온라인 적응 학습을 위한 유전자 프로그래밍의 진화 하드웨어 구현

석호식 이광주 장병탁
서 울 대 학 교 컴 퓨 터 공 학 과
{hsseok, kjlee,btzhang}@scai.snu.ac.kr

Implementation of Genetic Programming on Evolvable Hardware for On-line Adaptive Learning

Ho-Sik Seok Kwang-Ju Lee Byoung-Tak Zhang
Dept. of Computer Engineering, Seoul National University

요 약

본 논문에서는 유전자 프로그래밍을 이용하여 온라인 적응 학습이 가능 진화 하드웨어의 진화 전략을 구성하였다. 유전자 프로그래밍은 특유의 트리형 개체구조가 여러 개의 프로세스의 합을 통한 복합 임무의 수행 구조로 해석될 수 있다는 이점에 비하여, 하드웨어 구현이 어렵고 crossover 연산자의 사용이 어렵다는 단점등에 의하여 진화 하드웨어의 동적 재구성 알고리즘으로 널리 사용되지 못하였다. 본 논문에서는 유전자 프로그래밍의 이러한 단점을 극복할 수 있는 개체 표현 및 하드웨어 구현 방법을 제안하였으며, 제안된 방법론에 기존의 연구 결과를 결합하여 유전자 프로그래밍의 수행 효율을 높일 수 있는 진화 전략을 구성하였다. 제안된 진화 전략은 자율 이동 로봇 실험에 적용되어 효율성을 확인하였다.

1. 서론

본 논문에서는 유전자 프로그래밍에 기반하여 개발된 온라인 적응 학습 가능한 진화 하드웨어를 위한 진화 학습 방법에 관하여 소개하고자 한다. 진화 하드웨어는 환경 변화나 임무변경에 맞추어 스스로의 회로를 실행시간 중에 변경할 수 있는 새로운 개념의 하드웨어이다 [1]. 진화 하드웨어는 유전 알고리즘(Genetic Algorithms)이나 유전자 프로그래밍(Genetic Programming)과 같은 진화 알고리즘을 이용하여 동적 재구성을 실시하게 된다. 진화 하드웨어의 진화 방법으로는 유전자 프로그래밍이 선택되었다. 유전자 프로그래밍은 자연선택과 유전 연산자를 이용하여 가장 적합한 프로그램을 찾아가는 진화연산의 한 방법이다 [2]. 트리 구조를 이용한 유전자 프로그래밍의 개체 구조는 여러 스테이트의 결합을 통한 복합 임무 수행에의 적용 가능성을 열어주고 있다. 그러나 트리 구조를 하드웨어상에 표현하는 것은 여러 문제점을 가지고 있다 [3]. 이의 해결을 위하여 본 논문에서는 CSIBS (Context Switchable Identical Block Structure)를 제안한다. 제안 구조에서는 하드웨어의 자원이 허용하는 범위로 트리를 분할하여 해당 실행 시점에서 분할된 서브 트리를 하드웨어상에 표현한다. 하드웨어상에는 동일 구조의 블록들이 존재하며 하드웨어에 표현될 서브 트리의 노드에 맞추어 사전에 정의된 연산의 하나를 표현하게 된다.

논문은 다음과 같이 구성되어 있다. 2 장에서는 진화 하드웨어와 유전자 프로그래밍을 설명하며, 이와 연관하여 제안 구조를 설명한다. 3장에서는 하드웨어 구현 및 실험 결과를 설명하며, 4장에서는 결론 및 추후 연구 발전 방향에 대하여 설명한다.

2. 관련연구

진화 하드웨어란 진화 알고리즘의 제어하에 동적 재구성을 행하는 재구성하드웨어의 일종이다 [4]. 많은 연구자들이 유전 알고리즘을 선호하고 있다. 이것은 첫째 유전 알고리즘은 기반 하드웨어의 Configuration Bit Stream을 개체로 사용할 수 있고, 둘째 유전 알고리즘은 하드웨어 구현이 상대적으로 쉬우면서도 효율성이 검증된 탐색 방법이기 때문이다 [5]. 유전자 프로그래밍을 진화 알고리즘으로 사용하려는 시도가 없었던 것은 아니다. Bennett III는 유전자 프로그래밍을 이용하여 60-decibel OP amp를 진화시켰으며 [6], Sakanashi는 binary decision diagrams의 하드웨어 구현 효율을 높이기 위하여 유전자 프로그래밍을 이용하였다 [7]. 유전자 프로그래밍을 진화 알고리즘으로 이용할 경우, 유전 알고리즘에 비하여 분명한 이점이 있다. 유전자 프로그래밍 트리 구조의 함수 노드는 하드웨어 회로의 함수 블록으로 바로 치환될 수 있다. 이것은 유전자 알고리즘을 사용할 경우 거쳐야 하는 불필요한 탐색 단계-무작위 분포된 Configuration Bit Stream에서의 함수 블록의 진화-를 피할 수 있도록 해준다.

유전자 프로그래밍은 그림 1과 같은 트리 구조를 사용하여 프로그램을 표현한다. 그림 1의 트리는 로봇의 움짐임을 제어하는 프로그램으로 루트 노드에서 말단 노드까지의 경로가 센서 입력의 해석 경로에 해당한다.

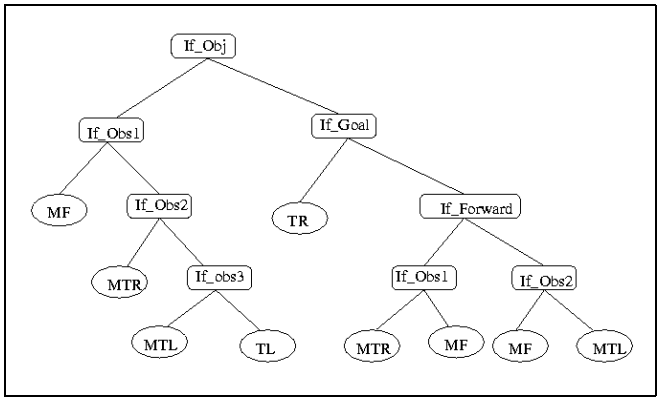

그림 1. 트리 구조로 표현된 프로그램의 모습

유전자 프로그래밍은 다음과 같은 과정을 통하여 원하는 프로그램을 찾는다. 첫째, 후보群을 무작위적으로 생성시킨다. 둘째, 군의 각 개체에 적합도 함수에 맞추어 적합도를 부여한다. 셋째, 부여된 적합도에 따라 유전 연산자의 피연산자가 될 개체를 선택한 후 연산 결과에 따라 군의 구성을 변경하도록 한다. 이 과정의 반복을 통하여 원하는 프로그램을 찾게 된다.

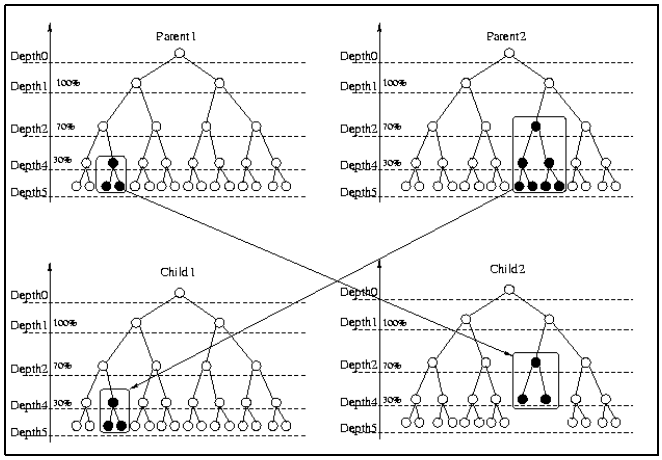

그림 2. Depth-dependant Crossover ([8]의 그림 16.1변경)

유전자 프로그래밍의 주요 연산자는 crossover와 mutation이다. Crossover는 그림 2와 같이 2개의 부모 트리에서 서브트리를 선정 교환함으로써 새로운 개체를 생성시키는 연산자이며, Mutation은 트리를 구성하는 노드를 무작위적으로 다른 종류의 노드로 변경하는 연산자이다. 이 두 연산자 중에서도 적합도 향상에 주로 기여하는 것으로 판단되는 연산자는 crossover이다. 그러나 무자위적으로 crossover point를 선정할 경우, 대부분의 crossover가 트리의 말단 노드 부근에 집중하여 일어날 수 있다는 위험이 있다. 이의 방지를 위하여 본 연구에서는 simple depth-dependant crossover [8]를 사용하였다. Depth-dependant crossover는 그림 2에서와 같이 트리의 depth에 따라 crossover probability를 부여하여, 일정 depth에 crossover point가 집중되는 현상을 막기 위한 연산자이다. Ito의 접근 방법과는 달리, 각 depth의 crossover probability는 변하지 않는다.

적합도 연산은 다음 세대를 구성할 개체를 선택하는 과정에서 선택압을 가하기 위한 연산자이다. 기존의 적합도 연산자를 이용하는 경우 문제가 되는 것은 로봇 행동 제어와 같이 여러 개의 기본 행동이 모아져서 하나의 복합 행동을

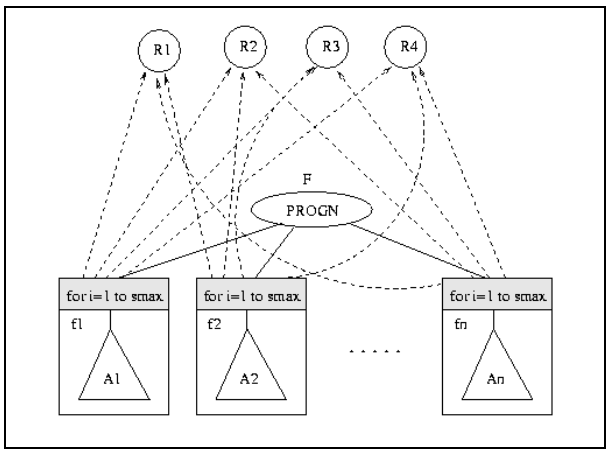

그림 3. Fitness Switching. ([9]의 그림 18,2)

하는 경우이다. 기존의 적합도 연산자로는 이러한 행동을 표현하는 것이 매우 힘들다. 이러한 문제점을 해결하기 위하여 fitness switching이라는 기법을 사용하였다 [9]. Fitness switching에서는 여러 개의 기본 행동에 해당하는 복수개의 적합도 함수群이 존재한다. 순차적으로 해당 적합도 함수를 이용하여 각 기본 행동들을 진화시킨 후 이 행동들을 모아 하나의 복합 임무를 수행하도록 하는 것이 Fitness switching의 기본 아이디어이다. 본 연구에서 응용하고자 하는 자율 이동 로봇 문제에서 필요한 기본 행동은 herding과 homing이다. herding은 밀어야 할 물체를 찾는 단계를 의미하며, homing은 목적지를 향하여 물체를 밀고 나가는 단계를 의미한다. 두 가지 기본 행동의 진화를 위하여 식1과 식2의 2가지 적합도 함수를 이용한다.

$$f_1 : F_{new} = F_{old} + w_1 \times (\#\,collisions) + w_2 \times (\#\,steps)$$

$$f_2 : F_{new} = F_{old} + w_1 \times (\#\,miss) + w_2 \times (\#\,steps) + w_3 \times (line\_vision)$$

3. 구현 및 실험

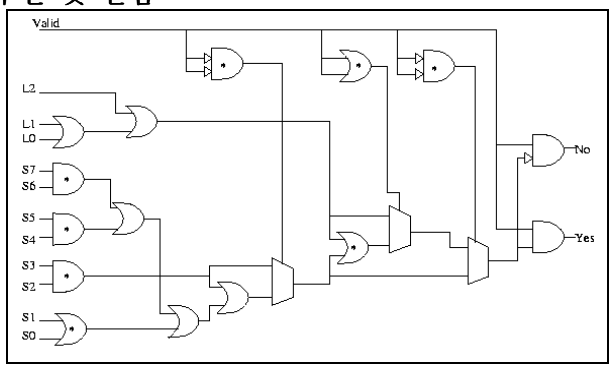

그림 4. 기본 블록도

그림 4는 하드웨어에 구현된 기본 블록을 보이고 있다. 하드웨어상에는 2가지 종류의 기본 블록이 위치하게 되는데, 한 가지 종류는 Function node에 해당하며, 나머지 하나는 terminal node에 해당한다. 하드웨어상에는 그림 4와 같은 블록들이 하드웨어의 자원이 허용하는 한도에서 최대한 분포하고 있게 되며, 블록간의 라우팅을 통하여 유전자 프로그래밍의 트리 구조를 표현하게 된다.

기본 블록에서 (*)로 표현된 gate를 변경하여 해당 function이나 행하여 할 운동을 표현하도록 한다.

Function node는 실험에 사용한 Khepera robot의 센서 체계에 맞추어 7가지로 구분되며 센서 정보 해석을 담당한다.

- *If_OBJ*: 로봇이 밀어야 할 물체에 접근하였는지 판단
- *If_GOAL*: line vision에 목적지가 포착되었는지 여부 판단

•*If_FORWARD*: 목적지-물체-로봇의 순으로 나열되었는지 여부 판단
•*If_OBS1~4*: 4 그룹의 광센서 중 장애물을 포착한 센서 그룹이 있는지 여부 판단

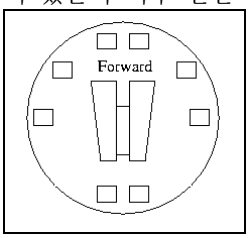

그림 5. Khepera Robot의 센서 체계:
1개의 line vision과 8개의 IR센서로 구성

로봇의 운동에 해당한 terminal node 는 다음의 7 종류로 구성된다. *MF: Move Forward, MTL: Turn Left & Move Forward, MTR: Turn Right & Move Forward, MB: Move Backward, RANDOM: Random move, TL: Turn Left, TR: Turn Right*.
상기한 7 개의 function node 와 7 개의 terminal node 를 구성하여 그림 6 과 같은 환경에서 장애물을 피하면서 물체를 목적지까지 밀고 가는 로봇의 제어 논리를 찾아내는 것이 본 연구의 목적이다.

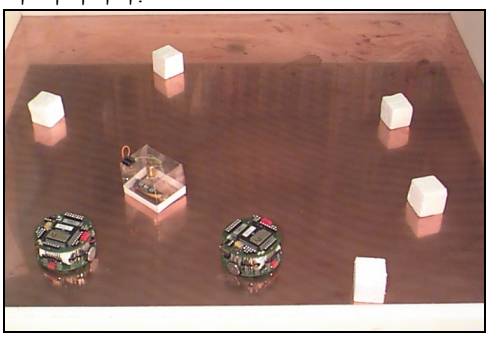

그림 6. 로봇의 실험 환경

실험에 사용한 진화 하드웨어는 XILINX 사의 XC6216 FPGA 이며 V.C.C.사의 H.O.T. (Hardware Object Technology) 보드에 장착되어 있다. 실험 보드는 host PC 의 PCI slot 에 꽂혀 있으며, 로봇과는 serial communication 을 통하여 센서 정보와 해당 센서 정보에 해당하는 운동 정보를 주고 받게 된다.

4. 결론 및 추후 연구

본 연구에서는 범용 프로세서와 주문형 프로세서 (ASIC)의 한계를 극복할 새로운 대안으로 떠오르고 있는 재구성 하우웨어의 동적 재구성 알고리즘으로 유전자 프로그래밍을 이용하여, 온라인 적응 학습이 가능한 진화 하드웨어에 적합한 진화 전략을 제안하였다. 유전자 프로그래밍의 트리 구조 개체는 여러 프로세서의 합을 통하여 복합 임무를 수행할 수 있다는 가능성에 비하여, 진화 하드웨어의 동적 재구성 알고리즘으로 크게 이용되지 못한 것이 사실이다. 이는 트리 구조를 하드웨어상에 표현하는 것이 어렵고, crossover 등의 연산자를 사용하는 것이 어렵다는 유전자 프로그래밍의 약점에 기인한 것이다. 본 연구에서는 이러한 약점을 극복하기 위하여 CSIBS 를 제안하였으며, 주어진 문제를 성공적으로 해결할 수 있음을 확인하였다. 본 연구에서는 문제 공간의 크기가 상대적으로 작은 이유로 Context switching 을 요구하는 개체는 제작할 필요가 없었다.

추후 연구에서는 문제 공간의 크기가 상대적으로 큰 문제를 선정한 후 Context-switching을 통하여 제안한 방법이 실제적인 문제를 해결하는데 도움이 될 수 있다는 사실을 실증하여야 할 것이다.

감사의 글

본 연구는 과학재단 핵심전문 연구(과제 번호 981-0920-107-2)에 의해 지원되었음

5. 참고 문헌

[1] T. Higuchi, et al. Real-world Applications of Analog and Digital Evolvable Hardware, *IEEE transactions on evolutionary computation*, vol3, no 3, pp. 220-235, september 1999.
[2] Koza, J. R., Genetic Programming: On the Programming of Computers by Natural Selection, Cambridge, MA, USA: MIT Press.
[3] Ho-Sik Seok, Kwang-Ju Lee, Je-Gun Joung and Byoung-Tak Zhang, An On-Line Learning Method for Object-Locating Robots using Genetic Programming on Evolvable Hardware, *Proc. of the Fifth Int. Symp. on Artificial Life and Robotics*, pp. 321-324, 2000.
[4] A. Stoica, D. Keymeuler, R. Tawel, C. Salazar-Lazaro, and W. Li, Evolutionary Experiments with a Fine-grained Reconfigurable Architecture for Analog and Digital CMOS circuits, *Proc of the First NASA/DoD Workshop on Evolvable Hardware*, pp. 76-84, 1999.
[5] M. Perkowski, A. Chebotarev, and A. Mishenko, Evolvable Hardware of Learning Hardware? Induction of State Machines from Temporal Logic Constraints, *Proc of the First NASA/DoD Workshop on Evolvable Hardware*, pp.129-138, 1999.
[6] F. H. Bennett III, J. R. Koza, D. Andre, and M. A. Keane, Evolution of a 60 Decibel OP Amp using Genetic Programming, *First Int. Conference on Evolvable Systems*, pp. 455-469, 1996.
[7] H. Sakanashi, T. Higuchi, H. Iba, and Y.Kakazu, Evolution of Binary Diagrams for Digital Circuit Design Using Genetic Programming, *First Int. Conference on Evolvable Systems*, pp.470-481, 1996.
[8] T. Ito, and S. Sato, A Self-Tuning Mechanism for Depth-Dependent Crossover, *in Advances in Genetic Programming 3*, MIT Press, pp. 377-399, 1999.
[9] Byoung-Tak Zhang and Dong-Yeon Cho, Fitness Switching: Evolving Complex Group Behaviors Using Genetic Programming, *Proc. of Genetic Programming 1998*, pp. 431-438, 1998.