# 하드웨어 진화를 위한 Niching 방법

이재성[0] 황금성 조성배
연세대학교 컴퓨터과학과
{ritsz, yellowg}@candy.yonsei.ac.kr, sbcho@cs.yonsei.ac.kr

# Niching Methods for Evolvable Hardware

Jae-Sung Lee[0] Keum-Sung Hwang Sung-Bae Cho
Dept. of Computer Science, Yonsei University

**요 약**

진화 가능한 하드웨어는 사용 환경에 대해 유연하며, 목표 하드웨어에 대해 다양한 최적해를 가질 수 있는 특성이 있다. 하지만 일반적인 유전 알고리즘을 적용할 경우 이러한 특성을 충분히 살리기 힘들기 때문에 niching 방법을 적용하면, 해공간에서 보다 넓은 범위의 해를 탐색해 낼 수 있으며, 일정 수준 이상의 다양성을 유지 할 수 있을 뿐만 아니라 보다 빠르게 최적해에 접근할 수 있다. 실험결과 clearing 알고리즘이 보다 우수한 탐색능력을 보여주었다. 본 논문에서는 적합도 공유와 clearing을 통한 niching 방법을 진화 하드웨어에 적용하고, 실험을 통해 그 성능을 비교한다.

## 1. 서 론

진화 가능한 하드웨어(Evolvable Hardware : EHW)는 환경과 상호 작용하여 동적으로 자신의 구조와 행위를 변경시킬 수 있는 하드웨어를 말한다[1]. 이러한 EHW는 1990년대 초반 FPGA(Field Programmable Gate Array)와 같은 사용자 조작에 의해 재구성이 가능한 하드웨어의 출현과 함께 관심을 모아 왔다. EHW는 환경에 대처하여 목적하는 바를 달성하기 위해서, 하드웨어 자신이 환경에 적응하여 스스로 하드웨어 구성을 변경시켜 나가는 것이 가능하기 때문에 복잡한 회로 설계에 있어서 인간이 직접 설계하는 것보다 빠른 결과를 보여주고 있으며, 회로 내의 오류나 고장, 사용환경의 변화에 보다 유연하게 대처할 수 있기 때문에 이에 대한 연구가 활발히 이루어 지고 있다. 이에 따라 EHW의 적용 범위는 보다 다양해져 각종 회로 및 시스템 설계, 로봇 제어, 패턴인식, 고장극복 시스템 설계 등의 여러 분야에 적용 되어지고 있으며 그 영역은 계속 확대되어 가는 추세이다.

대부분의 EHW는 현재 환경에 맞는 구조와 행위를 탐색하기 위하여 진화 알고리즘(Evolutionary Algorithm : EA)을 사용한다[1]. 생물이 자연에 적응하여 스스로를 변화시켜가는 것과 마찬가지로 EHW도 환경에 맞추어 자신을 재구성함에 있어 생물의 진화와 같은 면을 보이기 때문이다.

본 논문에서는 진화 하드웨어를 구현하기 위해 유전 알고리즘(Genetic Algorithm : GA)을 사용한다. 하지만, 기존의 유전 알고리즘(simple Genetic Algorithm : sGA)의 경우, 하나의 최적해를 탐색하는 문제에 있어서는 우수한 결과를 보여주고 있으나, 여러 개의 최적해를 가지며, 사용 환경에 대해 유연한 특성을 갖는 EHW에 적용하기에는 적합하지 않다.

이러한 EHW의 특성에 맞추어 다양한 최적해를 탐색하기 위하여 niching 방법을 사용한다. niching 방법을 GA에 적용함으로 genetic drift를 감소시키고[2], GA로 하여금 여러 개의 다양한 해를 유지하도록 할 수 있다[3].

적합도 공유(Fitness Sharing)는 가장 잘 알려진 방법으로 Holland[4]가 제안하여, Goldberg와 Richardson에 의해 개량되었으며[5], 이미 EHW 분야에 적용되어져 왔다[6]. 하지만, 다중해에 보다 우수한 niching 방법으로 알려진 clearing[2,7]에 대하여 EHW에 적용한 예는 없었다. 본 논문에서는 clearing을 EHW에 적용하여 보다 넓은 영역에서 해를 탐색하여 보다 빨리 목적 하드웨어에 도달할 수 있도록 하였다.

## 2. 배 경

### 2.1. 유전 알고리즘

유전 알고리즘은 자연 선택과 자연 발생의 기계적인 과정에 기초한 탐색 알고리즘이다. 일반적으로, sGA는 독립적인 개체들로 이루어진 모집단(population)을 생성하고, 선택, 교차, 돌연변이 등의 연산자를 통해 개체들을 진화시킨다. 선택은 자연 선택의 인공적 형태로의 변환이라 할 수 있는데, 집단에서 주어진 환경에 좀더 적합한 개체들이 선택되어 다음 세대에까지 살아 남을 수 있다. 교차는 선택된 개체들에서 임의로 두 개체를 선택하여 그 중 일부분을 교환하는 것이다. 돌연변이는 임의로 선택된 개체의 염색체 중 일부분을 바꾸는 것이다. 이후 새로이 생성된 모집단에 대해서 각 개체에 대한 적합도를 평가하면서 일정 조건이 이루어질 때까지 반복한다.

### 2.2. 적합도 공유

조기에 해에 수렴하거나 genetic drift 현상이 일어 나는 것을 방지하고, 해의 다양성을 유지하는 방법으로 다음과 같은 공유 적합도를 사용한다.

$$sf_i = \frac{f_i}{\sum_{j=1}^{n} sh(d_{ij})}, \qquad (1)$$

이때, $f_i$는 각 개체의 적합도이고, $n$은 개체수이며, $sh(d_{ij})$ 는

공유함수이다.

$$sh(d_{ij}) = \begin{cases} 1 - \dfrac{d_{ij}}{\sigma_s}, & for\ 0 \le d_{ij} < \sigma_s \\ 0, & for\ d_{ij} \ge \sigma_s \end{cases} \quad (2)$$

여기서 $d_{ij}$는 개체 $i$와 $j$간의 거리이고, $\sigma_s$는 환경변수로서 공유반경(sharing radius)이다.

### 2.3. Clearing

clearing은 동일한 subpopulation 내의 개체중 가장 우수한 개체의 적합도를 보존하는 방법으로[8], 각 개체들의 적합도 계산 이후 선택 연산이 이루어지기 전에 적용된다. 기본적인 알고리즘은 아래에 제시된 유사코드와 같다.

```
Function Clearing (Sigma, Kappa){
  SortFitness (P)
  for (j=0 ; j<n-1 ; j++) {
    if Fitness(P[j]) > 0 {
      numOfWinners = 1;
      for (h=j+1 ; h<n-1 ; h++) {
        if (Fitness(P[h]) > 0 &&
              Distance(P[j],P[h]) < Sigma) {
          if (numOfWinners < Kappa)
            numOfWinners++;
          else
            Fitness(P[h]) = 0;
}}}}}
```

여기서 P는 모집단, n은 개체수이며, Sigma와 Kappa는 clearing의 환경변수로 각각 clearing radius, niching capacity이다. 기타 사용되어지는 함수는 아래와 같다.

- SortFitness(P) : 모집단 P를 적합도에 따라 정렬
- Fitness(P[i]) : 모집단의 i번째 개체의 적합도 계산
- Distance(P[i],P[j]) : 모집단의 i,j번째의 거리계산

환경 변수인 clearing radius는 적합도 공유와 마찬가지 방법으로 결정되어진다. 따라서, 적합도 공유에서 공유반경을 결정하는 것처럼 반경을 결정하는 것은 여러 실험을 거쳐서 적당한 반경을 결정하여야 한다[2]. niching capacity의 범위는 1에서 개체 수 사이가 되며, 만약 niching capacity가 개체 수와 같은 경우에는 sGA와 같은 성능을 보이게 된다. 하지만 niching capacity가 커질 경우 부집단(subpopulation)의 숫자를 감소시키며, 연산시간을 절약할 수 있으나, 조기에 수렴하여 넓은 영역을 탐색하는데 적합하지 않다[9]. 따라서 niching capacity는 1로 설정하는 것이 가장 좋은 결과를 얻을 수 있다[9].

## 3. 진화 하드웨어

EHW를 구성하기 위해 OLMC(Output Logic Macro Cell) 1개만을 포함한 간단한 GAL(Gate Array Logic) 16V8을 설정하였다. 그리고 OLMC의 기능은 회로조합 기능만을 선택하여 사용하였다. 그림1은 실험 모델이 된 간단한 GAL 회로를 나타낸다.

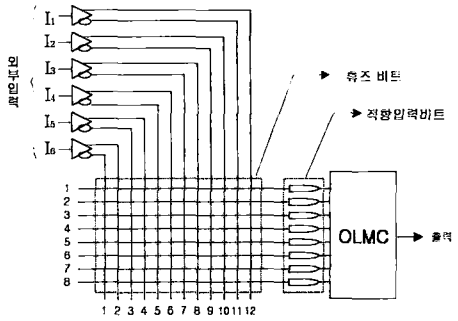


그림 1. 단순화한 GAL 회로

설정된 GAL 칩에 맞게 염색체를 설계하였다. 그림1의 점선영역을 보면, 하나의 OLMC에 들어가는 8개의 입력선의 조합 휴즈 제어부와 입력 허가 여부를 결정하는 8개의 적합입력 제어부가 있다. 이를 제어하는 비트를 유전 코드로 사용하였다. 이 때, 휴즈비트가 '$Ik$ AND not $Ik$'와 같은 조합일 때에는 값이 항상 '0'이 되므로, 이를 배제하기 위해 휴즈비트 2비트를 3치 숫자로 바꿔 표현하였다. 그림 2는 이렇게 표현한 염색체를 보여준다. 여기서 가장 오른쪽의 8비트는 적합입력 제어비트이다.

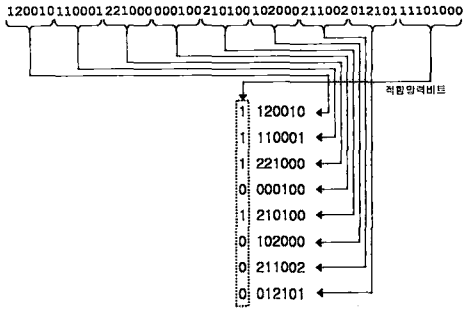


그림 2. 염색체의 예제

개체간 거리는 조합 입력식 별로 계산한 거리를 유클리드 거리(Euclidean distance)로 합산하였다[6]. 조합 입력식 단위의 거리 계산은 유전형 거리의 하나인 해밍 거리(Hamming distance)를 이용하였으며, 전체 거리로의 합산은 아래 수식과 같이 유클리드 거리를 이용하였다[7].

$$d_{ij} = \sqrt{(d_{ij}^1)^2 + (d_{ij}^2)^2 + (d_{ij}^3)^2 + (d_{ij}^4)^2 + (d_{ij}^5)^2 + (d_{ij}^6)^2 + (d_{ij}^7)^2 + (d_{ij}^8)^2}$$

여기에서 $d_{ij}^k$는 $k$번째 조합식에 대한 $i$와 $j$의 해밍거리이다.

## 4. 실험 결과

진화를 위한 환경변수는 개체수 50, 돌연변이율 0.02, 교차율 0.9로 하였고, 적합도 공유와 clearing에 대해서는 공유반경과 clearing 반경을 각각 3으로 하였다.

그림 3은 하드웨어를 2000세대까지 진화시키는 실험을 10회 실시한 결과에 대해 각 세대별 최대 적합도의 평균을

1300세대까지 표시한 것이다.

그림 3에서 알 수 있듯이 sGA와 적합도 공유를 적용한 방법에 비하여, clearing이 가장 빠르게 최적해에 접근하는 것을 관찰할 수 있다. 또한 적합도 공유의 경우 진화 초반에는 해에 대한 접근이 미진하나 진화후반에 이르러서는 sGA에 비해 보다 빠르게 최적해에 접근함을 알 수 있다.

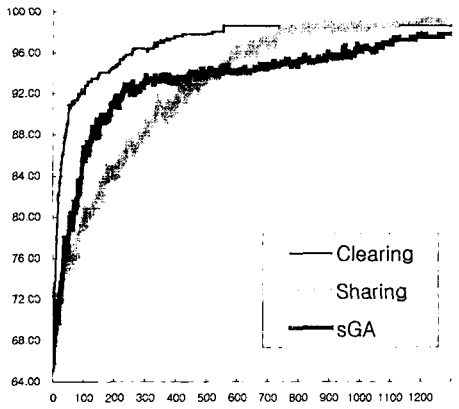


그림 3. 최대 적합도의 세대별 변화(10회평균)

또한, 탐색능력의 비교를 위하여 각 방법을 적용하였을 때 가장 빠르게 최적해 발견한 평균 세대를 비교한 결과는 표1과 같다.

표 1. 최초해 발견 평균 세대 비교

| | clearing | sharing | sGA |
|---|---|---|---|
| 평균 발견 세대 | 413.11 | 906.67 | 693.67 |

clearing이 다른 방법에 비해 평균적으로 해를 발견하는 속도가 빠름을 알 수 있다. 그림4,5,6은 각각 clearing과 적합도 공유를 적용하였을 때와 sGA에 대하여 실험 결과 중 해의 수가 중간인 결과에 대해 single linkage clustering한 덴드로그램(dendrogram)이다.

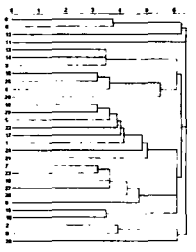

그림 4. Clearing

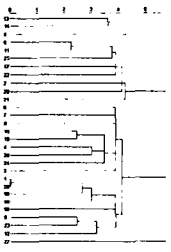

그림 5. 적합도 공유

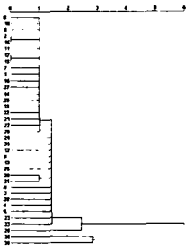

그림 6. sGA

clearing의 경우 발견된 해가 최소한 3이상의 거리를 두고 분포하고 있음을 알 수 있다. 적합도 공유의 경우도 2에서 7의 거리 분포를 나타내고 있어 해가 다양함을 알 수 있다. 하지만 sGA의 경우에는 발견된 해들이 대부분 가깝게 모여 있다. 이는 niching 방법을 사용했을 때에 탐색공간에서 보다 다양한 해를 찾아 낼 수 있음을 의미하고, clearing의 경우, 최소한 환경변수로 주어지는 clearing 반경 이상의 다양성(diversity)을 항상 유지함을 보여준다.

다양성을 유지하게 되면 회로 고장 발생시에도 보다 유연하게 대처할 수 있다[6]. 하지만, sGA와 같은 경우에는 회로 구성 형태가 유사하게 되기 때문에 niching 방법을 적용하여 발견되어진 최적해들에 비해 고장으로부터 덜 유연하다. 따라서 다양성을 유지하고 있는 적합도 공유, clearing를 적용한 경우가 sGA에 비해 EHW에 적합하다.

## 5. 결론 및 향후 과제

본 논문에서는 sGA에 niching 방법들을 적용하는 것이 EHW 특성에 맞는 해를 찾을 수 있다는 사실을 실험을 통하여 알 수 있었다. niching 방법을 적용한 경우에 sGA보다 최적해 탐색시 다양성을 유지하면서 보다 넓은 해 공간을 탐색함으로써, 환경에 적응적이고, 오류에 유연한 목표 하드웨어를 탐색할 수 있다. 또한 clearing 적용시 빠른 속도로 최적해를 탐색하며, 일정수준 이상의 다양성을 유지함을 관찰할 수 있었다. 따라서, EHW 특성에 비추어 볼 때, clearing이 EHW의 진화에 적합한 방법 중의 하나라는 사실을 알 수 있다. 더욱이 기존에 널리 알려진 niching 방법인 적합도 공유에 비해, clearing을 EHW에 적용하였을 경우 우수한 결과를 보임을 확인할 수 있었다.

하지만, 본 실험은 단순한 GAL 회로를 대상으로 이루어 진 것이기 때문에 보다 다양한 회로에 적용하여 연구할 필요가 있다.

## 6. 참고문헌

[1] X. Yao, "Promises and Challenges of Evolvable Hardware," *the 1st International Conference on Evolvable Systems : from Biology to Hardware,* Tsukuba, Japan, 7-8 October 1996.

[2] B. Sareni and L. Krahenbuhl, "Fitness Sharing and Niching Methods Revisited", *IEEE Transactions on Evolutionary Computation,* Vol 2, No 3, pp. 97-105, September 1998.

[3] S. W. Mashfoud, "Niching Methods for Genetic Algorithms," ph.D. dissertation, Univ. of Illinois, Urbana-Champaign, pp62, 1995.

[4] J. H. Holland, Adaptation in Natural and Aritficial Systems, Univ. of Michigan Press, 1975.

[5] D. E. Goldberg and J. Richardson, "Genetic Algorithms with Sharing for Multimodal Function Optimization," *in Proc. 2nd Int. Conf. Genetic Algorithms,* pp.41-49, 1987.

[6] 황금성, 조성배, "종분화를 이용한 다품종 하드웨어의 진화," 정보과학회 봄 학술발표 논문집(B), 제 16권, 1호, p 229-232, 2001.

[7] S. W. Mahfoud, "Niching methods," *Evolutionary Computation 2, Advanced Algorithms and Operators,* institute of Physics Publishing, pp.87-92, 2000

[8] A. Petrowski, "A Clearing Procedure as a Niching Method for Genetic Algorithms," *in Proc. 1996 IEEE Int. Conf. Evolutionary Computation,* Nagoya, Japan, pp. 798-803, 1996.

[9] A. Petrowski, "A New Selection Operator Dedicated to Speciation," *in Proc. Of the 7th Intl. Conf. Of Genetic Algorithms*(ICGA'97)