12

Evolutionary Morphology for Polycube Robots

Takahiro Tohge and Hitoshi Iba
University of Tokyo
Japan

1. Introduction

Recent developments in robot technology have led to modular robots that are assembled from multiple parts. Modular robots have pros and cons. Their shape can be changed to suit a particular purpose and the same parts can be reused, which makes them fault-tolerant and extendable. However, such freedom and flexibility lead to higher cost so as to decide the configuration and actions to meet the purpose.

With conventional robots, modular or non-modular, much research has been conducted on the way that a design is determined first and connections with sensory input, internal memory and motor output are acquired through learning. The designs are often based on existing creatures such as insects, dogs and earlier robot models, or parts of such creatures, for example, robot arms. As robot bodies operate in the real world, it makes sense to model them after living creatures that have been successful in nature. In fact, computer simulation patterns akin to living creatures have been created mainly in the field of artificial life, represented by the well-known L-System imitating plants and Dawkins' biomorph (Dawkins, 1986) imitating insects. Sims' virtual creatures (Sims, 1994a; Sims 1994b) feature not only a pattern but also a body and nerves, which are actually stylized art rather than robotics. Modular robots, which are easy to configure and recombine, have an advantage in that optimal robot shapes can be achieved, without modeling the existing creatures. Many existing modular robots are based on the block shape. Although automatic design research is not limited to the block shape, the shape has certain advantages: The basic units are easy to make, handle and combine. Furthermore, blocks are similar to cells, which enabled the simulation of biological development in the research on building structures.

GOLEM Project (Lipson, 2000; Pollack, 2000) is one of the implementation models of Karl Sims experiments. They evolved frame-type virtual creatures using a linear actuator and fabricated them as a robot in a real world. In the modular robotics, new types of robots have been proposed with the functions such as a self-reconfiguration (Murata et al., 2002; Kurokawa et al., 2003) or self-reproduction (Zykov et al., 2005; Griffith et al, 2005). The LEGO Mind Storm is a famous block-based modular robot, parts of which are so small that half-assembled modules are combined in different placements (Lund, 2001). Ideally, robots can actively change their configuration by automatically recombining modules to fit a particular environment. However, this chapter deals with cube-shaped modular robots that can be manually recombined to achieve significant configurations that can adjust to actual environments using evolutionary morphology. This chapter describes how successfully EC

Source: Frontiers in Evolutionary Robotics, Book edited by: Hitoshi Iba, ISBN 978-3-902613-19-6, pp. 596, April 2008, I-Tech Education and Publishing, Vienna, Austria

is applied to the morphology for real cubic robots. More precisely, we empirically show the following points by several experiments:

- EC is effectively applied to designing the morphogenesis of cubic robots.
- Evolutionarily designed morphogenesis proves to be superior to manually designed robots in terms of the performance, e.g., speed or stability.
- The effectiveness of our approach is demonstrated by testing the generated behavior with a real robot, i.e., ROBOCUBE.

The rest of this chapter is organized as follows. The next section introduces the ROBOCUBE we have used as a kind of cubic modular robot. Section 3 describes two proposed methods of the evolutionary morphologies. Section 4 and 5 present the experimental results to show the effectiveness of our approach. The discussion of these experiments is provided in Section 6. Finally, Section 7 concludes the chapter.

2. ROBOCUBE

A block-based modular robot, ROBOCUBE, is used for the experiment. A ROBOCUBE block is a 5cm3 cube, each face of which has four connectors to be connected by button-shaped contact points to interlock the power supply line and the communications line to automatically establish a network for each module. Blocks are classified as sensor-related blocks (e.g., supersonic waves, light, sense of touch, gyro), motor blocks, output-related blocks (e.g., emission of light, buzzer, relay) and wireless blocks, which can provide wireless connectivity. Although one core block and one power source are essential for one robot body, once they are available, the robot is activated regardless of the connection; therefore, an enormous number of combination patterns exist. The number of blocks required to activate the actual robot is limited, and in case a block with certain functions is used, the maximum number of blocks to be used for the whole body is 16. The actuator for ROBOCUBE is a motor block with a DC servomotor embedded in it. The motor block has a turning-angle sensor for velocity and position controls and is used for movement with wheels or to make a turn by connecting to a sensor block. However, connecting such functional blocks necessitates wireless blocks or cables because each line is split at the motor axes.

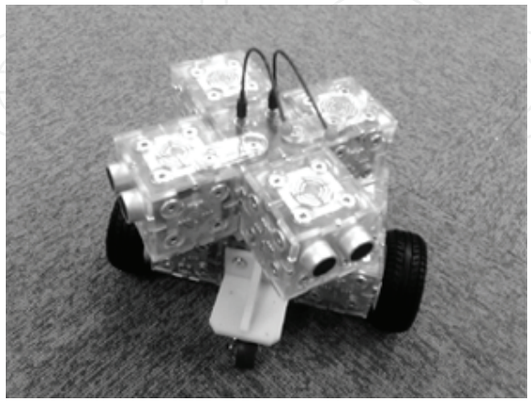


Figure 1. A sample of ROBOCUBE

3. Morphogenesis

We used the following models for robot morphogenesis. The first model used Messy GA (Goldberg, 1989) and cellar automaton, whereas the second adopted GP (ANT). The first model easily configures a lump mass with a method modeling the development of living creatures, and the second effortlessly achieves a string-shaped structure with a method using the ANT route.

There are two restrictions on morphogenesis for ROBOCUBE: the body must neither split nor interfere. The restrictions interact, and eliminating a block in a certain position to avoid interference creates the possibility of decoupling. There are several methods of expressing morphology via a certain kind of coding, however, the important issues are the fitness of the evolutionary computation and how to easily avoid the above restrictions. Each model is detailed below:

3.1 MODEL Using Messy GA

This development model is based on the conditions for connecting with adjacent blocks. The development model, or the cellar automaton that never dies, is a set of conditions to determine in which positions to connect a new block according to how the surrounding blocks are connected.

Think of a simple 2-D case. Focusing on one block, there are four adjacent dimensions: top, bottom, left and right. Four flags, which correspond to each direction, are prepared both for the condition and the action parts. If the flag for the condition part is 1 (meaning true), the block must be connected in that direction. A 0 (meaning false) flag shows that the block must not be connected in that direction. When the condition part fits in each of the four directions, a block is connected in a direction for which the condition part flag is 1.

This is the basic idea of the development model based on the conditions for connection. However, in reality, various extended models exist by adding other conditions including a coding method. Use of the development concept helps to reduce the parameters for determining the robot design and works well with the evolutionary method. This model can also easily resolve the interference issue by introducing dummy blocks. To insert a motor block, for example, dummy blocks are aligned along a trajectory of moving parts to prevent other blocks from being inserted in positions that have the possibility to interfere. (Nakano et al, 1997; Asai, 2002) is part of the research on this development method for actual robots.

The easiest method for implementing this development model is to prepare a set of rules covering all these conditions. One development condition is obtained given a character string 16*4 = 64 bit if the position of the character string is considered to be the condition part. The 0th 4 bit is an action part corresponding to 0000 of the condition part, the first 4 bit is an action part corresponding to 0001 of the condition part and so on. The merit of this method is that only character strings of a fixed length are necessary (64 bit), which allows easy adjustment to a genetic algorithm. A risk of this method is that a small number of genes could determine the final morphology. The genes of the former half, for which the condition parts are 0, are highly influential, therefore the location of each gene is a key factor and a 64 bit chromosome has only 16 genes.

To solve this problem, we consider providing more than one pair of a set of conditions and action parts, i.e., a basic model and its corresponding rule. For instance, the length of a chromosome of an 8 bit gene consisting of a 4 bit condition part and a 4 bit action part is 8

bit * N. This rule corresponds to a gene of Messy GA, checks the condition part from the front gene and starts connecting blocks based on the action part of the first gene that meets the condition. Each gene is moved from front to back and from back to front by crossover; therefore, a dependency on the location of genes is reduced for the aforementioned model.
This model may have the disadvantage of developing slowly because the small size of the genes can cause difficulty in finding a corresponding condition part. In this case, introducing ''*:don't care'' (meaning that the conditions for a connection are disregarded for the location) would solve the issue of a smaller number of corresponding condition parts. However, the action parts for genes of variable length are still 4 bit and therefore the condition parts containing many 1s make the corresponding action part meaningless. For example, an insignificant rule such as the action part 1100 corresponds to the condition part 1100 could be generated. Although such an insignificant action part can be considered an intron in a chromosome, the action part is disassembled as follows by increasing the condition parts to reduce the impact of one gene on the entire development process.
Genes originally have a one-directional condition part. Now focusing on one block in the development stage, to connect a new block with the block, an opening must exist in at least one of the four directions. The directional condition part leaves it to the manifestation condition of the genes. When conditions for connection other than the directions written on the rearward condition part of a gene are confirmed and met, the manifestation of the gene occurs and it adds additional blocks (i.e., action) in the direction of the directional condition part. In case of two dimensions, for instance, if the bottom is selected as the directional condition part, additional blocks are connected in the eight patterns as shown in Fig.2. This method reduces dependency on one gene and therefore the size of a chromosome, i.e., the number of genes must be increased to some extent. In that case, dummy blocks, which are assumed for the interpretation of directional conditions, are not considered to exist when conditions for connection are interpreted.

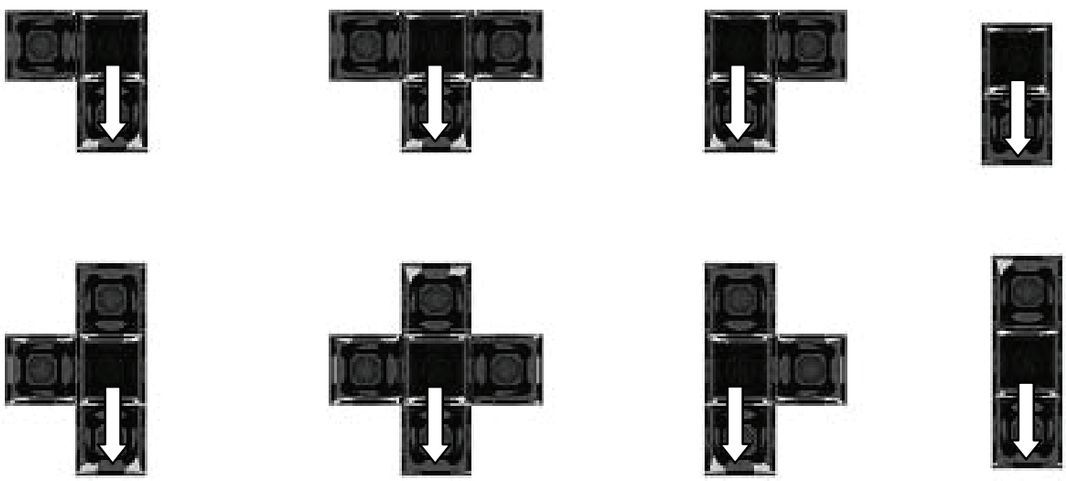


Figure 2. The eight development patterns in 2-D (Each white arrow means a new connection)

As only simple conditions for connecting cellar automaton are expected to lead to a monotonous development process, including the manifestation conditions by the age of the block during the development process in what step the block was connected can ramify the blocks. Considering the above, the G-TYPE is defined as a gene of Messy GA, which is 3-D as in the experiment.

Condition			Action
Direction	Connection	Step	Kind

- Direction: directional conditions (top, bottom, left, right, forward, backward)
- Connection: conditions for connecting in the five rest directions (5 bit)
- Step: manifestation occurs when the age of the block is less than the step (from 0 to the maximum size of the block)
- Kind: kinds of blocks (normal, motor, sensor)

Considering the condition parts as the locus, the use of such genes can cause Messy GA evolution. The actual procedure is as follows:

1. Initialize chromosomes at random to generate the initial population.
2. Morphogenesis for the initial-state robot by gene manifestation.
3. Morphogenesis ends when the genes for manifestation run out or the specified number of blocks is reached.
4. Implement a task to determine the fitness.
5. Generate the next-generation population through selection, crossover and mutation.
6. Return to No. 2 above.

3.2 Models Using GP

In these models, the Agent (ANT) aligns the functional blocks of module-combination-type robots in 3-D. That is, an agent exists that can take actions such as ''turn right,'' ''turn left,'' ''look up,'' ''look down'' and ''put a block and go forward,'' and the morphology that is formed when the agent has finished aligning becomes that of a robot that implements the task. As an agent must align the blocks before it moves, the blocks are continuously aligned. In 3-D, the length is expressed by the x axis, the width by the z axis and the height by the y axis. Only two kinds of blocks are used: the motor block, which can rotate in any arbitrary angle, and the normal block, which cannot. Agents directly align only normal blocks, among which those that are already aligned and meet the following conditions are converted into motor blocks.

$$d_{xz}(RG;BG) > 1 \tag{1}$$

$$l = \max(d_{xz}(RG;BG)) \bullet r \qquad (0.1 < r < 0.9) \tag{2}$$

The equation $d_{xz}(a; b);RG;BG$ expresses the Euclidian distance, the gravity point of a robot and that of a block, respectively, with regard to a and b on the xz plane. The terminal and non-terminal nodes of GP used in this paper are shown in Tables 1 and 2. The expression ''allows connection of motor'' refers to whether connecting with a block that serves as a supporting point of motor rotation is possible. If there is more than one candidate that can be connected, the plus (+) directions of the z, y and x axes are searched sequentially and connected to the first block that is found. A motor block is vertically connected to the z axis. After determining where the motor is to be connected, the normal blocks that interfere with the rotation are eliminated. A body is formed in this way, and the GP procedure is applied to the balance.

Function name	Definition
Prog2	Take two terminal nodes and implement them in order.
Prog3	Take three terminal nodes and implement them in order.

Table 1. Non-terminal nodes

Function name	Definition
Right	Turn right
Left	Turn left
Up	Turn up
Down	Turn down
Put and move 1	Put a normal block that allows connection of motor and move forward.
Put and move 2	Put a normal block that does not allow connection of motor and move forward.

Table 2. Terminal nodes

4. Experiment 1: Moving the plane surface

4.1 Setting of the Environment

The first experiment attempts the evolutionary morphology of the working leg parts instead of wheels, without using sensors. A plane surface is assumed, and a robot proceeds where there are no obstacles. The initial state is a straight line of core blocks between two motor blocks that has one block as the leg part as shown in Fig.3. Even though the robot moves forward in this state, its rotating block touches the floor for such a short time that it moves only incrementally.

The Messy GA + CA method was applied to morphogenesis. A maximum of 10 blocks were used for the leg part. The core block has two rollers to help reduce the impact of friction on the simulator. However, the rollers can become meaningless objects or even obstacles depending on the configuration of the robot. Controllers to output constant rotations and sin waves should be prepared in advance to define the following action genes in a chromosome.

The action genes are initiated at random and exist in chromosomes at a constant probability. If an action gene corresponds with a certain motor, the action is output; otherwise, no action occurs.

Fitness is computed by the moving distance on the plane [cm] multiplied by the stable movement time, and the simulation is conducted up to a maximum of 5[s]. If a controller does not exist, fitness is computed by reducing the maximum number of blocks (10) from the number of blocks of the leg part. Better fitness is obtained where more blocks are used.

The following GA parameters were used. The Messy GA method used cut and splice for the crossover. If the length of a chromosome exceeds 1000 due to crossover, the experiment is repeated from the selection.

- Number of individuals: 128
- Number of generations: 512
- Length of the initial chromosome: 20
- Maximum length of a chromosome: 1000
- Probability of mutation: 0.05

- Method of selection: tournament method (size 4)
- Generation Gap: 0.1

The experiment was made using a dynamic 3-D simulator-loading engine as in an actual environment, and the actual machines were used to ensure functionality of the obtained individuals.

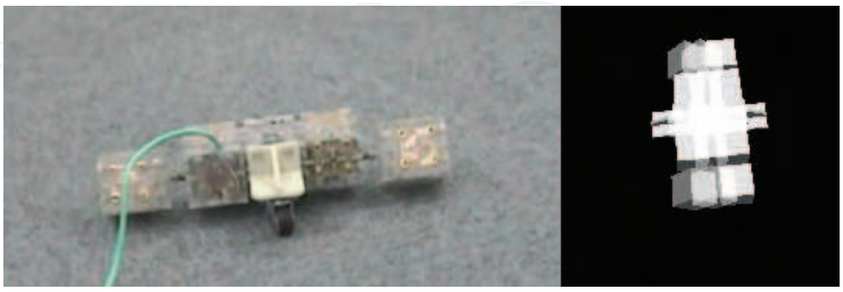


Figure 3. The initial structure and the movement (Plane surface)

4.2 Result

Fig.4 shows the maximum and average fitness for each generation at the time of action. The ”wheel” means the fitness of a normal wheel robot (Fig.7), of which we can think as a target example for this moving task. Fig.5 shows some of the individuals that emerged during the evolution, and Fig.6 shows the second optimal individual resulting from the evolution. The optimal individual uses the leg part as a support during rotation, and the central rollers handle the rotating movement. As you can see from 4, the fitness of the best individual (e.g., Fig.6) is superior to the normal wheel robot. We performed the same experiment with the actual machines and observed the behavior in the real world.

It may be easy to imagine individuals that adopt a controller of constant rotation, maximizing the diameter of the leg parts, however, an individual as in Fig.6 was actually obtained. The moving distance was maximized by rotation using the rollers attached to the body, which was realized by spreading the leg part in one direction. In reality, the individuals that maximize the diameter, such as the one with the cross-shaped feet, cannot take advantage of friction as a driving force, thereby spinning around and going nowhere.

Table.III shows the fitness values of evolved individuals. Fitness is measured as the distance from the initial position at the end of simulation. Note that the performance of the suboptimal robot is 150% better than the wheel robot. The cross-shaped robot (see Fig.7) was stuck by its edge and could barely move. As a result of this, its fitness value is very small. This table clearly shows the effectiveness of the proposed evolutionary morphology in terms of the fitness values.

During evolution, some were seen to move forward, leaping lightly by generating asymmetrical leg parts to take sin-wave output. From the viewpoint of the practical movement method of modular robots, they can be classified into those that transform like ameba to change a whole location and those that use leg parts or wheels. Although this result is an option due to a higher velocity than wheels, it requires sufficient strength and a function to shift directions.

In this model, each gene does not have significant impact and therefore the initial state must contain various genes, which is made possible by increasing the number of individuals and

the initial chromosome length. In the experiment, the fitness computation from implementing the tasks was so costly that the initial chromosome length had to be extended. As this issue tends to increase the selection pressure, we tried to avoid it by reducing the generation gap. However, the average fitness was actually apt to shrink as if it was pulled by the optimal individual.

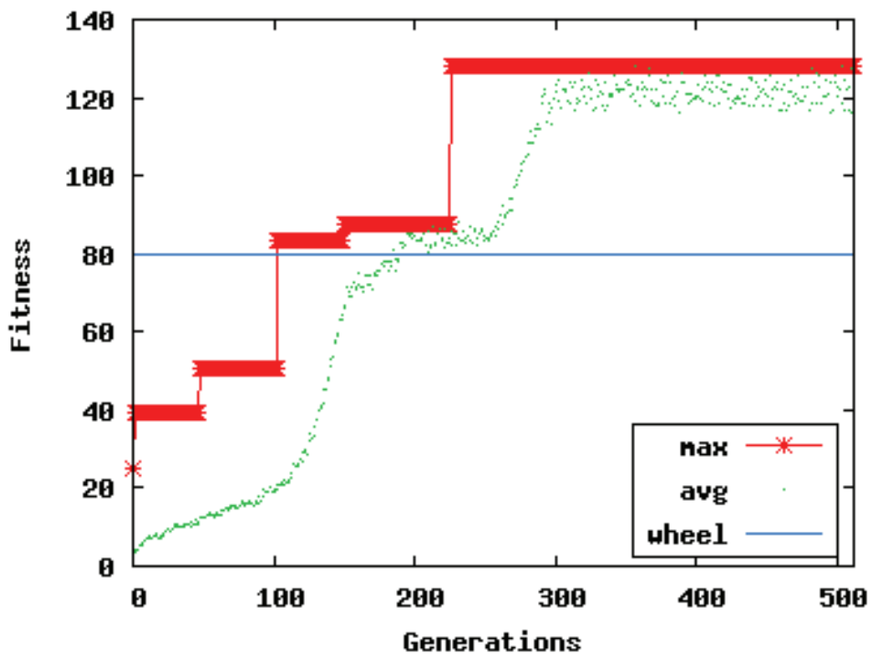


Figure 4. Change in fitness (Plane surface)

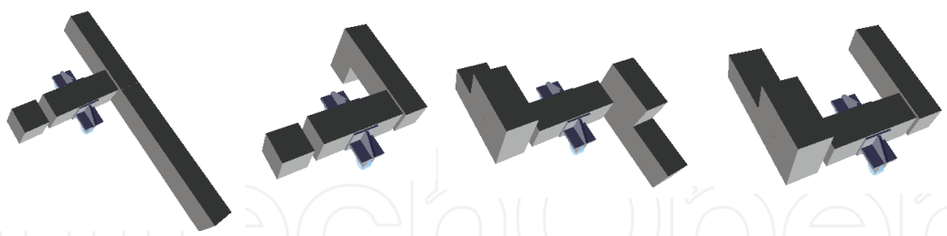


Figure 5. The suboptimal individuals in each generation (Plane surface)

Figure 6. The best individual and the movement (Plane surface)

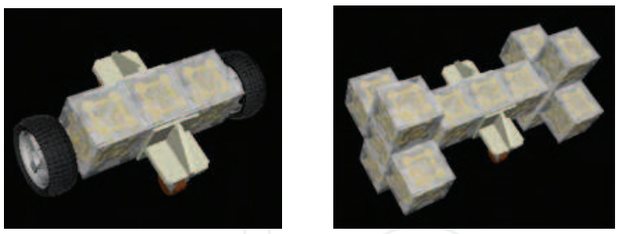


Figure 7. The wheel robot and the cross-shaped robot

Type	Fitness
Best	127.94
Wheel	80.83
Initial	4.68
Cross-shaped	2.46

Table 3. Fitness values

5. Experiment 2: Ascending a slope

5.1 Setting of the Environment

This experiment attempted to ascend stairs via morphogenesis using GP. Fig.8 shows the stairs used in the experiment. The stair height was set higher than the usual wheel forward movement. The colored part shows the stairs that the robots ascended. The stair width was 2 m, exceeding which meant the robot falling and terminating the attempt. Staircases were located 1 [m] away from the initial location of the robot in each of the four directions. The direction of the robot's initial location did not matter.

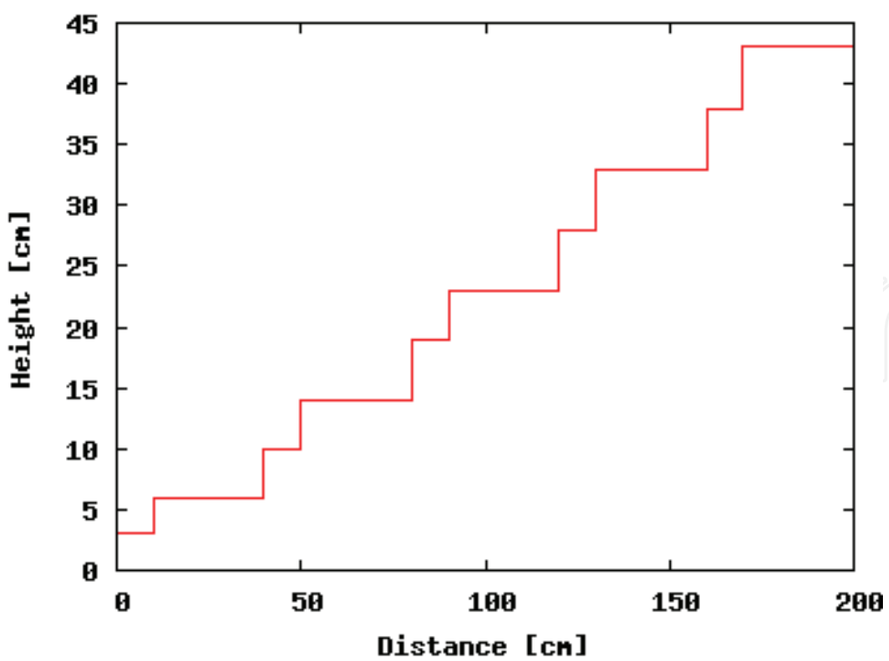


Figure 8. Section of a staircase

The controllers were not changed, and the robots were made to rotate under a certain rule in a constant direction. The initial evaluated value is 1500, and the value is reduced by 500 when the robot reaches the first stair and by 100 for each subsequent stair that the robot ascends. The value of the trial time divided by 10 is added at the end of the trial. The maximum trial time was 1000. The less the evaluated value is, the better. If no normal blocks emerge during the action, an evaluated value of 3000 is given.
The following parameters were used:

- Number of individuals: 250
- Number of generations: 50
- r = 0:9
- Selection method: Tournament method (size 6)

5.2 Result

Fig.9 shows the minimum fitness and the average fitness for each generation at the time of implementing GP. Fig.10 shows the good individuals that emerged in each generation. The light-colored blocks in the center are normal blocks, and the dark-colored blocks at the edge are motor blocks.
Fig.10(a) shows an individual that emerged in the third generation. Because only one motor emerged, which made proceeding straight ahead difficult, it was not able to achieve a task. Fig.10(b) shows an individual that emerged in the eighth generation, featuring five blocks vertically aligned.
The robot supported itself using its long vertical part and got on the gap between the stairs, but the tilt could not be supported. Afterward, the robot deviated from the right orbit. The individual that emerged in the 14th generation, shown in Fig.10(c), ascended the stairs, supported by the five central blocks as well. Starting with the eighth generation, there were two more blocks at the front. The wider shape could avoid deviation from the orbit caused by the tilting at the time of climbing over the gap in the stairs.
Fig.11 shows the best individual during the simulation, which emerged in the 16^{th} generation. Putting the three blocks on the right-side front of the robot upon the next stair, the robot could start climbing the staircase. Compared with Fig.10(c), there were fewer front blocks, thereby eliminating the front block obstacles and allowing the robot to ascend over the gap in the stairs and shorten the time to realize the task.

6. Discussion

We showed two experiments with different coding methods, i.e., messy GA and GP. In modular robotics, it is very important to acquire not only one best pattern but also various sub-optimal patterns. This is because modular robots are used and developed for the sake of search or rescue task, which requires the flexibility against the real-world environment. In such a case, moving is a basic and essential ability skill.
Our experimental results have shown how successfully the significant pattern was acquired during the evolution. As can be seen from Fig.4 and Table.3, some of those patterns were superior to those designed manually for the target task.
In this experiment, sensors were not used to avoid complicating controllers. Introducing sensors necessitates a certain level of dynamic generation of controllers and symbiosis with morphology models. Some computation cost is necessary, for which an application of

Neural Net or Q-Learning is a solution. Research in this field includes the co-evolution of morphology and behavior as well as the automatic generation of controllers. However, we intend to deepen our knowledge of expression methods of morphology for actual robots, and in that regard we will attempt the co-evolution of vegetable-type robots using only sensors and animal-type robots with motors.

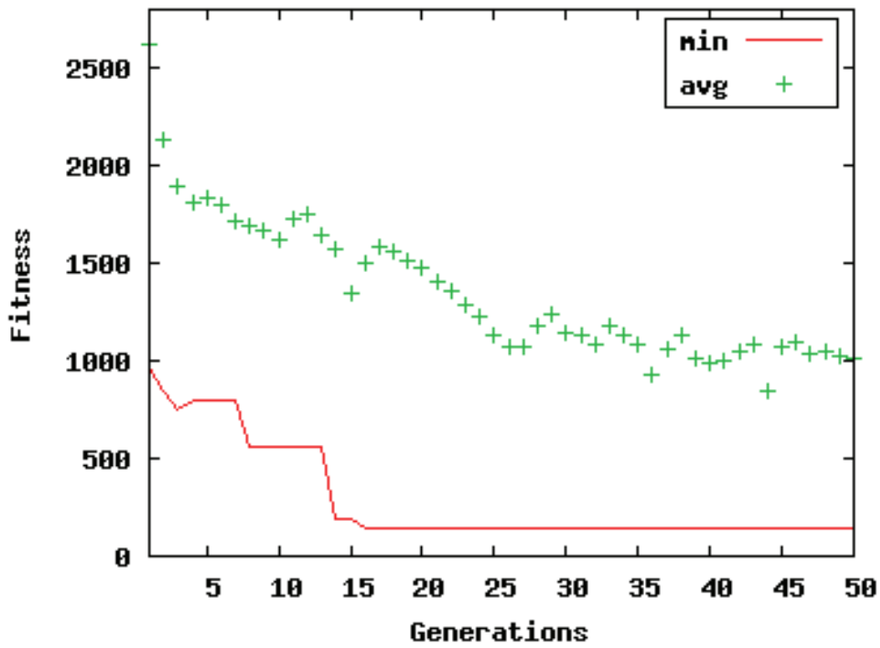


Figure 9. Change in fitness (Slope, less is better)

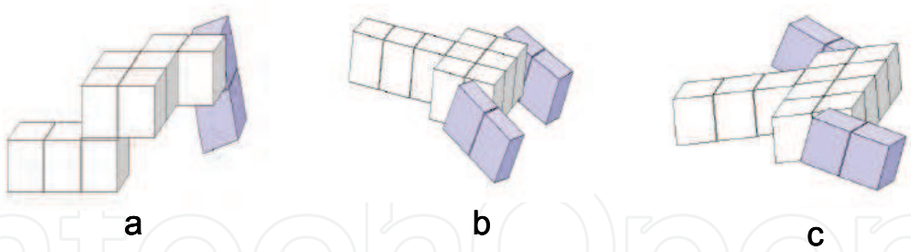


Figure 10. The suboptimal individuals in each generation (Slope)

Figure 11. The best individual (Slope)

7. Conclusion

The chapter covered the evolutionary morphology of actual robots using a 3-D simulator. An individual fitted for plane movement using Messy GA rotated its body by extending the leg parts in a direction perpendicular to the rotation axis to increase stability. The individual that fitted itself to the ascending stairs by GP adjusted well to the stair gap by extending the support on the back of the body in a flat and a hook shape. Either shape is typical of block-type robots, but uncommon for humans, and reflects the versatility of blocks as a basic unit. A future issue will be to apply to more complicated tasks or multi-agent cooperation using modular robotics.

8. Acknowledgements

The authors thank Kenta Shimada and Taiki Honma for helpful comments and for other contributions to this work.

9. References

Asai, Y. & Arita, T. (2002). Coevolution of Bodies and Behavior in a Modular Reconfiguration Robot, *IPSJ Journal (in Japanese)*, Vol. 43, No. SIG10, pp. 110-118

Dawkins, R. (1986). *Blind Watchmaker*, Longman

Goldberg, D. E.; Korb, B. & Deb, K. (1989). Messy Genetic Algorithms: Motivation, Analysis, and First Results, *Complex Systems*, vol. 3, pp.493-530

Griffith, S.; Goldwater, D. & Jacobson, J. M. (2005). Robotics: Self-replication from random parts, *Nature*, Vol. 437, pp. 636

Kurokawa, H. et al. (2003). M-TRAN II: Metamorphosis from a Four-Legged Walker to a Caterpillar, *Proceedings of International Conference on Intelligent Robots and Systems (IROS 2003)*, pp.2452-2459

Lipson, H. & Pollack, J. B. (2000). Automatic Design and Manufacture of Artificial Lifeforms, *Nature*, Vol. 406, pp.974-978

Lund, H. H. (2001). Co-evolving Control and Morphology with LEGO Robots, *Proceedings of Workshop on Morpho-functional Machines*

Murata, S. et al. (2002). M-TRAN: Self-Reconfigurable Modular Robotic System, *IEEE/ASME Trans. Mech*. Vol. 7, No. 4, pp. 431-441

Nakano, K. et al. (1997). A Self-Organizing System with Cell-Specialization, *IEEE International Conference on Evolutionary Computation*, pp.279-284

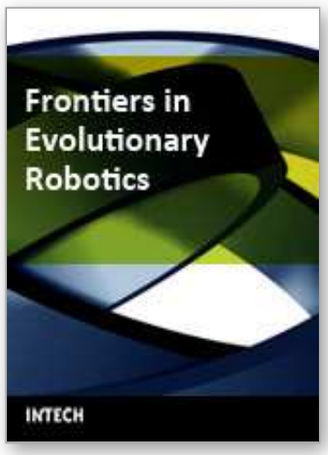
Pollack, J. B. & Lipson, H. (2000). The GOLEM Project: Evolving Hardware Bodies and Brains, *The Second NASA/DoD Workshop on Evolvable Hardware*

Sims, K. (1994a). Evolving Virtual Creatures, *Proceedings of Computer Graphics*, pp.15-22

Sims, K. (1994b). Evolving 3d morphology and behavior by competition, In R. Brooks and P. Maes, editors, *Proceedings of the International Conference Artificial Life IV*

Takahiro, T; Shimada, K. & Iba, H. (2006). Evolutionary Morphology for Cubic Modular Robot, *Proceedings of 2006 IEEE World Congress on Computational Intelligence*

Zykov, V. et al. (2005). Self-reproducing machines, *Nature*, Vol. 435, pp.163-164



Frontiers in Evolutionary Robotics

Edited by Hitoshi Iba

ISBN 978-3-902613-19-6

Hard cover, 596 pages

Publisher I-Tech Education and Publishing

Published online 01, April, 2008

Published in print edition April, 2008

This book presented techniques and experimental results which have been pursued for the purpose of evolutionary robotics. Evolutionary robotics is a new method for the automatic creation of autonomous robots. When executing tasks by autonomous robots, we can make the robot learn what to do so as to complete the task from interactions with its environment, but not manually pre-program for all situations. Many researchers have been studying the techniques for evolutionary robotics by using Evolutionary Computation (EC), such as Genetic Algorithms (GA) or Genetic Programming (GP). Their goal is to clarify the applicability of the evolutionary approach to the real-robot learning, especially, in view of the adaptive robot behavior as well as the robustness to noisy and dynamic environments. For this purpose, authors in this book explain a variety of real robots in different fields. For instance, in a multi-robot system, several robots simultaneously work to achieve a common goal via interaction; their behaviors can only emerge as a result of evolution and interaction. How to learn such behaviors is a central issue of Distributed Artificial Intelligence (DAI), which has recently attracted much attention. This book addresses the issue in the context of a multi-robot system, in which multiple robots are evolved using EC to solve a cooperative task. Since directly using EC to generate a program of complex behaviors is often very difficult, a number of extensions to basic EC are proposed in this book so as to solve these control problems of the robot.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen

IntechOpen