Studies on motion control of a modular robot using cellular automata

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. In this paper we report preliminary findings of using cellular automata (CA) as an underlying architecture in controlling the motion of a five-legged brittle star typed robot. Three control models were incrementally designed making use of genetic algorithm (GA) as well as co-evolutionary algorithm in finding appropriate rules for automaton. Simulations using Open Dynamics Engine (ODE) was used to verify the rules obtained for each of the models. The indications from the results are promising in support for CA as feasible means for motion control.

Keywords: modular robot, control system, cellular automata, and evolutionary algorithm

1 Introduction

Inspired by the characteristics of a brittle star (*Ophiuroidea*), our laboratory developed the brittle star robot (Fig. 1a) consisting of independent modules (Fig. 1b) with each module incorporating 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. Robot of such nature can be useful in the search and rescue operations; for instance during earthquakes the robot can be deployed to seek for survivors trapped under collapsed buildings which would otherwise be hazardous for human rescuers to reach.

Initially the motion of the robot was achieved through coordinated movement of the modules via trial and error process [1]. Later genetic algorithm was used to better the process of finding near optimal motion pattern [2]. Though the approaches adopted in the past produced desired motion characteristics in the robot, the underlying concern was that should there be a failure of some modules the overall mobility of the robot would get compromised. To this effect, we have embarked on exploring strategies for evolving emergent global behavior of the robot, such as walking in a straight line, from the combined action of all the functioning modules in the a given configuration. As an initial step, in this research we consider using cellular automata as an underlying architecture for controlling the forward locomotion of the brittle star robot.

a) b)

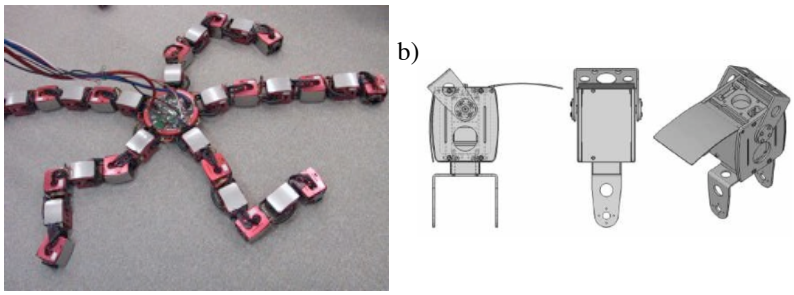

Fig. 1. a) The brittle star robot b) Individual module connected to make up the leg

Cellular automata [3] consist of a lattice of identical finite state machines (cells) whose state changes is governed by some common transition rule. The next state of a cell at time, t+1 is determined from the current state of the cell and its neighboring cells at time t. Even the simplest rules can result in emergence of interesting patterns over a period of time, as in the case of Conway's game of life [4] which is perhaps the best known two-dimensional cellular automaton used to model basic process in living systems.

In the field of robotics cellular automata has been used to control self-reconfigurable robots, which can autonomously change their shape to adapt to their environment [5][6]. Furthermore, Behring et. al [7] has demonstrated promising use of cellular automata as a means to perform path planning using real robot in an environment clustered with obstacles.

In our approach, we modeled the individual modules of the robot as cells in the cellular automata lattice. Based on this we incrementally developed three control models for the brittle star robot. Genetic algorithm [8] was used to evolve suitable rules for each of the models. Finally simulation using Open Dynamics Engine was used to verify the effectiveness of each of the models.

2 Singular transition rule for disjoint leg set

In our initial attempt, the robot (with $n = 5$ legs and $m = 6$ modules per leg) was modeled as a set of disconnected one-dimensional CA lattice representing the legs of the robot and with each cell representing a module. Furthermore we decided to derive a singular CA transition rule for all the modules in the robot irrespective of their connection topology. The rational behind this was that since the rule would operate on a small neighborhood of modules, should any of the modules or legs fail, the overall mobility of the robot should not get compromised to a large extent. Furthermore, a single unifying transition rule would make the modules truly modular from both hardware and software point of view thus maintenance would be a ease in terms of replacement of modules.

The motion of the actual robot involves applying electric pulses of varying levels to the individuals modules based on which the servomotors rotate at differing angles. To model each module as a cell in the CA framework, we first quantize the input to the module to certain discrete levels. Initially we experimented with 16 levels, how-

ever to reduce the search space of possible transition rules without affecting the robot's motion too much we decided on eight levels as being adequate, thus the number of states per cell, $k = 8$.

The neighborhood, η of a cell can encompass any number cells in its vicinity. The smaller the neighborhood, lesser the communication overhead per cell update. Moreover this also translates into fewer entries in the rule table. For this reason we choose the neighborhood of a cell to include only its adjacent cells, that is radius, $r = 1$ (Fig. 2) .

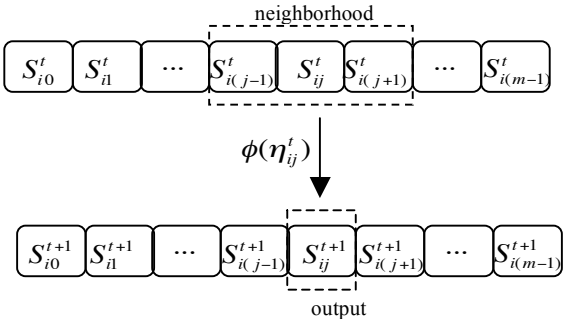

Fig. 2. Lattice configuration showing states of the modules in the i^{th} leg of the robot. Radius r =1 groups 2 cells (modules) adjacent to the current cell to form a neighborhood, η which is applied as an input to the transition rule, $\phi(\eta)$. The resulting output is used to update the state of the current cell at time, t+1.

To apply the same rule consistently on all the cells, we handle the cells on the edge of the lattice with the following boundary conditions for all $u \in \{1, 2, \ldots r\}$

$$S^t_{i(j-u)} = \begin{cases} S^t_{i(j-u+m)} & if\ j-u<0 \\ S^t_{i(j-u)} & otherwise \end{cases} \quad (1)$$

$$S^t_{i(j+u)} = \begin{cases} S^t_{i(j+u-m)} & if\ j+u \geq m \\ S^t_{i(j+u)} & otherwise \end{cases} \quad (2)$$

As an illustration, for 0^{th} leg with 6 modules and neighborhood radius of 1, the neighbors for the cell S^t_{00} are S^t_{05} and S^t_{01}. Likewise the neighbors for cell S^t_{05} are S^t_{04} and S^t_{00}.

2.1 Genetic Encoding

With 8 states per cell, 3 bits are required for equivalent binary representation. Moreover, for a neighborhood with a radius $r = 1$ the number of entries in the rule table (Table 1) is $8^{2r+1} = 512$

The rule table is encoded in a chromosome as shown in Fig. 3. The output bits are listed in lexicographical order of neighborhood. Interestingly, even though the radius is minimum possible for cell-cell interaction to take place, the rather large number of states per cell has made the search space of possible transition rules phenomenal (2^{1536}). For this reason we did not consider increasing the radius.

Table 1. A sample rule table for $(k,r) = (8,1)$

Index	Neighborhood			Output
0	000	000	000	010
1	000	000	001	100
2	000	000	010	000
⋮				
511	111	111	111	101

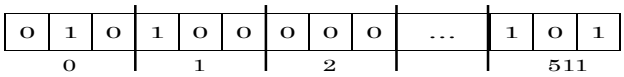

Fig. 3. Representation of the rule table in a chromosome

2.2 Fitness Function

The fittest of each chromosome is evaluated by first decoding the rule table it represents. The rule is then applied to a simulated model of the brittle star robot whose initial position, (x_i,y_i) and final position, (x_f,y_f) in the desired direction of motion is recorded. Since the focus of this paper is on forward locomotion of the robot, the fitness of the chromosome is thus proportional to the straight-line distance covered by the robot.

$$F = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2} \tag{3}$$

2.3 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 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.

2.4 Simulation Results

The simulation was carried out over numerous trials using parameters shown in Table 2. 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.

In the fitness evaluation of each chromosome, the CA lattice structure representing all legs and its constituent modules in robot is initialized randomly with a fixed seed. The transition rule encoded in the chromosome being evaluated is applied for a number of successive iterations (SIM_STEPS) to transform the cell lattice. In each iteration the object model in the ODE environment is updated accordingly. The final position of the robot at the end of SIM_STEPS iteration is used in fitness calculation above (3).

It is worth mentioning that for comparability of results, the same seed for the random number generator is used for the initialization of all the lattice configurations in all the models discussed in this paper.

Table 2. Summary of simulation parameters

Parameter	Value
P_C	0.85
P_M	0.15
POP_SIZE	25
SIM_STEPS	1500
MAX_GEN	250

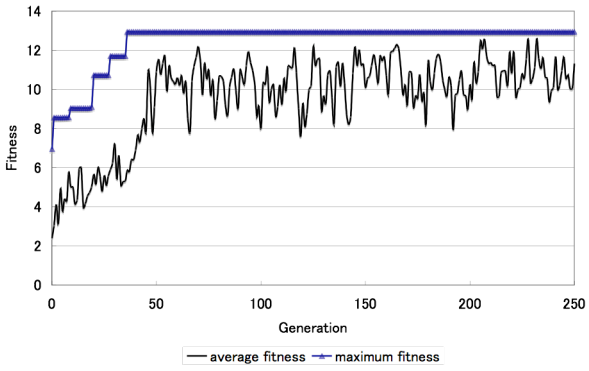

Fig. 4. Graph of average and maximum fitness of population against generation for the disconnected one-dimensional CA model of the robot.

The evolution of transition rules across generations is captured in Fig. 4. Notably, in most of the trials there was premature convergence of GA. From the observations of the motion of simulated robot in ODE it became apparent that there was lack of

coordination amongst the legs and thus the overall mobility of the robot in terms of maximizing the fitness function is affected. In summary the results of modeling the robot as set of disconnected one-dimensional cellular lattice have provided a valuable insight about the need for greater coordination between these one-dimensional structures. Thus our next step would be to connect the lattice into a two-dimensional structure.

3 Singular transition rule for connected leg set

In this section the shortcomings of the model described in the previous section is addressed. In particular the set of one-dimensional disconnected CA lattice representing the legs are connected via modules closest to the central disc of the robot (Fig. 5), thus forming a two-dimensional structure. The notion of single transition rule for all the modules is adopted for reasons described in § 2. Furthermore the radius describing the neighborhood of a cell is set to 1 (same as § 2).

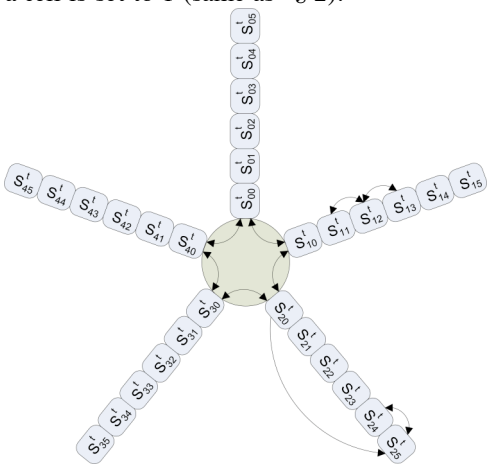

Fig. 5. The two-dimensional CA lattice. The arrows represent the interaction between a cell (possibly located somewhere in the middle or on the edge of the lattice) and its neighboring cells. Essentially the interaction is similar to the model in § 2 except modules closest to central disc interact with modules in similar position located on the neighboring leg.

The two neighbors ("right", R and "left", L) for a cell at position (*i*,*j*) in the *n* x *m* lattice is given as follows:

$$R = \begin{cases} S^t_{(i+1)j} & \text{if } j = 0 \text{ and } i+1 < n \\ S^t_{(i+1-n)j} & \text{if } j = 0 \text{ and } i+1 \geq n \\ S^t_{i(j+1-m)} & \text{if } j > 0 \text{ and } j+1 \geq m \\ S^t_{i(j+1)} & \text{otherwise} \end{cases} \quad (4)$$

$$L = \begin{cases} S^{t}_{(i-1)j} & \text{if } j = 0 \text{ and } i-1 \geq 0 \\ S^{t}_{(i-1+n)j} & \text{if } j = 0 \text{ and } i-1 < 0 \\ S^{t}_{i(j-1)} & \text{otherwise} \end{cases} \quad (5)$$

3.1 Simulation Results

Using the same procedures and parameters described in § 2, the simulations were carried out using the revised model. The results obtained (Fig. 6) indicated that performance of the robot worsened compared to the previous model, which used disconnected one-dimensional lattice. Closer examination of the motion pattern of the simulated robot revealed that the current two-dimensional CA lattice does introduce a degree of coordination between the legs. However since all the legs are following the same rule, eventually all the legs converge to the same formation (Fig. 7) thus being counterproductive. While the motivation behind using a singular transition rule for all the modules was to reinforce modularity of the robot, the results on the other hand provide strong justification for using differential transition rule.

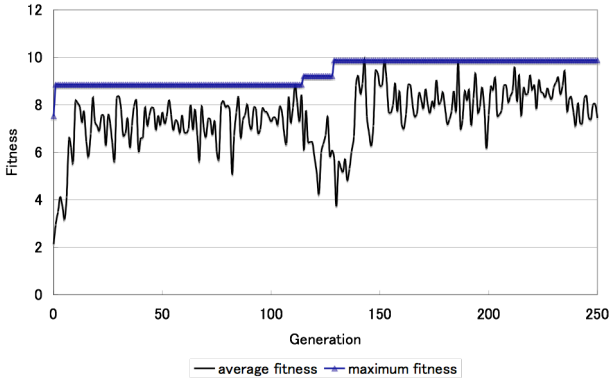

Fig. 6. Graph of average and maximum fitness of population against generation for the two-dimensional CA model of the robot.

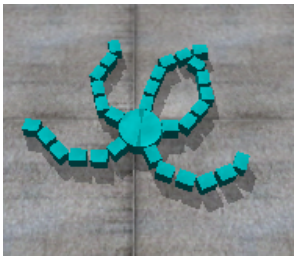

Fig. 7. Snapshot of the simulation for the robot using single transition rule for all the modules represented by a two-dimensional CA lattice

4 Differential transition rule for connected leg set

Building on the earlier models, in this section we present the final model, which represents the robot using a two-dimensional CA lattice as in § 3, however there are two distinct transition rules for updating the cell states in the lattice. These rules are termed as control rule, *CR* and leg rule, *LR*. The position of a cell (*i,j*) in the lattice determines which of the two rules would be used to update its state and is given as follows.

$$Rule = \begin{cases} CR & if\ j = 0 \\ LR & otherwise \end{cases} \quad (\mathbf{6})$$

Essentially modules closest to the central disc are covered by control rule and modules located on other parts of the leg are taken care of by the leg rule.

4.1 Co-evolving the Rules

In the previous sections using GA to search for good transition rules often resulted in premature convergence to local optima. In this section we use co-evolutionary algorithm to remedy this problem. Co-evolutionary algorithm [9] is a computational model where two or more populations simultaneously evolve in a manner such that the fitness of an individual in a population is influenced by individuals from other populations. This kind of external pressure, forces the populations to adaptively evolve thus avoiding suboptimal solutions. Co-evolution can be competitive in a nature such as in [10] where the population includes opponents trying to beat each other in a game of Tic-Tac-Toe, or it could be cooperative in nature such as in [11] where populations of match strings collaboratively contribute individual string to form a set of binary vectors which is to closely match a target set of binary vectors.

In our case a population of control rules and another population of leg rules is co-evolved cooperatively such that rules from both population need to be combined to control the robot. The following describes the pseudo code for the co-evolutionary algorithm.

Create random population of control rules $\mathbf{P}_{CR}$ and leg rules $\mathbf{P}_{LR}$
LOOP UNTIL MAX_GENERATION
  Get best control rule //*Rule with the highest fitness; for initial population*
    //*best rule is randomly chosen*
  LOOP UNTIL MAX_COEVOL_GENERATION
    //*Evaluate fitness of individual leg rules*
    LOOP FOR all individuals in $\mathbf{P}_{LR}$
      - Transform lattice model using best control rule and current leg rule for SIM_STEPS
      - Get final robot position
      - Calculate fitness, $F = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$

    END LOOP

Perform GA operations on $\mathbf{P}_{LR}$
END LOOP

Get best leg rule
LOOP UNTIL MAX_COEVOL_GENERATION
//Evaluate fitness of individual control rules
LOOP FOR all individuals in $\mathbf{P}_{CR}$
- Transform lattice model using best leg rule and current control rule for SIM_STEPS
- Get final robot position
- Calculate fitness, $F = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$

END LOOP
Perform GA operations on $\mathbf{P}_{CR}$
END LOOP
END LOOP

4.2 Simulation Results

The simulations were carried out again using the same parameters as in §2.4. The results show marked improvement in the fitness of the best rules discovered (Fig. 8). Noticeably the average fitness of the populations is less compared to simulation results from other models (Fig. 4&6). The reason for this is that the interaction between the populations within the scope of the co-evolutionary algorithm creates greater diversity thus requiring greater adaptation on the part of individuals in the populations. Consequently the stagnation of the search algorithm at local optima that was prevalent in the previous simulations was not encountered. Finally observations using the best rule discovered in controlling the movement of the simulated robot in ODE environment showed far greater coordination and fluidity of motion than any of the other models.

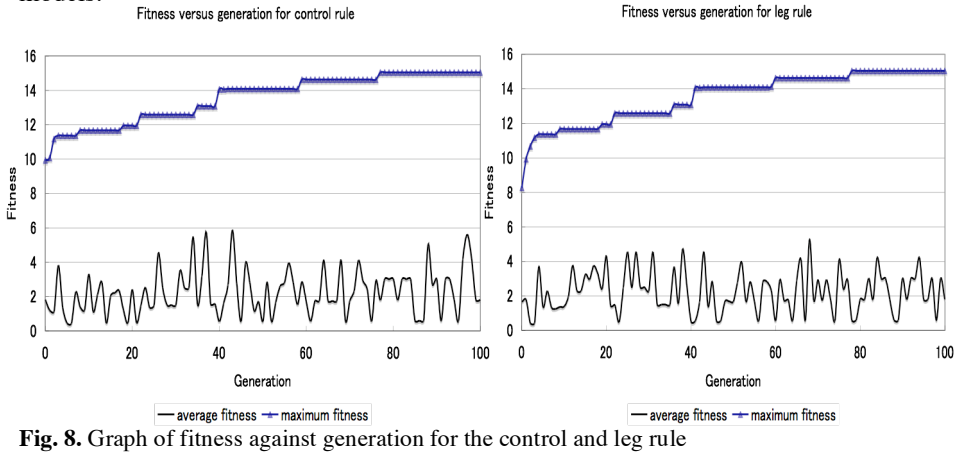

Fig. 8. Graph of fitness against generation for the control and leg rule

5 Conclusion

In this paper we demonstrated the feasibility of using cellular automata based architecture for control of a modular robotic system. Out of the three control models considered the model, which used differential transition rule on two-dimensional CA lattice representation of the robot, proved to be the most effective in controlling the motion of robot in a straight path.

Needless to say there are still much room for improvements. One of the crucial areas that need further research to be carried out deals with the fact that the rules obtained were tightly coupled with the initial state of the lattice configuration. Naturally the performance of the rules and the robot as a whole would get affected should the initial configuration be changed. To address this, in our future work we plan on using reinforcement learning as a means to train the robot to appropriately select rules based on any given class of initial configuration so as to accomplish some user-specified high-level goal.

References

1. Takashi, M.: Studies on Forward Motion of Modular Robot. MSc Dissertation, University of Ryukyu (2005)
2. Futenma, N., Yamada, K., Endo, S., Miyamoto, T.: Acquisition of Forward Locomotion in Modular Robot. Proceeding of the Artificial Neural Network in Engineering Conference, St. Louis, U.S.A (2005) 91-95
3. Crutchfield, J. P., Mitchell, M., Das, R.: The Evolutionary Design of Collective Computation in Cellular Automata. In: Crutchfield J. P., Schuster P.K. (eds): Evolutionary Dynamics-Exploring the Interplay of Selection, Neutrality, Accident, and Function. Oxford University Press, New York (2003) 361-411
4. Weisstein, E. W.: Life. Mathworld-A Wolfarm Web Resource. http://mathworld.wolfram.com/Life.html
5. Butler, Z., Kotay, K., Rus, D., Tomita, K.: Cellular Automata for Decentralized Control of Self-Reconfigurable Robots. Proceedings of ICRA Workshop on Modular Self-Reconfigurable Robots (2001)
6. Stoy, K.: Using Cellular Automata and Gradients to Control Self-reconfiguration. Robotics and Autonomous Systems, Vol. 54 (2006) 135-141
7. Behring, C., Bracho, M., Castro, M., Moreno, J. A.: An Algorithm for Robot Path Planning with Cellular Automata. ACRI-2000: Theoretical and Practical Issues on Cellular Automata. Springer-Verlag (2000) 11-19
8. Goldberg, D. E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley (1989)
9. Koza, J. R.: Genetic Evolution and Co-evolution of Computer Programs. In: Langton, C. G., Taylor, C., Farmer, J. D., Rasmussen, S. (eds): Artificial Life II, SFI Studies in the Science of Complexity, Vol. X. Addison-Wesley, Redwood City, CA (1991) 603-629
10. Rosin, C. D., Belew, R. K.: Methods for Competitive Co-evolution: Finding Opponents Worth Beating. In: Proc. 6th International conference on Genetic Algorithms. San Francisco (1995) 373-380
11. Potter, M. A., Jong K. A. D.: Cooperative Coevolution: An Architecture for Evolving Co-adapted Subcomponents. Evolutionary Computation, Vol. 8(1). MIT Press (2000) 1-29