x) = w_1f_1(x) + w_2f_2(x) + w_3f_3(x) + w_4f_4(x) \tag{8}$$

where f(x) is the fitness function of x, and w1,w2,w3,w4, are non-negative weights for the partial fatnesses. These weights satisfy the following relations:

$$w_i >= 0 \text{ for } I=1,2,3,4 \text{ ,} \quad w_1+w_2+w_3+w_4=1 \tag{9}$$

Weightings for the forward-backward motion, side-walk and rotation motion are proportional to the frequency of its usage during on-line training, while weighting w_4 for the stability fitness has constant value.

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 (10) and (11) below. The minimisation of the variance and maximisation of the achieved speed are targeted.

$$EX = \sum_{i=1}^{S} X_i \Big/ S \quad , \quad EY = \sum_{i=1}^{S} Y_i \Big/ S \quad , \quad EZ = \sum_{i=1}^{S} Z_i \Big/ S \tag{10}$$

$$G = \sqrt{[\sum_{i=1}^{s}(X_i - EX)^2 + \sum_{i=1}^{s}(Y_i - EY)^2 + \sum_{i=1}^{s}(Z_i - EZ)^2] \Big/ s} \tag{11}$$

where s is the number of samples from the gyro.

AIBO robot's gyro sensor returns 3 double values ranging from -9 to 9. During execution 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 because the robot should track the ball 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.

6 Experimental Results

The evaluation takes place inside of AIBO robot's football pitch. In this experiment we used Essex Rovers football playing code during evaluation. Each generation member produces a gait that runs for approximately 15 seconds. Time necessary to finish one evaluation varies because it is necessary the robot finish at least several steps before evaluation takes place. 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 each time the robot completes a step and notifies about its movement PC controller. On the other hand robot is supplying to the controller information about type of its movement.

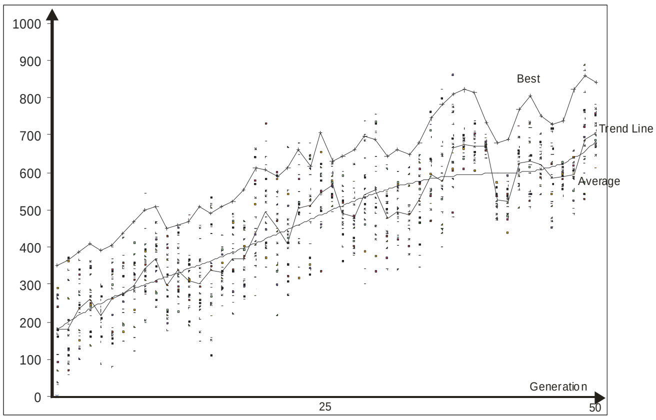


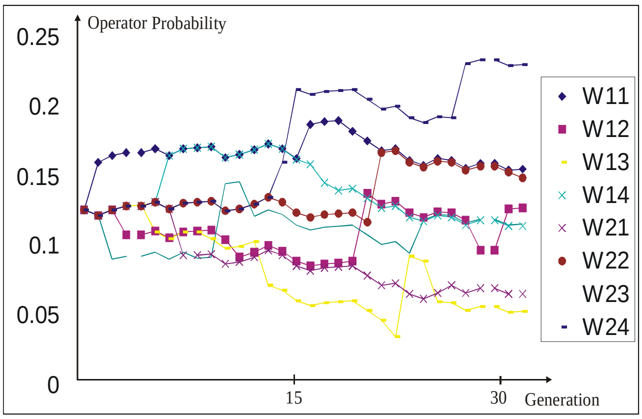
Fig. 11 The results of fitness during gait generation based on GA

When we used a population size of 20 and run it for 30 generations, most of the population members didn't perform very well at the beginning. 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.

At first 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 of the robot.

It is interesting to see changes in the operator probabilities during this experiment. All operator probabilities have the same value at the beginning, but during the experiment execution more successful ones in improving average fitness tend to gain greater chance of being used.

From these results (Fig 12) we can see that the best results are obtained from fittest-selection and Average recombination operators that became dominant especially towards the end of the experiment. We could have expect that because averaging genes can't produce very diverse population and population similar to parents used to have good average fitness and is useful for search of the local maxim but it is not very useful for exploring whole search space. Anyway, primary role of mutation is to maintain a diverse population in order to prevent premature convergence and achieve a well-distributed, wide spread trade-off front.



i	Roulette-Wheel		Sel. of the fittest	
Index value	1		2	
j	Small Creep	Big Creep	Uniform Crossover	Average
Index value	1	2	3	4

Fig. 12 Operator Probabilities

$W_{i,j}$ – presents selection probability of the i-th selection and j-th recombination operator

The primary goal of creating variable operator probabilities was to put forward those operators that produce better average fitness. To determine the extent we have done it we performed another set of experiments.

From the previous experiment we took the seeds at 10th, 20th and 30th generation and then applied the same experiment but now only using one pair of operators at a time for the period of one generation of 30 members. After determining fitness values $f_{i,j}$ for every pair of operators we recalculated desirable weights of those operators $WD_{i,j}$. By desirable I mean that those operators that have greater fitness value have also proportionally greater weights but still maintaining sum of all weights to be 1. The results of the experiments are shown in the table 3.

Seed from generation 10								
Experi -ments	W_{11}	W_{12}	W_{13}	W_{14}	W_{21}	W_{22}	W_{23}	W_{24}
1st	168	94	108	168	93	139	91	139
2nd	182	112	87	159	99	132	98	131
Seed from generation 20								
1st	171	134	55	139	77	125	89	210
2nd	193	122	67	129	088	112	98	191
Seed from generation 30								
1st	156	128	49	119	54	154	117	223
2nd	171	125	55	119	64	122	105	239

Tab. 3 Seeds from different generations

Two important conclusions can be derived from this experiment. Some recombination operators contribute more than others to the average fitness of a population and contribution of operator probabilities changes over time. Better results in terms of increasing the speed of evolution can be accomplished by creating variable recombination and mutation operator probabilities.

7 Conclusion

This paper presents a hybrid system for Sony robots to 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 real world. The mutating and combination behaviors of the Genetic Algorithms allow the process to develop to a useful Behaviour over time. 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.

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. It is tailored to a steady state reproduction scheme. It would not be literally applicable to problems with noisy evaluation functions.

References

[1] R. Beer, J. Chiel, and L. Sterling. Evolution of plastic neurocontrollers for situated agents. An artificial insect, American Scientist, 79, 444-452, 1989.

[2] H. Cruse, C. Bartling, J. Dean, M. Dreifert, T. Kindermann, and J. Schmitz. Walking: a complex behavior controlled by simple networks. Adaptive Behavior, 3, 385-418, 1995.

[3] D. Floreano and F. Mondada., Genetic evolution of a neural-network driven robot. Proc. of the 3rd International Conference on Simulation of Adaptive Behavior, MIT Press, 421-430, 1994.

[4] S. Nolfi. Evolving non-trivial behaviors on real robots: A garbage collecting robot. Robotics and Autonomous System, 22, 187-198, 1997.

[5] G. Taga, Y. Yamaguchi, and H. Shimizu. Selforganized control of bipedal locomotion by neural oscillators in unpredictable environment. Biological Cybernetics, 65, 147-159, 1991.

[6] 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.

[7] D. Golubovic and H. Hu, An interactive software environment for gait generation and control design of Sony legged robots, Proceedings of International Symposium on RoboCup, Fukuoka, Japan, June 2002.

[8] M. Hardt and O. von Stryk: Towards Optimal Hybrid Control Solutions for Gait Patterns of a Quadruped. CLAWAR: International Conference on Climbing and Walking Robots (2000) 385–392

[9] K. DeJong, “An Analysis of the Behavior of a Class of Genetic Adaptive Systems”, PhD Dissertation, University of Michigan, 1975.

[10] J. Grefenstette, “Optimization of Control Parameters for Genetic Algorithms”, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-16, No. 1, January/Febuary, 1986.