Proceedings of KIIS Spring Conference 2008 Vol. 18, No. 1.

트리구조 기반 GP 연산자의 구현 및 다양성 분석

Implementation and Diversity Analysis of Tree Structure based Genetic Operators in GP

방철혁, 서기성
Cheulhyuk Pang and Kisung Seo
서울시 성북구 서경대학교 전자공학과
E-mail: {brilliant, ksseo}@skuniv.ac.kr

요 약

본 논문은 GP 트리의 노드포화도를 제어함으로써 트리의 구조공간에서 효율적인 개체 분포를 유도하는 GP 진화연산자를 제안한다. 특정 영역으로의 트리 개체의 분포가 성능에 미치는 영향을 검증하고 진화과정에서 나타나는 군집내의 개체 다양성과의 관계를 분석한다. 제안된 진화연산자를 회귀다항식, 멀티플렉서, 짝수 패리티의 3가지 벤치마크 문제에 대해서 실험을 하였고, 표준 GP 연산자와 비교하였다.

키워드 : Genetic programming, Genetic operators, Distribution of tree structures,

1. 서 론

GP(Genetic Programming)[1,2]는 트리구조를 이용하여 개체를 표현하고, 구조의 가변성을 이용하여 복잡도가 높은 실용적 문제와 최적화 문제에 많은 응용이 이루어지고 있다[2].

개체의 표현에 트리를 이용하게 됨에 따라 발생되는 bloat 코드[3], 교배방식[4], 구조적 어려움[5] 등에 대한 연구가 진행되고 있다.

트리의 구조적인 문제에 대한 Daida의 연구[6-7]에서 발생 가능한 트리 구조의 범위와 실제적으로 해로써 작용하는 트리 구조의 분포 범위는 차이가 있으며, 전체적인 범위 내에서 균등하게 나타나지 않음이 발견된 바 있다. 일반적인 GP 진화연산자는 임의선택에 의한 서브트리 구성으로 인해 효과적인 트리구조를 조합하기 어려운 특성이 있다. 또한 넓은 형태(full-tree)나 좁은 형태(narrow-tree)등의 특정 형태는 구성하기가 어렵다.

본 논문에서는 트리의 구조공간을 기반으로 효율적인 개체 재조합연산을 위해 노드포화도 정보를 이용하는 방법을 제시하고자 한다. 트리개체의 노드포화도 정보를 이용하여 개체를 재조합시킴으로써 구조공간의 영역(I)의 방향으로 생성을 유도할 수 있는 진화연산자를 제안하고자 하며, 진화연산자에 의하여 형성되는 군집을 구성하는 개체의 다양성이 미치는 알고리즘의 탐색능력에 대한 분석적인 접근을 하고자 한다.

새로운 진화연산자의 성능평가를 위하여 3차 이항식(binomial-3) 회귀분석, 멀티플렉서(multiplexer) 그리고 짝수 패리티(even parity) 문제에 대하여 실험한다.

2. GP의 구조적 어려움

GP 진화연산이 가지는 탐색 공간은 트리의 형태, 즉, 깊이와 노드수 두 가지 정보를 이용하여 표현될 수 있고, 임의 선택에 의한 진화연산으로 생성되는 트리 개체집단을 깊이와 노드수 분포로 나타내면 일정한 분포의 패턴을 보여주고 있다[6].

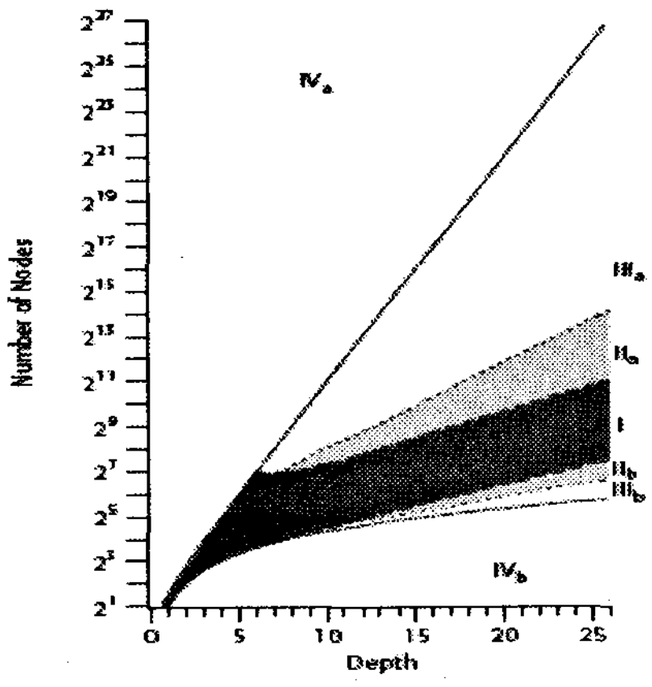

그림 1. 노드 & 깊이 영역
- 저자의 동의하에 [7]에서 인용

트리구조의 분포는 그림1과 같이 크게 4개의 영역으로 나누어지며, 영역(I)에 일반적인 GP 진화연산자에 의한 대부분의 해의 분포가 나타나고, 영역(II,III)으로 벗어나면서 그 분포가 줄어들며, 영역(IV)에는 분포가 존재하지 않는다. 이러한 특성은 트리 해의 구조공간과 분포에 따른 영역이 편중됨을 나타낸다.

본 논문에서는 해로써의 작용 확률이 높은 영역(I)에 대한 개체의 분포 효율을 높일 수 있는 새로운 진화연산자를 제안하고자 한다.

3. 트리 구조 기반 진화연산자

3.1 트리 구조 기반 진화연산자

제안된 트리 구조 기반의 진화연산자의 주목적은 교배와 돌연변이 연산을 통해 재조합되는 개체의 분포 위치를 트리구조 공간 중 영역(I)의 방향으로 생성될 수 있도록 유도하기 위함이다.

이를 위하여 트리가 가지는 깊이와 노드수 정보를 이용하여 노드포화도를 계산하고, 이 정보를 바탕으로 진화연산을 수행하게 된다. 일정 깊이를 가진 트리에 대하여 완전(full) 트리가 될 경우를 1로 하고, 이 값에 대해서 노드가 채워진 비율을 노드포화도로 나타내게 된다.

제안된 진화연산자는 부모 개체의 노드포화도 정보를 통해 부모 개체의 형태를 파악하고, 트리구조 공간 내에서 부모 개체가 존재하는 위치를 파악한다. 선택된 부모 개체에 대하여 진화연산이 수행되어질 위치의 서브트리에 대한 노드포화도를 계산하고, 현재 부모개체의 위치로부터 이동할 진화방향 정보에 기반하여 재구성될 서브트리를 선택하게 된다.

이러한 과정을 통해 트리의 분포가 가장 적합한 영역에 가깝게 자손개체를 생성시킬 수 있게 된다.

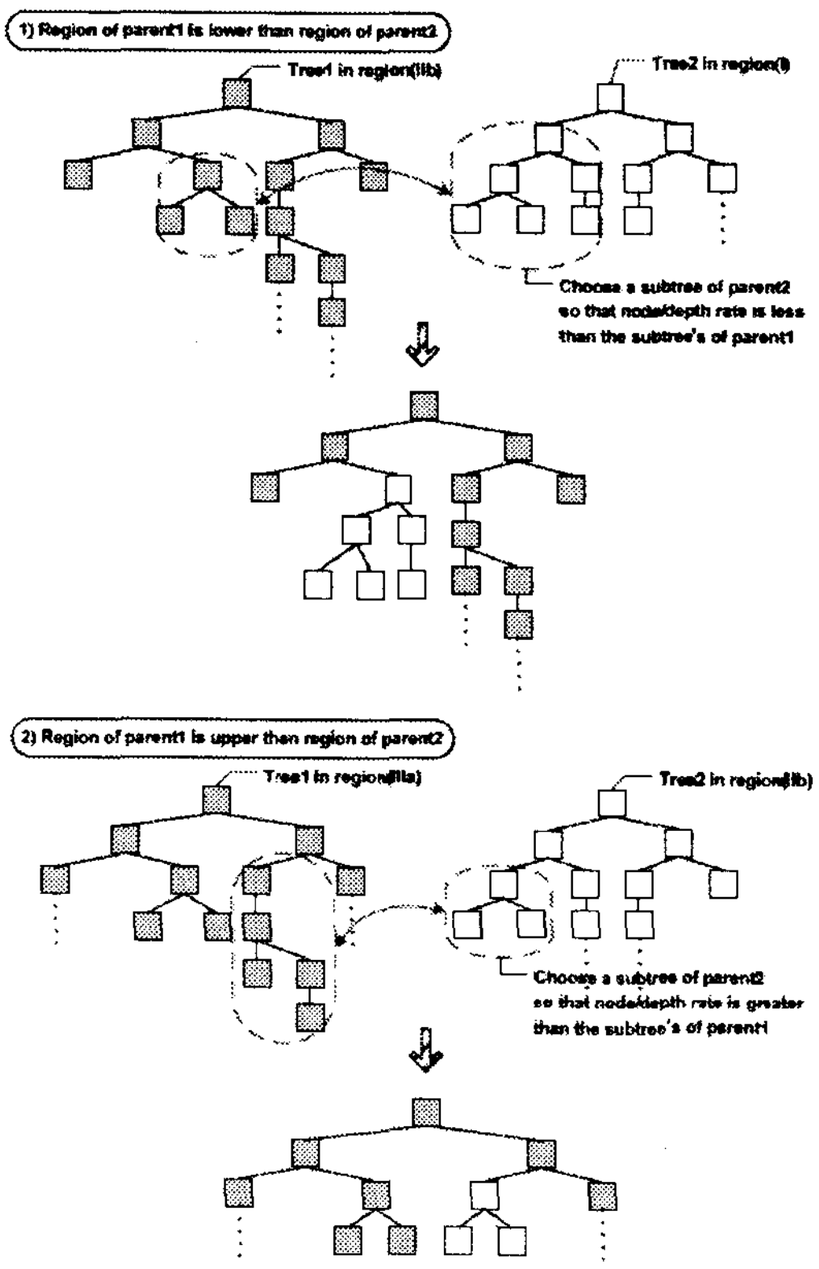

그림 2. 구조기반의 교배연산자

3.2 트리 구조 기반 교배연산자

그림 2와 그림 3은 트리 구조 기반 교배연산자의 알고리즘을 설명하고 있다. 부모의 선택은 최우수(best) 개체와 두 개의 랜덤(random) 개체를 먼저 선택하고, 이 중에서 룰렛방식을 통해 두 개의 부모 개체를 선택한다. 선택된 부모개체의 노드포화도를 통해 구조공간에서의 위치를 파악하고, 진화방향을 결정한다. 부모1 개체에서 랜덤방식을 통해 서브트리를 선택하고, 부모2 개체에서는 부모1 개체의 위치정보를 영역(I)에 위치시킬 수 있는 조건에 부합되는 서브트리 중에서 선택하여 서브트리를 교환하게 된다.

```
Select two parents using roulette wheel selection
Examine a region belonged for each parent
Choose a subtree  at random in parent1
Calculate node/depth rate for the subtree of parent1
If (region of parent1  is
                lower than region of parent2)
  Choose a subtree of parent2
        so that node/depth rate is
                less than the subtree's of parent1
else if (region of parent1  is
                upper than region of parent2)
  Choose a subtree of parent2
        so that node/depth rate is
                greater than the subtree's of parent1
else
  Choose random subtree in parent2
Iterate above process
        again for generation other offspring
```

그림 3. 구조기반 교배 연산자의 수행절차

3.3 거리기반 돌연변이연산자

돌연변이 연산은 부모개체의 구조공간 내에서의 위치에 따라, grow, full 그리고 hal&half 방식을 제한적으로 적용하는 방식을 취하였다.

돌연변이 발생 시 생성되는 서브트리의 포화도를 조절함으로써 영역(I)에 위치시킬 수 있도록 유도한다. 그림 4와 그림 5는 이러한 돌연변이 과정을 설명하고 있다. 부모개체가 영역(I)위에 존재하면 grow 방식으로, 반대면 full 방식을 통해 돌연변이를 발생시키며, 영역(I)의 개체일 경우 half&half 방식으로 돌연변이를 수행한다.

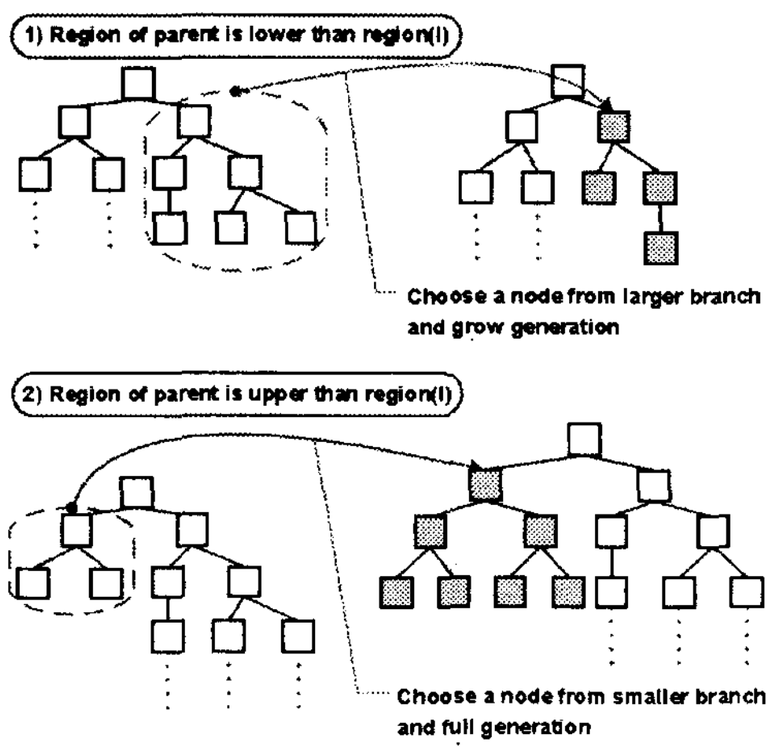

그림 4. 구조기반의 돌연변이 연산자

Proceedings of KIIS Spring Conference 2008 Vol. 18, No. 1.

```
Select a parent using roulette wheel selection
Examine a region belonged for the parent
Calculate node numbers of each branch of parent
If (region of parent is upper than region I)
  Choose a node from larger branch and
            mutate with grow generation
else if (region of parent is lower than region I)
  Choose a node from smaller branch and
            mutate with full generation
else
  Choose a random node and
            mutate with half_and_half generation
```

그림 5. 구조기반의 돌연변이 연산자의 수행절차

4. 실험 및 결과

4.1 실험 조건

트리 구조 기반 진화연산자를 세 가지 벤치마크 문제를 통해 성능평가하고, 기존의 진화연산자와 비교하였다. 3차이항식 회귀문제, 멀티플렉서 그리고 짝수패리티 문제를 이용하였으며, 각 문제 및 난이도 마다 20회의 실험을 통해 그 평균값을 비교하였다.

표 1. GP 수행 파라미터 설정

Number of generations	500 for binomial-3, 2000 to 10,000 for multiplexer, 100 to 7,000 for even-parity
Population size	500
Initialization method	half_and_half
Initialization depths	2-6
Maximum depth	17, (25 for 11-multiplexer)
Crossover rate	0.9
Mutation rate	0.1

lil-GP[8]를 이용하여, 기존의 진화연산자와 제안한 진화연산자를 수행하였으며, 해의 탐색능력과 연산량을 기준으로 비교하였다.

4.2 3차 이항식 회귀분석

첫 번째 실험은 Daida에 의해서 제안된 문제를 이용하였다. 식 (1)을 대상으로 삼았으며 적합도는 구간[-1,0)에서 등간격 50개의 지점 값들을 비교하여 오차'±0.01'을 해로써 정의하였다.

$$f(x) = x^3 + 3x^2 + 3x + 1 \quad (1)$$

함수 집합은 {+,-,×,÷}, 터미널 집합은 {X,R}로 구성하였으며, 실수 R은 [-a,a]의 범위에서 발생하는 난수로 문제의 난이도를 결정짓는 요인으로 사용되었다.

표 2에 실험결과가 나와 있다. 해의 탐색능력(success rate)은 두 진화연산자가 비슷한 성능을 보이고 있지만, 연산량 측면에서 대폭적인 개선이 이루어 졌음을 알 수 있다.

표 2. 3차 회귀문제 실험결과

Regression ($f(x) = x^3 + 3x^2 + 3x + 1$) - 20회 실험 평균값			
Operator	Value range	success rate	No. of Evals
standard crossover & mutation	noERC	100%	4,850.0
	[-1,1]	85%	149,676.5
	[-2,2]	85%	23,794.1
	[-3,3]	85%	116,740.6
	[-10,10]	85%	109,411.8
	[-100,100]	95%	143,000.0
proposed crossover & mutation	noERC	100%	4,552.5
	[-1,1]	95%	70,368.4
	[-2,2]	85%	37,629.4
	[-3,3]	85%	26,968.8
	[-10,10]	90%	65,975.0
	[-100,100]	85%	54,182.4

4.3 멀티플렉서

멀티플렉서 문제는 n 비트 멀티플렉서 기능을 가지는 논리회로를 구성하는 문제이다. 함수집합으로 {and, or, not, if}, 터미널 집합으로 {a0,…,an, d0,…,dn}을 사용하였다.

표 3에 두 실험의 결과가 정리 되어있다. 제안된 진화연산자가 해의 탐색능력과 연산량에서 모두 표준 진화연산자에 비하여 성능이 향상되었음을 확인할 수 있다. 성공률의 경우 표준 연산자가 45%인데 비해 제안된 연산자는 70%의 성능을 보여주고 있다. 특히 연산량에서는 표준 연산자의 성능보다 17%정도의 낮은 연산량만으로 해를 찾아내는 결과를 보여주었다.

11비트 문제의 경우 ADF 없이는 해결하기 어려운 문제로 알려져 있으며, 표준 진화연산자로는 해결하지 못하던 문제를 제안된 연산자를 통해 60%의 확률로 해를 찾을 수 있었다.

표 3. 멀티플렉서 실험결과

multiplexer - 20회 실험 평균값			
Operator	no. of bits	success rate	No. of Evals
standard crossover & mutation	6bit	45%	979,444.4
	11bit	0%	-
proposed crossover & mutation	6bit	70%	164,807.7
	11bit	60%	617,214.3

4.3 짝수 패리티

세 번째 실험은 짝수 패리티 문제로서 함수집합 {and, or, nand, nor} 및 터미널집합 {d0, d1,…,dn}을 이용하였다. 주어진 함수집합과 달리 문제해결에는 EQ(==)와 XOR의 논리연산자가 필요하며, 조합을 통해 구성해야 하기 때문에 문제의 난이도가 높다.

3, 4, 5비트의 입력을 가지는 문제를 테스트 하였으며, 표 4에 비교결과가 나와 있다. 3비트의 경우 두 연산자 모두 해를 찾아내었으나, 연산량을 큰 폭

감소시킬 수 있었다.

표 4. 짝수 패리티 문제 실험결과

even parity - 20회 실험 평균값			
Operator	no. of bits	success rate	No. of Evals
standard crossover & mutation	3bit	100%	35,815.8
	4bit	75%	1,987,333.3
	5bit	0%	-
proposed crossover & mutation	3bit	100%	15,266.7
	4bit	95%	237,631.6
	5bit	75%	2,924,100.0

문제의 난이도가 높아짐에 따라 5비트의 경우 표준 연산자는 해를 찾지 못하게 되었지만, 제안된 연산자의 경우는 75%의 탐색 성공률을 보여주었으며, 중간난이도인 4비트의 결과도 난이도와 비례하여 탐색 성공률과 연산량면에서 제안된 연산자의 성능 개선을 확인할 수 있었다.

5. 진화 연산의 다양성 분석

이상적인 진화연산의 수행은 집단을 구성하는 개체들이 다양한 유전형질을 유지함으로써 진화연산 과정 동안 목적하는 방향으로의 유전형질 조합을 통해 최적해를 생성해 나가는 것이다. 특히, 다양성이 유지되지 못할 경우 지역 최적해에 조기 수렴되는 현상이 발생한다.

그림 6-8에 각 문제의 진화과정에 따른 표준 GP 연산자와 제안된 연산자에 의한 군집 분포의 형태들이 나와 있다.

그림 6은 회귀 다항식 문제에 대한 군집의 분포 결과로, 위쪽의 그래프가 기존의 진화연산자에 의한 군집의 분포를 나타내고 있으며, 각각, 초기, 중기, 말기의 개체 분포상태를 보여주고 있다. 일반 진화연산자의 경우 진화과정이 진행됨에 따라 규모가 작은 트리는 거의 나타나지 않고 있는 모습을 확인할 수 있으며, 특히 제한으로 둔 깊이 17 로 모든 개체가 집중되는 모습을 볼 수 있다. 이러한 수렴현상 쉽게 발생되므로, 최적해의 탐색에 방해를 받게 된다.

그림 6의 아래쪽은 제안된 연산자에 의한 군집의 이동 패턴을 보여주며, 전 세대에 걸쳐 다양한 유전형질을 유지하는 모습을 확인 할 수 있다. 다른 두가지 문제에 대해서도 그림 7과 8처럼 유사한 결과를 보여주므로, 제안된 연산자가 다양성을 유지하는데 큰 영향을 미치는 것을 알수 있다.

6. 결 론

트리의 깊이와 노드수 정보로 표현된 구조공간에서 특정 영역으로의 개체 분포를 제어할 수 있는 새로운 GP 진화연산자를 제안하였다. 세 가지 벤치마크 문제, 3차 이항식 회귀분석, 멀티플렉서, 짝수 패리티에 대하여 제안된 진화연산자를 실험하였고, 기존의 진화연산자에 비해 큰 폭의 성능의 향상을 보였다.

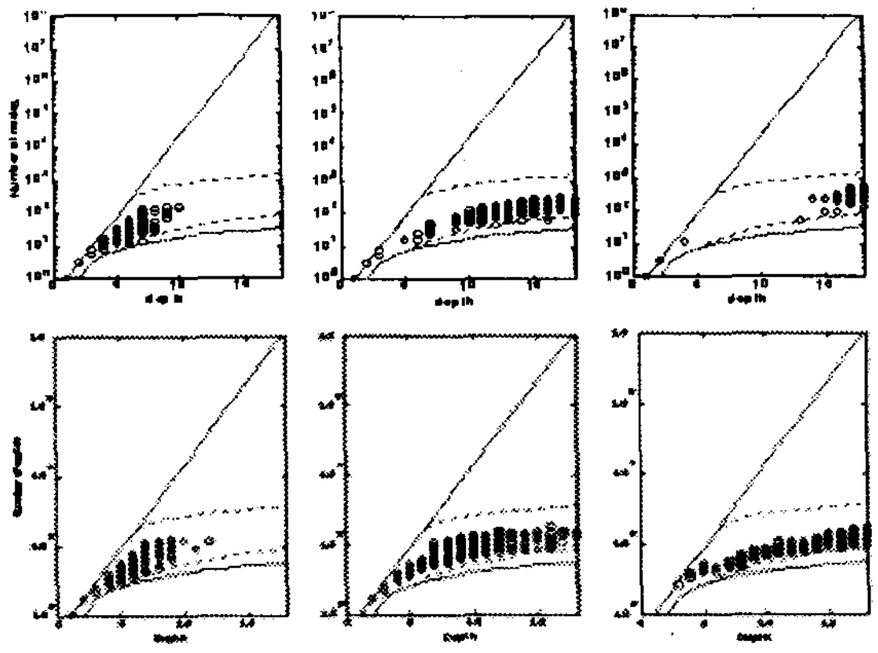

그림 6. 개체 분포 이동 (regression [-100,100])
- standard OP (위), proposed OP(아래)

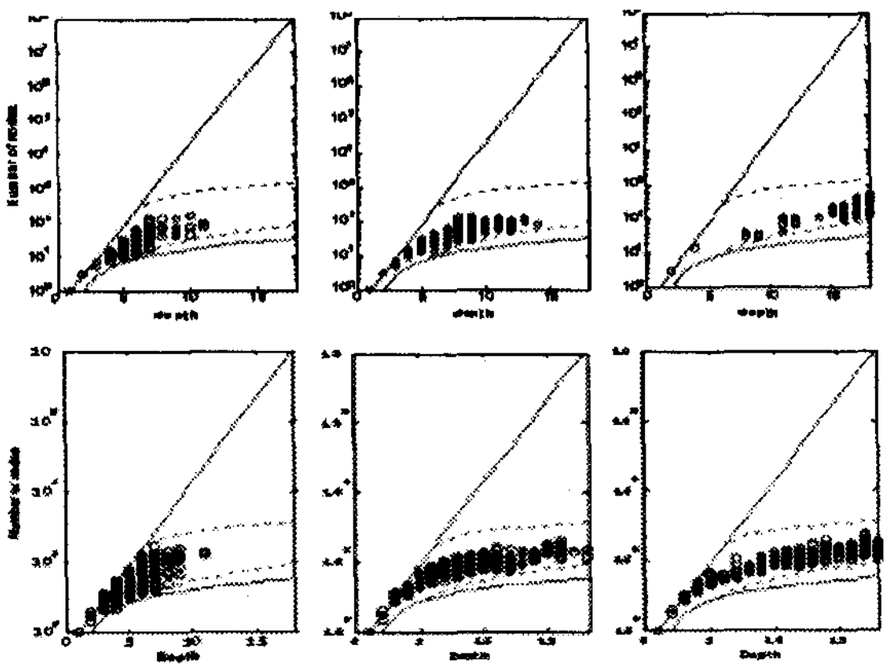

그림 7. 개체 분포 이동 (multiplexer 6 bits)
- standard OP (위), proposed OP(아래)

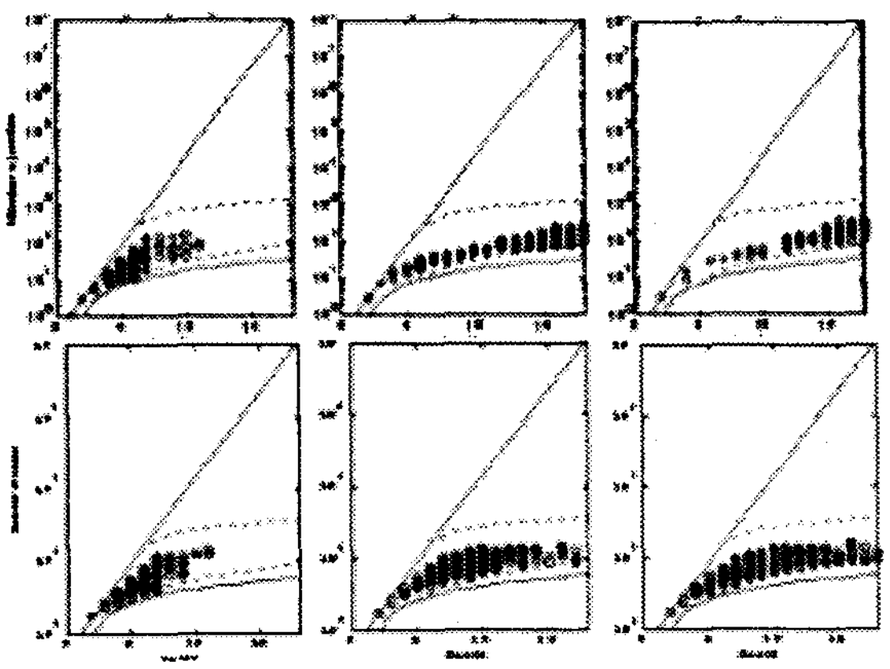

그림 8. 개체 분포 빈도 (even parity 4 bits)
- standard OP (위), proposed OP(아래)

부가적으로 제안된 진화연산자가 진화의 전 과정에서 개체의 다양성을 유지하는 현상을 발견하였고, 다양성의 유지가 진화연산의 효율을 높이고 있음을 확인할 수 있었다.

향후, 해의 분포 특성을 이용하여 제안된 진화연산자의 이론적 고찰 및 다양성 관계에 대한 심화된

분석, 그리고 실용적 문제에 대한 적용이 필요하다.

참 고 문 헌

[1] J. R. Koza, *Genetic Programming : On the Programming of Computers by means of Natural Selection*, MIT Press, Cambridge, MA, USA, 1992
[2] J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, *III, Darwinian Invention and Problem Solving*, Morgan Kaufmann Publishers, USA, 1999
[3] S. Silva, E. Costa, "Resource Limited Genetic Programming : The Dynamic Approach", in Proceeding of the Genetic and Evolutionary Computation Conference (GECCO'05), pp.1673-1680, Washington, DC, USA, June 25-29, 2005
[4] R. Poli, J. Page, "Solving High Order Boolean Parity Problems with Smooth Uniform Crossover, Sub Machine Code GP and Demes", Genetic Programming and Evolvable Machines, Volume 1, Issue 1/2, pp.37-56, April, 2000
[5] Nguyen, X. Hoai, B. McKay, D. Essam, "Representation and structural Difficulty in Genetic Programming", Evolutionary computation, IEEE Transactions on Volume 10, Issue 2, pp.157-166, April 2006
[6] J. M. Daida, J. A. Polito, S. A. Stanhope, R. R. Berttam, J. C. Khoo, "What Makes a Problem GP Hard? Analysis of a Tunably Difficult Problem in Genetic Programming", Genetic Programming and Evolvable Machine, Volume 2, Issue 2, June 2001, pp.165-191, 2001
[7] J. M. Daida, A. M. Hilss, "What Makes a Problem GP Hard? Validating a Hypothesis of Structural Causes", in Proceeding of the Genetic and Evolutionary Computation Conference (GECCO2003), LNCS 2724, pp.1665-1677, Chicago, IL, USA, July 12-16, 2003
[8] D. Zongker, B. Punch, *lil-gp User's Manual*, Michigan State University, July, 1995