Evolving Motion Control for a Modular Robot

Sunil Pranit Lal, Koji Yamada, Satoshi Endo

Complex System Laboratory, Department of Information Engineering, Faculty of Engineering, University of the Ryukyus, 1 Senbaru, Nishihara, Okinawa 903-0213, Japan

sunil@eva.ie.u-ryukyu.ac.jp, {koji, endo}@ie.u-ryukyu.ac.jp

Abstract

This paper documents our ongoing efforts in devising efficient strategies in motion control of the brittle star-typed robot. As part of the control framework, each robotic leg consisting of series of homogenous modules is modeled as a neural network. The modules representative of neurons are interconnected via synaptic weights. The principle operation of the module involves summing the weighted input stimulus and using a sinusoidal activation function to determine the next phase angle. Motion is achieved by propagating phase information from the modules closest to the main body to the remainder of the modules in the leg via the synaptic weights. Genetic algorithm was used to evolve near optimal control parameters. Simulations results indicate that the current neural network inspired control model produces better motion characteristics than the previous cellular automata-based control model as well as addresses other issues such as fault tolerance.

1 Introduction

Problem solving utilizing techniques harnessed from nature has been and continues to be a niche of computational intelligence field. Classical contributions in this respect include artificial neural networks (ANN), genetic algorithm (GA), ant colony optimization (ACO) and artificial immune system (AIS).

In this paper we draw from the strengths of genetic algorithm and neural network to devise efficient strategies for motion control of a modular robot. Developed by John Holland in early 1970s [1] genetic algorithm is a search technique inspired by biological adaptations used extensively to solve optimization related problems [2]. The algorithm involves representation of candidate solutions to a problem using chromosomes also known as individuals. The initial randomly generated population of individuals is successively transformed based on their fitness by applying genetic operators such as selection, crossover and mutation. Based on the survival of the fittest, it is anticipated that with each passing generation the fitness of the individuals improves thus providing near optimum solution to the problem at hand.

Inspired by the information processing ability of biological neurons, the modern day field of artificial neural networks has its origins in 1943 with pioneering work by McCulloch and Pitts [3] who developed computational model of a neuron. ANN has been applied in multitude of areas namely pattern classification, function approximation, forecasting, optimization and control. While there are numerous neural network architectures in existence the fundamental computational model of the neuron essentially remains the same. In a typical ANN model the neurons are inter connected via synaptic weights. The neuron interacts by transmitting signals to other neurons connected to its output depending on the weighted sum of input stimulus to the neuron and the activation function. Learning takes place by adjusting the weights such that given an input, the output of the ANN is as close as possible to the desired output. Interested readers are directed to [4] as good introductory reference material in neural networks.

In the literature many notable contributions have been made in the field of robotics and control leveraging on GA and ANN. Reil and Husbands [5] successfully simulated bipedal straight-line walking using recurrent neural network whose parameters were evolved by GA. Porting genetically evolved neural network controller for a hexapod robot from simulation model to actual hardware was demonstrated by Gallagher et al. [6]. One of the conclusions reached by them was that neural network controller performed extremely well in real world in spite of the fact that inertia, noise and delays were not taken into account in the simulation. Hickey et al. [7] developed a system called *creeper* featuring neural network controller for producing realistic animations of walking figures. The weights for the neural network were evolved using GA wherein the fitness of a chromosome encoding the weights was related to its performance in controlling the simulated walking figure. Application of neural network is not confined to controlling just single robotic agents as demonstrated by Lee [8] in controlling behaviour of multiagent system of simulated robots in a predator and prey type environment. Similar to other research work described above, GA was used to evolve weights for the neural network behaviour controller.

Moving on, the focus of this paper is the motion control of the modular robot developed by our laboratory inspired by the characteristics of a brittle star (*Ophiuroidea*) for search and rescue operation. The modular architecture enables decentralized control and ease of repair to the robot. Initially the motion of the robot was achieved through coordinated movement of the modules via trial and error process [9]. Later genetic algorithm was used to better this process [10]. Most recently cellular automata (CA) based motion control architecture was developed [11]. Though the approaches adopted in the past produced desired motion characteristics in the robot, the notion of recovering from module failure remained elusive. To this effect, in this paper we explore neural inspired control whereby the modules are modeled as neurons connected as part of a network within each leg. GA is used to evolve near optimal control parameters, which include initial state matrix and weight matrix.

The paper is divided as follows: Section 2 provides background information on the brittle star robot as well as the CA-based motion control framework. Section 3, which models the motion of the legs of the robot using characteristics of simple harmonic motion forms the prelude to section 4 on neural inspired motion control.

Simulations using Open Dynamics Engine (ODE) [12] is used to verify the effectiveness of the proposed control model in dealing with module failures in section 5. Finally, section 6 concludes this paper with directions for future research.

2 Background on the Brittle Star Robot

The brittle star robot (Fig. 1a) considered in this paper has a modular architecture consisting of modules as shown in Fig. 1b. Each module incorporates an onboard micro controller (BASIC Stamp 2sx), actuator (RC Servo Futaba S5301) and two touch sensors. In its current setup the actual robot hardware has five legs with six modules per leg. Modules of one degree of freedom are connected alternately in horizontal and vertical orientation. In this way, a set of adjacent modules has two degree of freedom joint.

a) b)

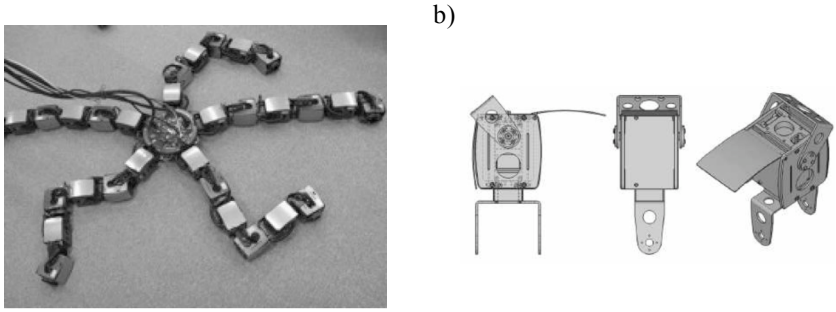


Fig. 1. a) The brittle star robot b) Individual module connected to make up the leg.

In our previous effort [11] the modular structure of the robot was modeled as two-dimensional cellular automata (CA) lattice with the individual modules of the robot represented as cells in the lattice. Differential transition rule for updating cell states in the lattice depending on its position in the leg (control rule for the lead modules closest to the central disc and leg rule for modules on other part of the leg) were discovered using co-evolutionary algorithm. The performance of the best control and leg rule combination evolved in terms of distance travelled by the simulated robot had a fitness measure of 15. Since the simulation environment remains unchanged this performance measure can be easily compared with the models described in this paper.

While CA-based control architecture produced satisfactory motion some issues remained outstanding. In particular due to the inherent nature of cellular automata, the rules obtained were tightly coupled with the initial state of the lattice configuration. In other words if the initial phase angles of the modules is changed then the resulting motion becomes incoherent. Moreover the shear size (2^{1536}) of the search space of possible transition rules made the task of learning computationally intensive. The model developed in the following sections tries to address these issues.

3 Simple Harmonic Motion Model

Observations of the brittle star reveal rowing or snake-like movement of legs to achieve locomotion, which is modeled in this section as simple harmonic motion (SHM). In this approach we begin by dividing the phase of the modules into 16 equal divisions ($N_d = 16$) as shown in Fig. 2.

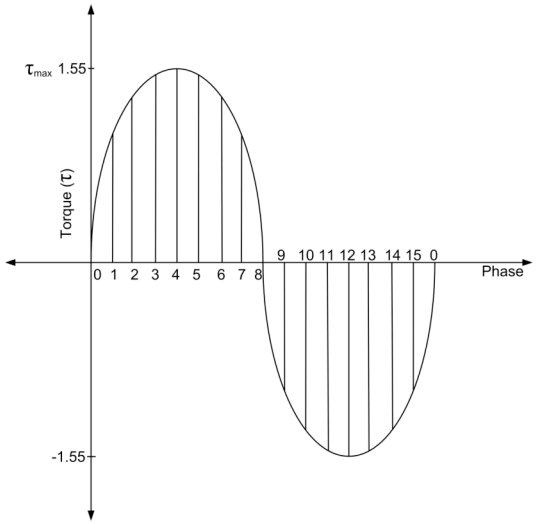


Fig. 2. Relationship between the torque required to rotate a module at a given phase was approximated using sinusoidal function.

The state transition (Fig. 3) is worked out using the following rules.

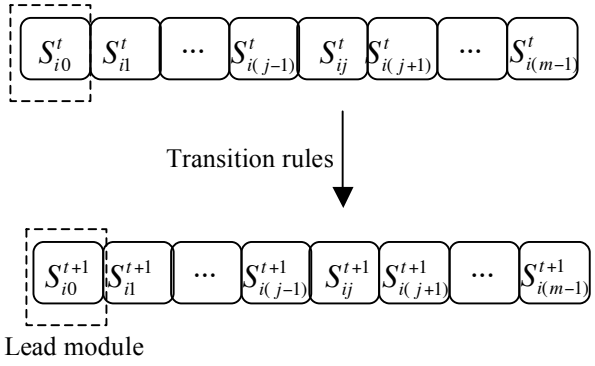


Fig. 3. The state of m modules in i^{th} leg at time t is transformed to the next state at time t+1 using the transition rule.

For the lead modules (j=0)

The lead module executes sine-like movement in time starting from some initial state. The initial state at time t = 0 is given by

$$S_{i0}^{0} = I_i \qquad where \quad 0 \le i < 5,\ i \in \mathbf{Z} \qquad 0 \le I_i < N_d,\ I_i \in \mathbf{Z} \tag{1}$$

The next state is determined by

$$S_{i0}^{t+1} = (S_{i0}^{t} + 1) \ \ mod \ \ N_d \tag{2}$$

For the remaining modules (j≠0)

The initial states of these modules do not matter. The next state of a module is worked out based on the state of the module connected directly in front of it as follows:

$$S_{ij}^{t+1} = (S_{i(j-1)}^{t+1} + 1) \ \ mod \ \ N_d \tag{3}$$

The state changes are propagated sequentially from the lead module towards the tail modules. Once the state transition concludes for all the modules at any given time step, the actuators corresponding to the modules are applied torque of magnitude related to the state of the module as follows

$$\tau_{ij}^{t} = \tau_{\max} sin\left(\frac{2\pi S_{ij}^{t}}{N_d - 1}\right) \tag{4}$$

3.1 Genetic Encoding

The initial state of the lead modules was discovered through evolutionary means utilizing genetic algorithm. With 16 states per module, 4 bits are required for equivalent binary representation (Fig. 4). The choice of having 16 states was to achieve required level of granularity in control while minimizing the number of bits. It is worth mentioning that search space is considerably small (2^{20}) compared to the CA-based approach mentioned in the previous section.

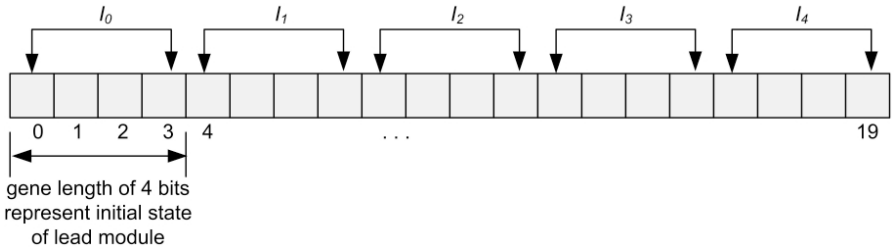


Fig. 4. Genetic encoding of the initial states of the lead modules in the five legs

3.2 Fitness Function

The fitness of each chromosome is evaluated by first decoding the initial state of the lead modules it represents. The lead modules are initialized and the state transition model described above is applied to the simulated model of the brittle star robot for successive iterations (SIM_STEPS) thereby transforming it from initial position, (x_i,y_i) to the final position, (x_f,y_f). Since the focus of this paper is on forward locomotion of the robot, the fitness of the chromosome is thus proportional to the Euclidean distance covered by the robot.

$$F = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2} \tag{5}$$

3.3 Genetic Operators

Based on the fitness of the chromosomes in the population, GA operations; namely selection, crossover and mutation are applied to whole population. Firstly selection was performed using roulette wheel selection method, which offered fitter individuals better chance of mating. The best individual in a generation is automatically carried over to the next generation as per the elite selection scheme, which ensures good solutions discovered are retained. The selected pairs of chromosomes are crossed over using one-point crossover at randomly chosen locus with a crossover probability of P_C. Finally, mutation operation involving bit flips is applied with a probability of P_M to individual genes in chromosomes after the selection and crossover operations.

3.4 Simulation Results

The simulation was carried out over numerous trials using parameters shown in Table 1. In each trial, initial population of chromosomes of size (POP_SIZE) was randomly generated and GA was executed for a number of generations (MAX_GEN). For each generation, the fitness of all the chromosomes in the population is evaluated after which genetic operators are applied to the population to create the next generation

Table 1. Summary of simulation parameters

Parameter	Value
P_C	0.85
P_M	0.15
POP_SIZE	25
SIM_STEPS	1500
MAX_GEN	200

The fitness of control parameters evolved across the generations is captured in Fig. 5. From the observations of the simulated robot in ODE environment it became apparent that the collective SHM motion of the legs were hardly translating into any linear locomotion thus the poor fitness measure.

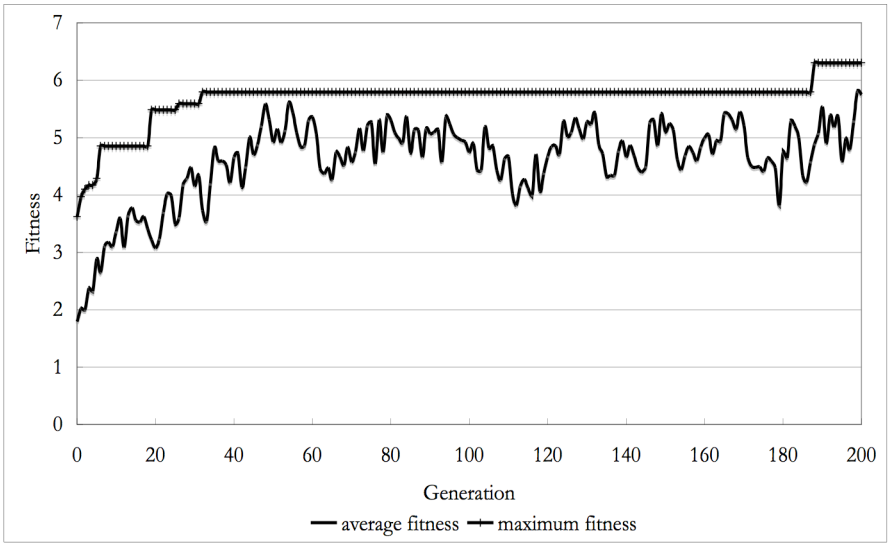


Fig. 5. Graph of average fitness and maximum fitness of population against generation for the SHM model.

While the notion of inducing motion by propagating phase information from the lead modules to all the modules in the legs is interesting, it appears that the interaction between the modules in a leg to cause desired motion is too complex to be modeled as sinusoidal wave function.

4 Neural Inspired Motion Control

Neural networks are good at dealing with system parameters whose relationships are not easily deducible. Inspired by this, we decided to model the interaction between the modules using the principles of ANN. It is worth mentioning that the functionality of the model we developed though similar to conventional ANN has subtle differences that are explained below.

Each of the leg is modeled as a fully connected neural network (Fig. 6) with the modules represented as neurons. The modules maintain state information about its current phase angle. Furthermore the modules are interconnected via binary weights to model inhibitory and excitatory stimulus between them. Formally, $w_{ab} \in \{0,1\}$, where w_{ab} represents weight between the connection from module a to b.

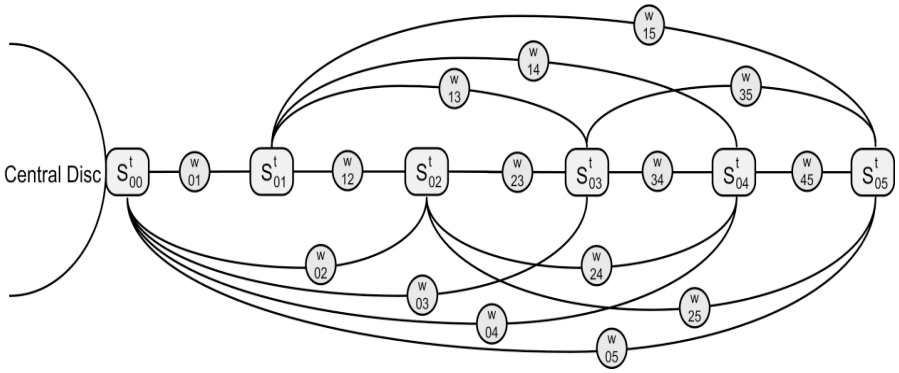


Fig. 6. Conceptual framework for the neural inspired motion control architecture.

Synchronized propagation of phase information through the network is used to update the current state of the modules. Consistent with § 3 only the lead module has an initial state which is updated in discrete time steps according to equations (1) and (2). Once the state of the lead module has been updated, the torque to be applied is computed using equation (4), which is then propagated to the rest of the network. The states of the remaining modules are then updated sequentially such that the module directly next to the lead module is updated first followed by the module next to it and so on.

The state of any given non-lead module in the leg is updated as follows. First the input stimulus from the modules closest to the central disc before it is summed.

$$X_{ij}^{t+1} = \sum_{k=0}^{j-1} S_{ik}^{t+1} w_{kj} \tag{6}$$

Given the rotational nature of the modules, sinusoidal activation function (7) is applied to the summed input to yield the next state, which is then propagated to the rest of the network.

$$S_{ij}^{t+1} = \tau_{\max} sin(X_{ij}^{t+1}) \quad (7)$$

To put it intuitively, equations (6) and (7) allow a module to undergo valid state transition using latest state information of modules which underwent state transition just prior to it. In summary the state transition of the modules occurs sequentially within a discrete time step.

4.1 Evolving Suitable Control Parameters

For the most part, GA framework is reused from § 3 in determining near optimal initial states of the lead modules and the weight matrix for each of the legs. The only change to the GA framework required is encoding of the chromosome (Fig. 7) and its subsequent fitness evaluation.

Compared to § 3 the length of the chromosome has significantly increased as it encodes 4 bits of initial state and 15 bits of weight matrix per leg. While real number could have been used for the weight matrix, in the interest of keeping search space manageable we decided just to use binary weights. Notably the search space (2^{95}) though significant is manageable in comparison with the search space for the CA-based model.

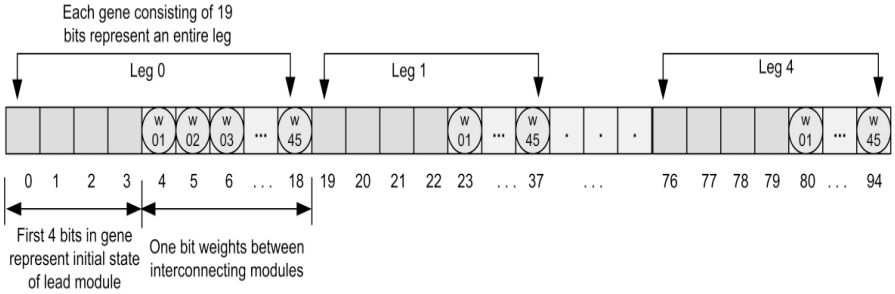


Fig. 7. Genetic encoding of initial state and weight matrix.

4.2 Simulation Results

Using the same procedures and parameters described in § 3 the simulations were carried out using the revised model. The results obtained (Fig. 8) indicated that the performance of the robot improved markedly. This is a significant step as it surpasses all the previous models we developed. Observations (Fig. 9) using the control structure encoded in the best chromosome discovered in controlling the movement of the robot showed far greater coherence and fluidity than any of our other models.

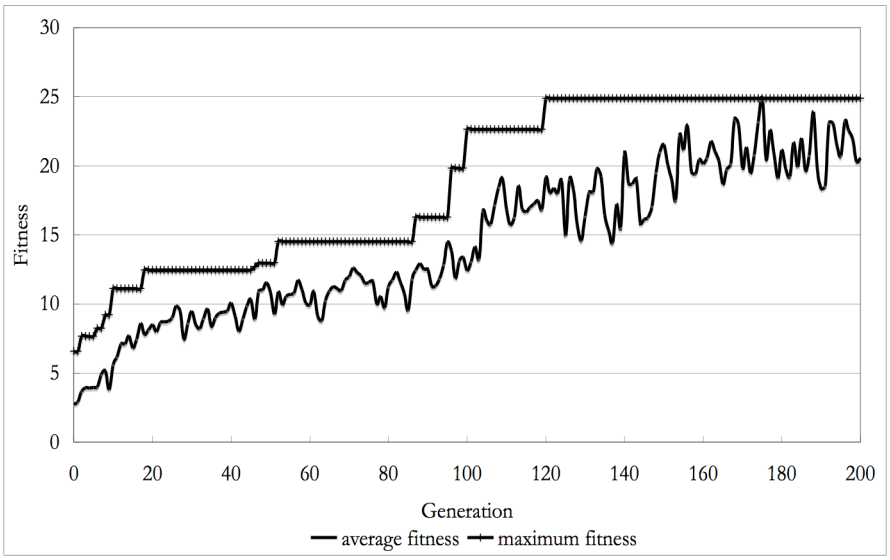
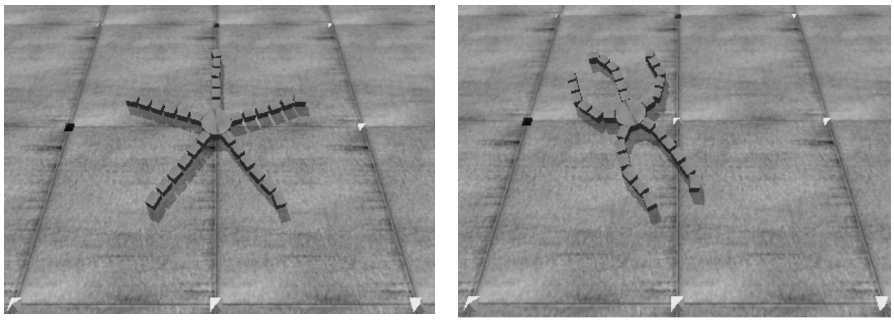


Fig. 8. Graph of average fitness and maximum fitness of population against generation for the neural inspired control model.



(a) (b)

Fig. 9. Snapshot of the simulated robot in ODE environment at a) the start of simulation b) the end of simulation.

5 Experimentation

In this section the proposed neural network based model is analyzed in terms of robustness of the control system in dealing with failure of modules and also the extrapolation of the obtained results beyond the simulation time steps.

5.1 Fault Tolerance

Evolving control parameters using GA is no doubt a time consuming process. Thus it would be highly desirable to have a robust control architecture, which can deal with module failures without having to be retrained.

In evaluating the robustness of the proposed model, module failure was simulated by configuring the module to be unresponsive to input stimulus thereby maintaining a fixed state and providing zero output to the rest of the connected modules in the network. Simulation of the robot with a set of failed modules was carried out for 2000 time steps. The distance is then measured and used to calculate the degree of mobility (M_D) defined simply as

$$M_D = \frac{distance\ \ traveled\ \ by\ \ robot\ \ with\ \ set\ \ of\ \ failed\ \ modules}{distance\ \ traveled\ \ by\ \ robot\ \ without\ \ failed\ \ modules} \times 100\% \qquad (8)$$

5.1.1 Single Module Failure

Failure was induced one by one in all the modules and the corresponding degree of mobility of the robot is depicted in Fig. 10.

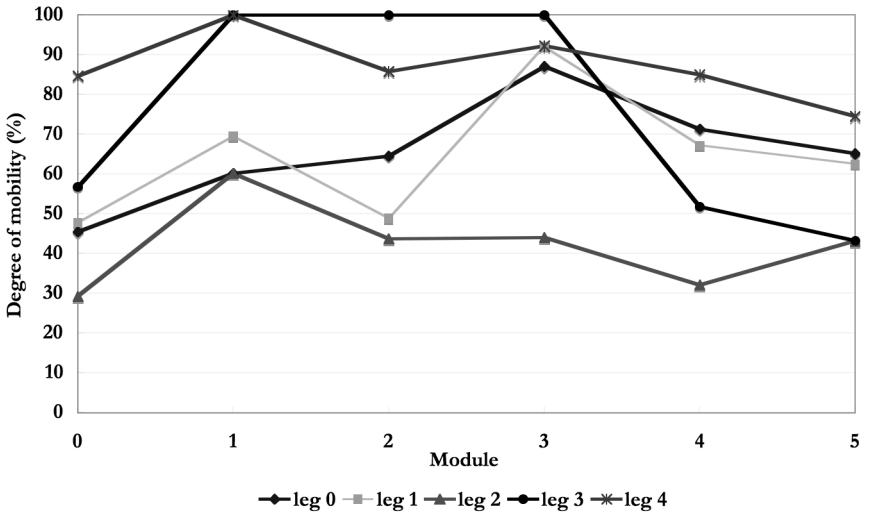


Fig. 10. Graph of degree of mobility against single module failures for all the legs

From the results it is apparent that the degree of mobility is greatly influenced by the position of the failed module. For instance, failure of module 2 in leg 2 causes greater performance degradation in comparison with failure of module 2 in leg 3.

Needless to say the lead module is an important part of the control system as it provides the initial state information, and also the state transition of other modules is synchronized with the propagation of phase information from the lead module. Thus we expected the failure of lead module to be the major contributor in hindering the overall mobility of the robot. However the results indicate the emergence of distributed control, where by failure of the last module in a leg can be equally if not more devastating than the failure of the lead module.

Finally, in the single module failure scenario 4/30 (13.3%) of the modules can fail without compromising the mobility of the robot at all.

5.1.2 Multiple Module Failure

Following on from the previous subsection, selected scenarios of multiple module failure was simulated and is presented in Table 2.

It is worth noting that the degree of mobility is not just depended on the number of failed modules but also the position of these modules. Moreover the modules, which adversely affected the degree of mobility in the single module failure scenario, were also the greatest contributors to performance degradation in multiple module failure scenarios.

Table 2. Influence of failure of the j^{th} module of the i^{th} leg on the degree of mobility. $*$ indicates all modules in the particular leg.

Failed module (i, j)	Distance traveled (cm)	Degree of mobility, M_d (%)
(3,1), (4,1)	31.48	100.0
(1,3), (4,2)	24.58	78.1
(1,0), (3,3)	15.03	47.7
(0,0), (2,0)	6.68	21.2
(1,3), (3,2), (4,1)	29.00	92.1
(0,3), (1,3), (3,3)	25.84	82.1
(1,0), (3,3), (4,5)	13.97	44.4
(0,0), (1,2), (2,0)	3.23	10.3
$(4,*)$	26.64	84.6
$(2,*)$	9.21	29.3
$(3,*), (4,*)$	12.05	38.3
$(2,*), (4,*)$	8.62	27.4

5.2 Extrapolation Beyond Simulation Time Steps

In evaluating the fitness of a chromosome, the control parameters represented by the chromosome was applied to the simulated robot for duration of 1500 simulation time steps. We are interested in knowing whether the results obtained can be extrapolated beyond this duration. Thus the motion of the robot using best control parameters discovered by GA was simulated beyond SIM_STEPS.

As shown in Fig. 11 the distribution is fairly linear. This is certainly an improvement with respect to the CA-based control architecture where most of the individuals evolved by GA exhibited cyclic behaviour after exceeding SIM_STEPS.

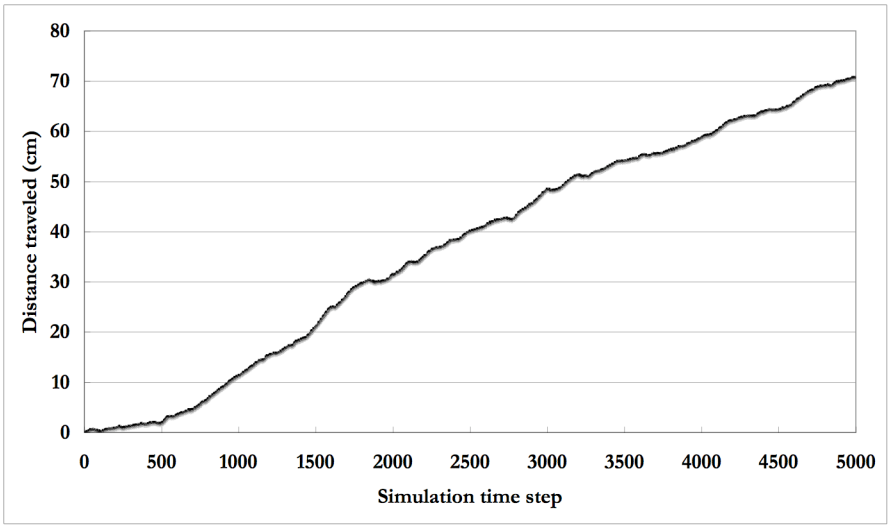


Fig. 11. Graph of distance traveled against simulation time step using the best-evolved control parameters.

6 Conclusion

The process of evolution by natural selection enable search for best solutions by adapting to the problem domain as time passes by. While we have utilized GA to find suitable parameters for our control models it is worth mentioning that from a global viewpoint the models in turn have evolved from trial and error means of motion control, to CA-based control architecture to simple harmonic motion model, which lead to the neural inspired motion control model presented in this paper.

The experimental results indicate that the current neural network based model outperforms the previous CA-based model. In designing the current model the search space of the control parameter has been reduced. More importantly the proposed model has shown some degree of resilience in overcoming scenarios involving failure of modules. For the single module failure scenario, 13.3% of the

modules can fail without at all affecting the overall mobility of the robot. Furthermore, degree of mobility has been noted to depend on the position of the failed module.

While the focus of this paper was to achieve motion control in a fixed direction, it is worth highlighting that since the robot is modular in nature, swapping control parameters between legs enables to change the direction of motion.

In concluding remarks, Gallagher et al. [6] hexapod robot performed better than their simulated robot by not falling down when faced with sensor failures even though sensor failure was not explicitly included in the training of the neural network controller. Comparatively, the brittle star robot is not highly robust, which implies there is a lot of room for future improvements to the current motion control model. The learning process can benefit by utilizing feedback from backpropagation algorithm in adjusting the weights. Furthermore, extending the scope of the fitness function to encompass the degree of mobility of the robot in the presence of module failure should hopefully lead to evolution of more robust control.

References

1. Holland J. H. Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, 1975
2. Goldberg D. E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, Massachusetts, 1989
3. McCulloch W. S., Pitts W. A logical calculus of ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 1943;5:115-133
4. Jain A. K., Mao J., Mohiuddin K. Artificial Neural Networks: A Tutorial. IEEE Computer 1996;29(3):31-44
5. Reil T., Husbands P. Evolution of central pattern generators for bipedal walking in a real-time physics environment. IEEE Transactions on Evolutionary Computation 2002;6(2):159-168
6. Gallagher J. C., Beer R. D., Espenschied K. S., Quinn R.D. Application of evolved locomotion controllers to a hexapod robot. Robotics and Autonomous Systems 1996;19:95-103
7. Hickey C., Jacob C., Wyvill B. Evolution of a Neural Network for Gait Animation. In: Leung H. (ed) Proceedings of Artificial Intelligence and Soft Computing. ACTA Press, Calgary, 2002;357-190
8. Lee M., Evolution of behaviors in autonomous robot using artificial neural network and genetic algorithm. Information Sciences 2003;155:43-60
9. Takashi M. Studies on Forward Motion of Modular Robot. MSc Dissertation, University of Ryukyu, Japan, 2005
10. Futenma N., Yamada K., Endo S., Miyamoto T. Acquisition of Forward Locomotion in Modular Robot. In Dagli C. H., Buczak A. L., Enke D. L., Embrechts M. J., Ersoy O. (ed) Intelligent Engineering Systems through Artificial Neural Networks. ASME Press, New York, 2005;91-95
11. Lal S. P., Yamada K., Endo S. Studies on motion control of a modular robot using cellular automata. In Sattar A., Kang B. H. (ed) AI 2006: Advances in Artificial Intelligence. LNAI 4304. Springer-Verlag, Berlin Heidelberg, 2006;689-698
12. Open Dynamics Engine [Online]. http://www.ode.org/