An On-Line Learning Method for Object-Locating Robots using Genetic Programming on Evolvable Hardware

Ho-Sik Seok, Kwang-Ju Lee, Je-Gun Joung and Byoung-Tak Zhang
Artificial Intelligence Lab (SCAI)
Dept. of Computer Engineering
Seoul National University
Seoul 151-742, Korea
{hsseok, kjlee, jgjoung, btzhang}@scai.snu.ac.kr

Abstract

Evolvable hardware is a new concept of FPGA which has a capability of dynamic reconfiguration during run time. Due to its dynamic reconfiguration ability, evolvable hardware can optimize itself through learning. In this paper, we present a method for learning robot controller on evolvable hardware. For learning, we employ genetic programming. Typically, genetic programming uses tree-structured representation. However, tree structures are inconvenient for crossover in hardware and tend to consume much resource. Therefore, we use a linear chromosomes to represent genetic trees on evolvable hardware. The learning objective of the robot is to locate a light source avoiding obstacles.

1 Introduction

Evolvable hardware is a new concept of FPGA (Field Programmable Gate Array) which has a capability of dynamic reconfiguration during run time. Due to its dynamic reconfiguration ability, evolvable hardware is applicable to many areas such as fault-tolerant systems and adaptive systems [1]. Various evolutionary algorithms have been used as evolutionary mechanisms for evolvable hardware. Many researchers used binary-string genetic algorithms. Due to their advantage that the configuration bitstreams can be regarded as chromosomes. Alternatively, genetic programming can be used for evolving codes on evolvable hardware.

Since genetic programming usually uses tree structures and trees are flexible, genetic programming is appropriate for evolving complex-problem solving strategies [2]. We attempt to combine the advantages of evolvable hardware and genetic programming. Robot behavior can be described as a mapping from sensor inputs to motor outputs. Therefore, after determining the inputs and outputs of a robot, genetic programs can be used to represent the mapping. Through reordering nodes of the genetic tree, the control structure of the robot can be adapted.

By combining genetic programming and evolvable hardware, we were able to construct a robot controller which can be adapted to environmental changes. In our approach, robot controller evolves its control structure using environmental data on evolvable hardware.

The paper is organized as follows. Section 2 reviews related work. Section 3 describes implementation details. In Section 4, some experimental results are shown. Section 5 summarizes the result and points out some future work.

2 Related work

There are many attempts to control an autonomous robot by genetic programming. Ebner evolved the control structure of a real mobile robot using genetic programming [3]. Wilson evolved hierarchical behaviors to locate a goal object in a maze for a mobile robot [4].

Some authors have attempted to evolve emergent collective behaviors using genetic programming. Bennett III used genetic programming to evolve a common program that controls foraging for food by ants [5]. Zhang introduced a framework, called fitness switching, that facilitates evolution of composite emergent behaviors of a multiagent system using genetic programming [6]. All these works have used software genetic programming as an evolutionary engine.

Genetic algorithms are frequently used as a mechanism for evolving hardware circuits. Here, configuration bitstreams of evolvable hardware are represented as chromosomes. Keymeulen used a genetic program-

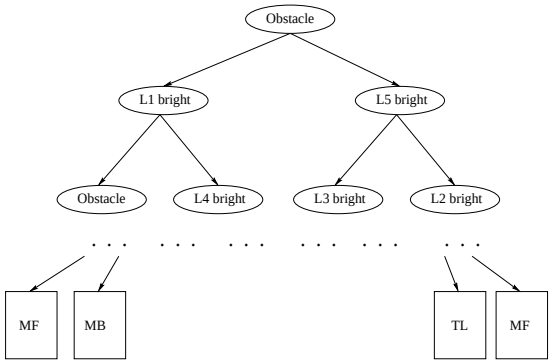

Figure 1: An example of a genetic tree. This tree represents a general control structure of an autonomous robot. The meaning of the left subtree is "if an obstacle is found, light sensor L1 is bright, and other conditions are met, then move forward (MF)."

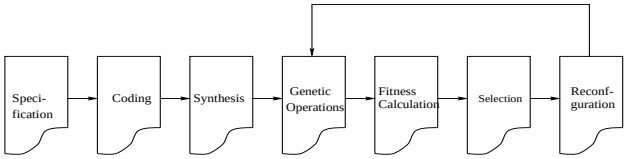

Figure 2: The procedure for hardware evolution. Starting from a population of initial circuits, the circuits are improved by repeating the evolutionary design steps.

ming algorithm for building a navigation system for an autonomous robot [7]. Thomson used genetic algorithms to design electronic circuits automatically [8].

Genetic programming has also been used for evolvable hardware. Sakanashi applied genetic programming to digital circuit design: Evolution of binary decision diagrams. He used genetic programming to improve the hardware description in binary decision diagrams [9].

3 Hardware evolution of genetic trees

3.1 Robot control using genetic programming

After specifying each sensor's input range and robot's output actions, the elements that determine the input-output of control programs are easily obtained. By transforming these elements to corresponding nodes of genetic tree, the preparation for genetic programming representation of control programs is completed.

Fig. 1 shows a control structure of a robot that finds a light-source while avoiding obstacles. This control program interprets environmental data as follows. First, it determines if there is an obstacle. Then, it goes to one of the two subtrees. At the subtree below the root node, it determines which light sensor directs to the light source. By repeating above procedure, it reaches one of the terminal nodes. Then, the terminal node selects one of the possible actions of the robot. The objective of genetic programming is to find an optimal tree structure that controls the robot to the target position.

Fig. 2 shows the procedure for hardware evolution. First, the designer specifies the circuit that he wants. Second, the designer writes some initial code. Third, the design is synthesized by using a commercial CAD program. Then, the following evolutionary steps are repeated until a termination condition is satisfied: genetic operation, fitness calculation, and selection.

3.2 Hardware evolution of genetic trees

While there have been several attempts to implement bitstring genetic algorithms on evolvable hardware, relatively few attempts have been made to implement genetic trees on evolvable hardware. The usual GP tree structures can directly be represented on evolvable hardware as shown in Fig. 3. But, in this approach, the designer faces several problems that do not occur in software genetic programming. First, due to partial tree stucture, only restricted crossover is allowed. Second, a dominant part of hardware resources are consumed by routing. Third, a significant part of hardware resource is never used for placement nor for routing. To overcome these problems, we use a linear representation scheme.

We use linear strings to represent trees on evolvable hardware. To represent a genetic tree in binary strings, we separate each path from the root node to each terminal node, then each path is transformed to a binary string. Binary strings are of the same length. By using a linear representation, several advantages can be obtained. First, crossover operator can easily be implemented. Second, we can save hardware resources for routing. Third, we can increase the total resources utilization ratio. In our preliminary experiments, we were able to save the resources for routing by 12%, for example.

The terminal and function nodes we adopted are shown in Table 1. The terminal nodes denote robot actions. These are move forward (MF), move backward (MB), move forward and turn left (MTL) and move forward and turn right (MTR). The function nodes interpret sensor inputs. They consist of two different kinds of IF-statements:

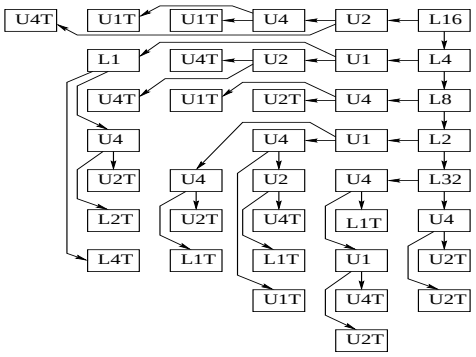

Figure 3: An example of hardware representation of a tree-structured chromosome. This representation scheme does not assure the same depth for all subtrees.

Table 1: Functions and terminals.

	Symbol
Terminal nodes	`MF, MB, MTR, MTL`
Function nodes	`IF-L0, IF-L1, IF-L2`
	`IF-L3, IF-L4, IF-L5`
	`IF-US0, IF-US1, IF-US2`

- `IF-L0` $\sim$ `IF-L5`: the intensity of the light sensors.
- `IF-US0` $\sim$ `IF-US2`: the ultra-sonic sensor values indicating the distance to an object.

4 Experiments and results

4.1 Evolution of robot controllers

The structure of the robot controller is determined through evolution of genetic trees. For efficient evolution, the evolution procedure is divided into two stages. In stage one, reordering of function nodes is performed. The function set consists of 9 function symbols. We use 4-bit binary string to represent each function nodes. Since we allow trees of depth 8, we should maintain 128 binary strings. In stage one, we repeat tree reordering until all interpretation paths of environmental data are obtained. Fig. 4 shows the fitness curves which indicate the advantage of using linear representation. By using linear representation, convergence speed is increased by a factor of approaximately three.

In stage two, proper behavior to each sensor-input is determined, and the evolution procedure is combined

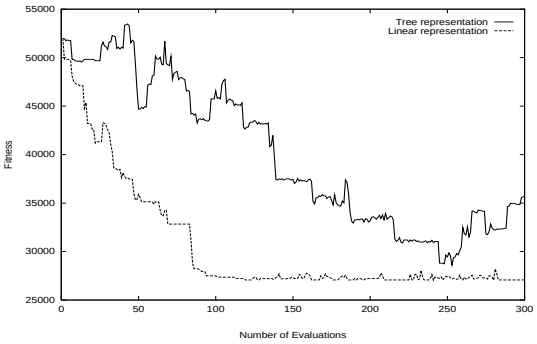

Figure 4: Comparison of experimental results for tree and linear representations. For tree representation, mutation was used as the only genetic operator since crossover caused difficulties in hardware implementation. For linear representation, both mutation and crossover were used.

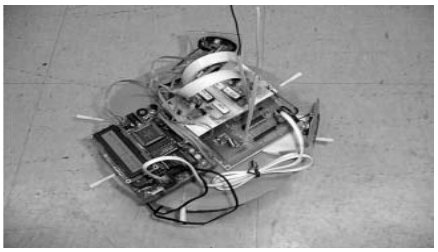

Figure 5: The robot used in the experiment.

with robot move. The result of movement by the previous control structure is used for reconfiguration of the genetic trees.

4.2 Experimental results

The robot has six light sensors for detecting the goal and three ultra-sonic sensors for estimating the distance between the robot and the obstacles. The learning objective of the robot is to locate a light source while avoiding obstacles. The robot is shown in Fig. 5.

Fitness of the robot controller was evaluated as follows:

$$\begin{aligned} Fit(t+1) &= [Fit(t) + (L_{max} - L_{fw})W_L \\ &+ (\frac{2U_{fw} + U_l + U_r}{2U_{max}})k \times W_U \\ &+ Penalty]/2 \end{aligned} \quad (1)$$

Where the meaning of the symbols are $Fit(t)$: Fitness value at step t, L_{max}: Maximum input value of the light sensor, W_L, W_U: Weights of the light and ultra sonic sensors, k: Scaling factor, U_{max}: Maximum input value of ultra sonic sensor, U_{fw}, U_l, U_r: Estimated distance of the forward, left, and right ultra sonic sensor.

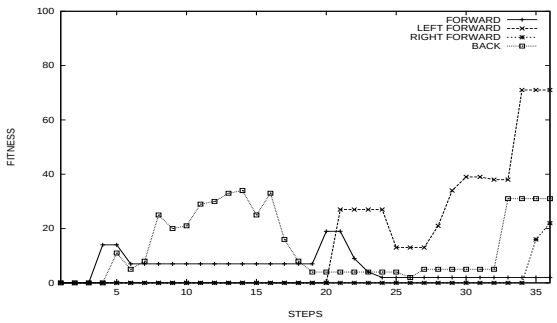

Figure 6: This graph shows the fitness curves about a sensor input when the left light sensor is the most intensified and the forward ultra sonic sensor estimated the longest distance to an obstacle. As known from fitness curves, the action MTL has the largest fitness value.

We assign more weights on the light sensor inputs since the ultra sonic sensor input may be affected by noise. By modifying W_L and W_U, we can change the objective of the robot. By assigning more weight on W_L, the learning objective of the robot is determined as locating the goal.

Fig. 6 shows the learning result of the robot. At 35th generation, the robot found the proper behavior (MTL) for the given sensor input pattern. As generation goes on, the robot finds proper behaviors for each sensor input pattern.

5 Conclusion

We use genetic programming for evolving a robot control structure. For implementation of genetic trees on evolvable hardware, we take a linear representation scheme. Linear representation has several advantages. First, it is easy to implement crossover operation. Second, it uses less resources for routing.

We applied our representation scheme to an object locating robot. Initially, a set of randomly structured genetic trees is created. As robot continues wandering, the robot uses sensor inputs to evolve proper control structures. For efficient evolution, we divided the evolution procedure into the interpretation path stage and the behavior stage.

Due to the limit in hardware resource, we evolved relatively simple control structure. For evolving more complex behaviors, future research on compact representation of genetic trees on evolvable hardware is required.

Acknowledgements

This research was supported by the Korea Science and Engineering Foundation (KOSEF) under grant 981-0920-107-2.

References

[1] L. Paul, "The 'evolvable motherboard': A test platform for the research of intrinsic hardware evolution," *Cognitive Science Research Paper 479*, 1998.

[2] J. R. Koza, "Genetic programming: On the programming of computers by natural selection," Cambridge, MA, USA, MIT Press.

[3] M. Ebner, "Evolution of control architecture for a mobile robot," *International Conference on Evolvable Systems*, pp. 303-310, 1998.

[4] M. S. Wilson et al., "Evolving hierachical robot behaviours," *Robotics and Autonomous Systems*, pp. 215-230, 1997.

[5] F. H. BennettIII, "Automatic creation of an efficient of an efficient multi-agent architecture using genetic programming with architecture-altering operations," *Genetic Programming 1996: Proceedings of the First Annual Conference*, pp. 30-38, 1996.

[6] B-T. Zhang and D-Y Cho, "Fitness switching: Evolving complex group behaviors using genetic programming," *Genetic Programming: Proceedings of the Third Annual Conference*, pp. 431-438. 1998.

[7] D. Keymeulen, "An evolutionary robot navigation system using gate-level evolvable hardware," *International Conference on Evolvable Systems*, pp. 195-209, 1996.

[8] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined with physics," *International Conference on Evolvable Systems*, pp. 390-405, 1996.

[9] H. Sakanashi et al., "Evolution of binary decision diagrams for digital circuit design using genetic programming," *International Conference on Evolvable Systems*, pp. 470-481, 1996.