First Asian Symposium on Industrial Automation and Robotics
BITEC, Bangkok, Thailand
May 6-7, 1999

SOLVING THE SIMPLE TASK OF A BEHAVIOR-BASED ROBOT BY GENETIC PROGRAMMING

Boonserm Keawkamnerdpong, Chanchai Chaisukkosol , Orawan Chanpen and Jumpol Polvichai
Department of Computer Engineering
King Mongkut University of Technology Thonburi
91 Suksawad 48, Tung-Kru, Bangkok 10140, Thailand.
Tel. (662) 470-9089 Fax. (662) 872-5050

Abstract: *The behavior-based approach has been adopted in designing the dog-like robot. Five 68HC11 boards are assigned to control the movement of the robot. Four boards perform leg primitive behaviors of each leg and one board performs robot behaviors as its brain. By combining primitive behaviors, the robot will manifest complex behaviors such as walking forward, walking backward, turn around, and avoid hitting. Finally, the robot will do the job, walking and finding the way out from the maze, our simple task of this research. Evolutionary technique, Genetic Programming, is used to generate robot programs for exploring the exit route by simulating on the computer. These programs will be executed in a simulation environment. After finding the successful simulated robot programs, we use a 4-legged robot to perform the robot programs to prove that the robot can appreciatively work in real world. By combining behavior-based control system concept and evolutionary computation concept, the results of these experiments illustrate that using both of these concepts can efficiently solve a simple problem.*

Key words*: Robotics, Behavior-Based Approach, Genetic Programming and 68HC11 Microcontroller.*

1. INTRODUCTION

Behavior-based robotic system is one of the most famous in robotics field that realizes an intelligent robot to work well in the real world. Behavior-based approach has been showed the successive expedient of designing the control system to achieve this purpose. A number of behavior-based robots demonstrated the performance of this approach (Arkin, 1998).

However, the design of layered control system is more difficult when the complexity of the task or the interaction between the robot and an environment are increased. To solve this disadvantage, some researches have used the different ways.

Using evolutionary computation, the robot controllers have been evolved in the various forms by using the measure of fitness according to the task and its environments. (Polvichai & Chongstivatana, 1995, Chongstivatana & Polvichai, 1996 and Pattana et al., 1998),

Reinforcement learning process (Yamaguchi et al., 1996), another method, is the process of acting in the real environment without knowledge on a task based on giving rewards when the learning task is performed. It can learn even in unknown environment. However, it requires a lot of trials of the given task and large computation costs, that bring about the needing a long time to converge learning result problem. Accordingly, most reinforcement learning research made by simulation where the real environment is simulated in a computer or a virtual environment.

The approach at intermediate level between coding a behavior-based system and evolving an overall control system have been proposed. (Lee et al., 1997) This approach uses the behavior-based architecture as the control design and evolves the particular components by using an evolutionary approach. They also proposed the behavior primitives and behavior arbitrators.

2. PROBLEM STATEMENT

In this paper, we build the 4-leg, dog-like, robot by using the concept of behavior-based approach controlling the movement of the robot. To manifest the efficient of the system, finding the way out of the simple maze is assigned as the simple specific task of this robot. In stead of thinking to program the robot by human, genetic programming process is chosen to find the programs that can command the robot out of the maze. This process will be taken in the simulation. In final, the successful programs generated by GP process will be tested in the real world.

3. THE ROBOT

3.1 Robot Design

The robot used in this research has a dimension of 23 cm. x 30 cm. x 25 cm. and a weight of 1.5 kg.. Its four legs are driven by twelve FUTABU servo motors. These motors can be set angle between 0 – 180 degrees by signaling a pulse width modulation encoding. The robot is equipped with six touch sensors: two distant sensors are positioned on the front and four bar sensors are located around its body. The layout of sensors, numbered from 0 to 5, is showed in figure 1. To control the robot, we send signals by the host computer via a parallel port. All commands will be sent to a parallel board, which sent signals to four independent 68HC11 broads that control each leg.

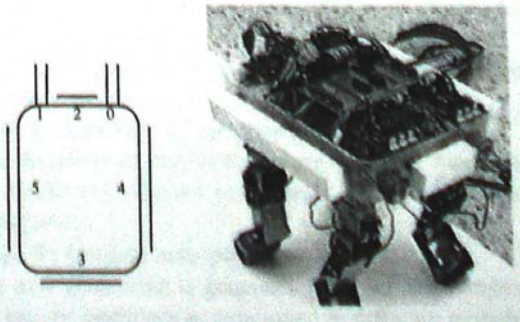

Fig. 1. The Dog-like robot and its sensor arrangement.

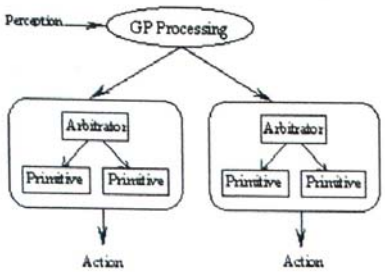

Fig. 2. The architecture of genetic programming processing and behavior-based system

3.2 Behavior Design

Behavior-based architecture is a new approach decomposed into task-achieving modules, also called behaviors. The general behavior-based system includes a set of behavior primitives and behavior arbitrators, coordinating them by the arbitrator used to select the primitive. All behavior components are autonomous and parallel working like the natural working.

A behavior arbitrator in our control system is treated as a reactive controller. It has the same structure as the primitives but it has a little bit difference. Outputs of a primitive are used to control the motor, but outputs of an arbitrator are used to activate a behavior primitive. We assign them as follow:

Primitives : forward swing, backward swing, and stop.

Arbitrators : forward walking, backward walking, left turning and right turning.

All primitives are downloaded into each 68HC11 broad controlled each leg and arbitrators are assigned at the host computer during executing robot programs. However, unlike the subsumption architecture (Brook, 1987), our control system is not a complicated architecture. It is used as a terminal in genetic programming process. Figure 2 illustrates our architecture control systems.

4. GENETIC PROGRAMMING PROCESS

Genetic Programming (Koza 1992) is a searching technique based on the mechanics of natural selection and natural genetics. The process of the genetic programming employs an evolution process to synthesize robot programs that appropriate with a specified problem. The performance of each program will be used as an indicator whether it can be selected into the genetic process for breeding the robot programs to the next generation. Then, the process continues repeatedly until reaching the programs that have a high performance.

To realize an intelligence robot that achieves the maze routing task reliably by genetic programming, the main evolutionary process of this work is similar to typical evolutionary approach. Given an environment and a goal formulated as the fitness function, initial population and selecting genetic operator are generated at random. By four genetic operators, reproduction, crossover, mutation and permutation, the new generation is generated.

In this work, there are three functions and ten terminals in Genetic programming process. That is,

Function : IF_NOT, IF_AND, IF_OR

Terminal : SL?, SR?, TS1?, TS2?, TS3?, TS4?, FORW, BACK, LEFT, RIGHT

In this research, the system uses the combination logic system which the output logic is determined by the current input state at each constantly time step. The program pattern consists of three or four arguments depend on selective function being if-CONDITION-then-ACTION-else-ACTION

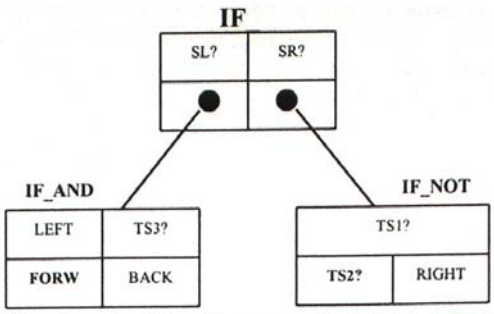

Fig. 3. The characteristic of our genetic system pattern and typical example of genetic program. In this figure, every node can be contained in any of all four-part.

pattern. The function IF_NOT is the three-argument complement operator that executes the second argument if the complement of the first argument is true, or otherwise, executes the third argument. The function IF_AND and IF_OR are four-argument comparative operators that executes the third argument if the logical result of combination logic between the first and the second argument is true, or otherwise, executes the forth argument. The characteristic of this pattern can be illustrated by figure 3.

It can be feasible that action terminal found in condition position and sensor conditional terminal found in action position. In case of deviate position, the system will do the action if it determines the deviate action terminal by alternately and do nothing if the deviate sensor conditional terminal is determined.

In the genetic programming process of this work, there are five stages in each searching cycle.

Stage 1 : Creation the initial population

First of all process of genetic programming, 100 initial population is generated at random by combining the functions and the terminals to form the given genetic pattern. Each initial genetic program is limited at four layers of repetitive searching.

Stage 2 : Verification of each genetic program

Each genetic program is executed repeatedly until any of following condition has been qualified.

- Execution time condition: not over 100 times.
- Dead condition: the robot did the same action over 50 execution time.
- Successful condition: robot found a goal area.

Stage 3 : Evaluation of genetic program

The genetic programming for maze routing task given the evaluating formula, or fitness function, that is

$$\text{Fitness}=20000 + (\text{isGoal} \times 200000) - (\text{minDistance} \times 1000) - ((-1)^{\text{hasObstacle}} \times 50000) - (\text{iskilled} \times 100000) - (\text{Time} \times 1000)$$

where *isGoal* is logical variable that indicate the position of the robot whether the satisfying position or not, *minDistance* is the shortest distance between the robot and goal in all hundred execution times, *hasObstacle* is logical variable that indicate the obstacle along the shortest distance between the robot and goal at the final execution time, and *iskilled* is another logical variable that denote dead condition of the robot.

This fitness function is gain function, higher value means better performance.

Stage 4 : Selection of the good programs

First hundred of the best genetic program evaluated by fitness function is selected to generate the new generation in 500 population.

Stage 5 : Genetic manipulation

The new generation is generated by using the combination of four genetic operators as mentioned in different probabilities as following.

Reproduction : 10%
Crossover : 60%
Mutation : 20%
Permutation : 10%

The reproduction is operation that duplicates the ten of best programs. Other operations are generated by randomly selection of evaluated programs.

5. EXPERIMENTS

In order to make possible for the robot can solve the problem in many environments, we select three simple mazes with different degree of difficulty. The first maze has one path to reach the goal. The robot has to walk in the right path. The second maze is defined two possible paths, the robot has to find the optimum path. The third maze is created a situation where a local minimum exists. The distance between start position and goal position is short, but the robot can not reach to the goal position directly. We also build the real maze that consistent with three mazes in simulation as well. (see Figure 4)

After genetic programming process by simulation, we get the optimum robot programs that can reach to the goal position in simulation. Finally, we locate these programs to the real robot for testing the different between running in real robot and running in simulated robot.

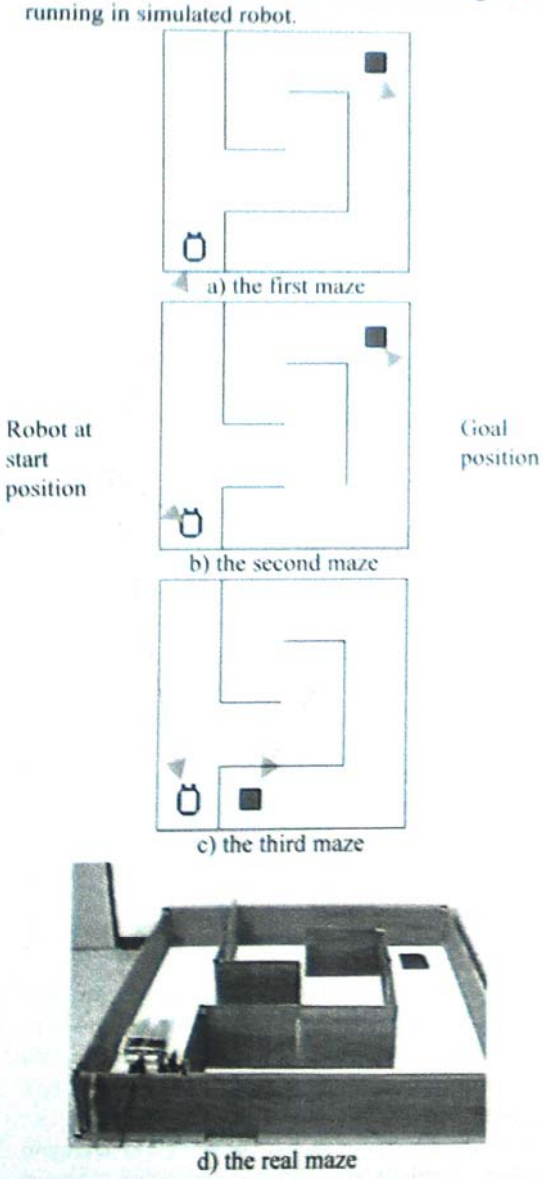

Fig. 4. Three simple mazes that use in experiments and the real maze

To evaluate a number of effective robot programs in real situation, an amount of time in learning is not appropriate to be used. We use the simulated robot instead for avoiding the speed limit of the real robot. The disadvantage of using the simulation is no corresponding to the real world due to some factors. We try to reduce these factors by construct the mapping function about robot walking. To construct this function, we get the real data by observing the inclined angle and walking speed. (the robot is controlled by walking commands such as 'FORW' : simulator move with speed 12 and inclined angle –0.12306 radian.)

Fig. 5. The example of the successful results in the first maze.

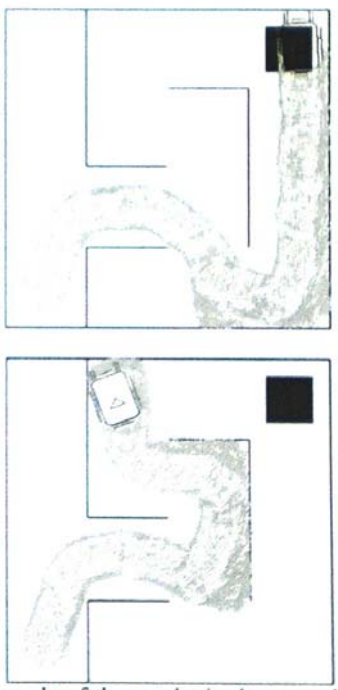

Fig. 6. The example of the results in the second maze, success (up) and failure (down).

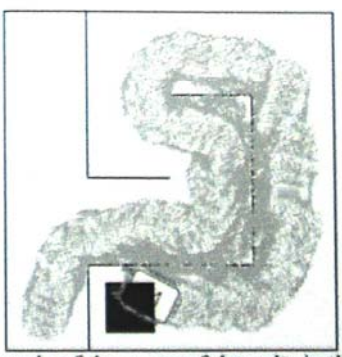

Fig. 7. The example of the successful results in the third maze.

6. RESULTS AND DISCUSION

6.1 The simulation process

The examples of the achieved robots of the first maze are showed in figure 5. This figure of maze rather spends more time to find the successful program, due to the difficult of generating the program with ability to turn left and turn right with the condition of a few touch sensors. The successful generated programs usually have a behavior of walking follow the right wall as well to bring the robot reach the goal position.

To find the successful programs of the second maze, it spends short time less than the first maze, because this maze has two possible paths to reach the goal position. Figure 6 shows the example of success and failure. For the second maze, pattern of the successful programs is somewhat the same way. At start, forward commands and turn right commands let the robot moving to the aperture without touching anything. When the robot hits the wall, the commands that make robot follow the wall to the left will reach the robot to the goal position. This pattern is easy to generate by genetic programming process. In another path, programs, which make the robot to turn left and turn right with a few touch sensors, are somewhat difficult to find by the genetic programming process.

For the problem of the third maze, with short time running, the results have showed no successful robot program that can reach the goal position by using the same fitness function of the others. Because the fitness function is designed to give the good score depended on the distance of the robot and the goal position. This will let the programs that try to move to the way reaching the final position die out. In the long run, the successful programs come out (show in figure 7), but it takes a lot of time. We have try to use the staged evolution method, which we use the successful running of the first maze to be the first generation of the genetic programming process of the third maze. The outcomes have showed that some programs can reach to the final position faster than randomly generating of the first population. However, there are many programs, successful programs from the first maze, die out. Because the time limit make these programs unable to reach the goal position.

6.2 The real running

In the real world, we pick up the successful programs generated by the simulation process to test with the real robot. Almost robots can do the task in the same way that show in the simulation run. The reason for these successful running in real world is the successful mapping from running in simulation and running in real world. We put unpredictable action of our robot in the simulated run. Corresponding with the characteristic of the real robot, when program orders 'FORW' in simulation, the simulated robot won't forward in the straight way but it will add with the inclined angle randomly generated by simulation program. This uncertain feature makes more robust of the generated program.

7. CONCLUSION

The consequences of this research show that combination of behavior-based control system concept and evolutionary computation concept can efficiently solve a simple problem. The robot created by the concept of behavior-based approach can properly work with the program generated by genetic programming process. The robot programs are evolved with behavior primitives and behavior arbitrator, making the robot works well in real world. Nevertheless, transferring the learning results by simulation to a real robot in the real environment probably gains poor performance because of difference of environment, which is reflected by the errors in performing the task.

8. FURTHER RESEARCH

With the creating of new sensors, the real robot can solve more complicated problems. More perception will be increase performance of the robot to dial with the new situation in the real environment.

Another is to find the new way to let the real robot evolves its control systems by never using the simulation program.

9. ACKNOWLEDGEMENTS

Boonserm Keawkamnerdpong and Orawan Chanpen are supported by the National Science and Technology Development Agency of Thailand (NSTDA). Facilities provided by Department of Computer Engineering of KMUTT.

10. REFERENCES

Arkin, R. C. (1998). *Behavior-Based Robotics*. The MIT Press, 0-262-01165-4, U.S.A.

Brooks, R. A. (1987). *A hardware retargetable distributed layered architecture for mobile robot control*. Proceedings IEEE Conf. Robotics and Automation, April, pp. 106-110. NC.

Chongstitvatana, P. & Polvichai, J. (1996). Learning a Visual Task by Genetic Programming, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 534-540, 0-7803-3213-X, Japan, 1996

Koza,J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, , Massachusetts U.S.A.

Pattana, S.; Saetang, C. and Polvichai, J. (1998) Learning to walk of a 4-legged robot by Genetic Programming, National Conference on Computer Science and Engineering, Bangkok, 1998.

Polvichai, J. & Chongstitvatana, P. (1996). Visually-Guided Reaching by Genetic Programming, Proceedings of ACCV'95 the second Asian Conference on Computer Vision, pp. III-329 - III-333, December 5-8, Singapore, 1995

Yamaguchi, T.; Masubuchi, M.; Fujihara, K. & Yachida, M. (1996) . Realtime Reinforcement Learning for a Real Robot in the Real Environment, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1321-1328, 0-7803-3213-X, Japan, 1996

Lee, W.; Hallam, J. & Lund, H. H. (1997) Apply Genetic Programming to Evolve Behavior Primitives and Arbitrators for Mobile Robots. Proceedings of IEEE 4th International Conference on Evolutionary Computation. IEEE Press, pp. 501-506, 0-7803-3949-5, 1997