유전 프로그래밍을 위한 트리 구조 기반의 진화연산자

Genetic Operators Based on Tree Structure in Genetic Programming

서 기 성*, 방 철 혁
(Kisung Seo and Cheulhyuk Pang)

Abstract : In this paper, we suggest GP operators based on tree structure considering tree distributions in structure space and structural difficulties. The main idea of the proposed genetic operators is to place generated offspring into the specific region which nodes and depths are balanced and most of solutions exist. To enable that, the proposed operators are designed to utilize region information where parents belong and node/depth rates of selected subtree. To demonstrate the effectiveness of our proposed approach, experiments of binomial-3 regression, multiplexer and even parity problem are executed. The experiments results show that the proposed operators based on tree structure is superior to the results of standard GP for all three test problems in both success rate and number of evaluations.

Keywords : distribution of tree structures, genetic operators, genetic programming

I. 서론

GA(Genetic Algorithm)에서 개체의 표현을 비트 스트링을 사용하고 있는 반면, GP(Genetic Programming)[1,2]는 개체의 표현을 트리구조를 통해 표현하고 있다. 트리구조를 이용하여 개체를 표현함으로써 개체의 표현에 가변적인 요소를 가지게 되었고, 복잡하고 실용적인 디자인과 최적화 문제에 많은 응용이 이루어지고 있다[2]. 개체의 표현에 트리구조를 이용하게 됨에 따라 bloat 코드[3,4], 파괴적인 교배방식[5,6], 구조적 어려움[7-9] 등의 문제가 제시되었다.

트리의 구조적인 문제에 대한 Daida의 연구[10-12]에서 발생 가능한 트리 구조의 범위와 실제적으로 해로서 작용하는 트리 구조의 분포 범위는 차이가 있으며, 전체적인 범위 내에서 균등하게 나타나지 않고 있음이 발견된 바 있다. 일반적인 GP 진화연산자는 랜덤선택에 의한 서브트리 구성으로 인해 효과적으로 트리 구조를 조합하기 어려우며, 연산이 진행될수록 불균형한 트리구조를 가지게 된다. 또한 넓은 형태의 트리(full-tree)나 좁은 형태의 트리(narrow-tree) 등 원하는 특정 형태의 구조를 구성하기가 어려운 문제가 있다.

본 연구에서 제안하는 새로운 진화연산자는 이러한 트리 구조의 영역과 해의 분포구역에 대한 특성을 이용한다. 트리개체가 가지는 노드수와 깊이정보를 이용하여 교배연산이 수행되는 서브트리의 노드포화도를 산출한다. 이를 토대로, 교배시에 자손개체의 생성이 Daida가 분류한 트리분포 영역 (I)의 방향으로 유도될 수 있도록 조건에 부합되는 서브트리를 선택한다. 돌연변이의 경우 서브트리의 재생성을 grow, full 그리고 half & half의 세 가지 방식 중에서 트리분포 영역 (I) 로 또는 근접한 방향으로 자손을 생성하도록 선택한다.

본 논문에서는 트리구조를 기반으로 하는 GP 진화연산자를 제안하며, 새로운 진화연산자의 성능평가를 위하여 3차 이항식(binomial-3) 회귀분석, 멀티플렉서(multiplexer) 그리고 짝수 패리티(even parity) 문제[13]에 대하여 실험한다.

* 책임저자(Corresponding Author)
논문접수 : 2008. 3. 28., 채택확정 : 2008. 7. 24.
서기성, 방철혁 : 서경대학교 전자공학과
(ksseo@skuniv.ac.kr/brilliant@skuniv.ac.kr)

II. 트리의 구조적 특성과 GP 진화연산자

1. GP 트리의 구조적인 특성

트리 구조 기반의 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[12]. 트리구조의 분포는 그림 1과 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 점은 트리 해의 구조공간과 분포에 따른 영역이 편중됨을 나타낸다. 즉 대부분의 해가 트리분포 영역 (I)에

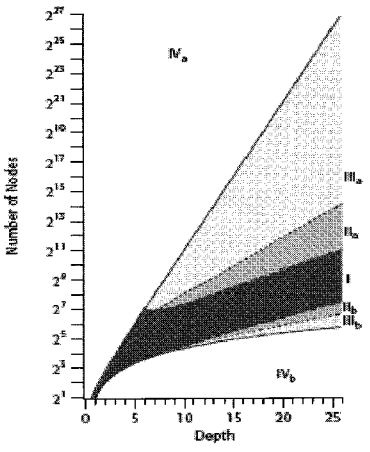


그림 1. 노드와 깊이 영역.

Fig. 1. nodes and depth region - Reprinted with permission from[12].

주로 나타나는 현상이 발견되었다.

이는 횡적으로 치우친 트리나 종적으로 치우친 트리보다는 트리 깊이와 노드수의 비율이 적당한 균형을 가진 트리가 우수해가 될 확률이 높음을 의미한다고 볼 수 있다.

본 논문에서는 이 영역들 중 해의 대부분을 차지하고 있는 영역 (I)에 대한 분포 효율을 높일 수 있는 진화연산자를 제안한다.

2. GP 진화연산자의 문제들

GP의 진화연산자는 다양한 방향에서의 개선이 시도되어 왔다. 교배시 개체의 효율적인 서브트리 교환을 위해서 GA의 uniform 교배기법을 응용하거나[8,9], 트리의 구조 형성에 문법의 체계를 이용하여 개체의 생성을 유도하는 기법[7], 교환되는 서브트리의 깊이정보를 일치시키는 기법[5], 트리 생성에 제한을 두어 bloat code의 발생을 줄이는 방법[3,4]등을 통하여 GP 진화연산의 효율을 높이고자 하였다.

일반적인 GP 진화연산자가 가지는 가장 큰 문제점은 연산자 자체가 개체에 대한 파괴적인 영향이 높다는 점이다. 교배를 통해 재생성 되는 개체가 부모의 유전형질을 가지고 있다고는 하나, 실제적인 유전자형(genotype)과 표현형(phenotype)에 있어서 차이가 많이 나타나게 되는 특징을 가지고 있다.

III. 트리 구조 기반의 진화연산자

1. 트리 구조 기반의 진화연산자

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

이를 위해 먼저 다음과 같이 노드 포화도를 정의한다. 노드 포화도는 일정 깊이를 가진 트리가 완전(full) 트리가 될 경우를 1로 하고, 이 값에 대해서 노드가 채워진 비율을 나타낸다.

제안된 진화연산자는 부모 개체의 노드포화도 정보를 통해 부모 개체의 형태가 넓은(wide)형태인지, 좁은(narrow)형태인지를 판별한다. 개체의 형태를 통해 트리구조 공간 내에서 부모 개체가 존재하는 위치를 파악한다. 그런 다음, 진화연산을 수행하기 위해 선택된 노드 위치에서의 노드포화도를 계산하고, 현재 부모의 위치로부터 이동할 진화방향 정보에 기반하여 교배와 돌연변이를 수행할 서브트리를 선택한다. 이를 통해 트리의 분포가 가장 적합한 영역에 가깝게 자손 개체를 생성시킬 수 있게 되게 된다.

2. 트리 구조 기반의 교배 연산자

그림 2와 그림 3에 트리구조기반의 교배연산자의 알고리즘이 설명되어 있다. 부모의 선택은 최우수 개체와 두 개의 랜덤 부모개체를 먼저 선택하고, 이중에서 룰렛방식을 통해 두 개의 부모를 선택한다. 선택된 부모개체의 노드포화도를 통해 구조공간에서의 위치정보를 얻어낸다.

부모1 개체에서는 랜덤방식을 통해 서브트리를 선택하고, 부모2 개체에서는 부모1 개체의 위치정보와 서브트리정

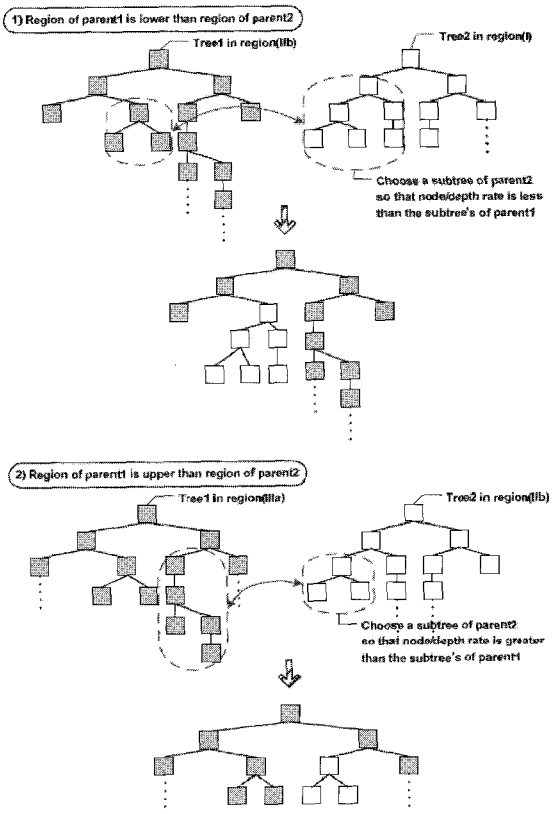


그림 2. 구조기반의 교배연산자.

Fig. 2. Crossover based on tree structure.

```
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 슈도코드.

Fig. 3. Pseudo code of the crossover based on tree structure procedure.

보를 이용하여 생성될 자손 트리를 영역 (I)에 분포시킬 수 있는 서브트리를 선택하게 된다(그림 2).

예로서, 부모1 개체가 부모2 개체보다 상위 영역에 존재하게 될 경우, 부모2 개체의 서브트리 중 부모1 개체에서 선택된 서브트리보다 낮은 노드포화도를 가지는 서브트리들 중에서 교배선택을 하게 된다. 반대로 부모1 개체가 부모2 개체보다 하위 영역에 존재할 경우 부모2 개체의 서브트리 중 노드포화도가 높은 서브트리들 중에서 선택을 하게 된다.

3. 트리 구조 기반의 돌연변이 연산자

돌연변이 연산자도 부모개체의 구조공간 내에서의 분포 위치에 따라, grow, full, 그리고 half & half의 서로 다른 세 가지 방식중에서 적합하게 선택하여 돌연변이 발생 시 생성되는 서브트리의 노드포화도를 조절한다.

그림 4와 그림 5에 트리 구조 기반의 돌연변이 연산자의 알고리즘이 나타나 있다. 돌연변이 발생을 위해 선택된 부모개체의 위치정보를 조사하여 영역 (I) 의 방향으로 유도한다. 즉, 영역 (I)의 위에 존재할 경우 grow 방식으로, 영역 (I)의 아래에 존재할 경우 full 방식을 사용하여 돌연변이 발생을 시키며, 이미 영역 (I)에 위치한 부모개체의 경우 half & half 방식을 이용하여 돌연변이 발생을 하도록 하였다.

부모개체의 위치영역에 따라 돌연변이 방식을 다르게 적용시킴으로써 개체의 노드포화도를 능동적으로 조절하고, 자손 트리 생성을 영역 (I)의 방향으로 유도할 수 있다.

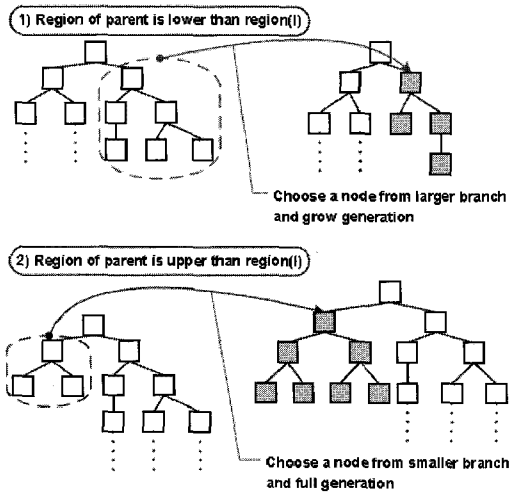


그림 4. 구조기반의 돌연변이 연산자.

Fig. 4. Mutation based on tree structure.

```
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 슈도코드.

Fig. 5. Pseudo code of the mutation based on tree structure procedure.

IV. 실험 및 결과

1. 실험 조건

트리 구조를 기반으로 한 GP의 진화연산자를 세 가지 벤치마크 문제를 이용하여 테스트 하였다. 3차 이항식 회귀문제, 멀티플렉서 그리고 짝수 패리티 문제를 이용하여 표준 GP 연산자와 트리구조기반 진화연산자의 각 조합에 따른 실험을 각각 20회씩 실험하여 그 평균값을 비교하였다.

진화연산자의 조합은 4가지 조건으로, 표준 GP 교배연산자와 돌연변이 연산자, 표준 GP 교배연산자와 트리구조기반 돌연변이 연산자, 트리구조기반 교배연산자만을 사용한 조합, 마지막으로 트리구조기반 교배연산자와 트리구조기반 돌연변이 연산자를 이용한 조합으로써 구성하였다.

각 진화연산자 조합에 대해서 성공률과 연산량을 기준으로 성능을 비교하였다.

트리구조기반 진화연산자는 널리 알려진 GP 프로그램인 lil-gp[14]를 수정하여 구현하였으며, lil-gp가 가지는 표준 진화연산자와 비교 실험하였다. 실험에 사용된 컴퓨터는 Core2Duo 2.13GHz, 2GB의 메모리의 사양을 가지고 있다. 각 벤치마크 문제풀이를 위한 공통적인 파라미터의 설정은 표 1과 같다.

2. 3차 이항식 회귀분석

첫 번째 실험은 식 (1)과 같이 Diada[10]에 의해서 튜닝이 어려운 문제로 제안된 3차 이항식 회귀분석 문제를 대상으로 하였다.

$$f(x) = x^3 + 3x^2 + 3x + 1 \tag{1}$$

이항식에 대한 적합도는 구간 [-1,0)에서 등 간격으로 50

표 1. GP 수행 파라미터 설정.

Table 1. GP 수행 Parameter setting.

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 depth	2-6
Maximum depth	17, (25 for 11-multiplexer)
Crossover rate	0.9
Mutation rate	0.1

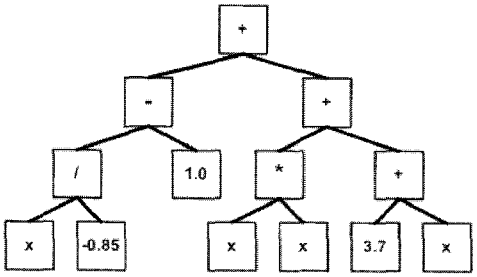


그림 6. 회귀분석 문제에 대한 초기해의 예.

Fig. 6. An example of initial solution for regression problem.

표 2. 제안된 연산자 수행에 의한 트리 분포.

Table 2. Distribution of trees by proposed GP operators.

regression ERC[-100,100] - 총 개체수 500개의 분포						
세대 / 영역	0	5	10	20	50	100
영역 (I)	490	497	497	500	500	500
영역 (II)	10	3	3	0	0	0
영역 (III)	0	0	0	0	0	0

개의 지점 값들을 비교하여 오차 '0.01' 범위 내에 든 지점의 개수를 통해 정의 하였다.

GP 함수 집합은 {+, -, ×, ÷} 으로 구성되며, 터미널 집합은 {X, R} 로 구성된다. 실수 'R'은 [-a, a]의 범위에서 생성되도록 하였으며, 범위 내에서 균등분포에 따라 발생 된다. 문제의 난이도에 따른 각 연산자의 해결 능력 비교를 위해 ERC(ephemeral random constants)의 발생범위를 a = 0, 1, 2, 3, 10, 100의 여섯 단계로 발생시켜 문제의 난이도 차에 따른 진화연산자의 성능을 비교하였으며, a = 0의 경우는 ERC가 존재 하지 않는 경우이다. 참고로, 회귀분석 문제에 대한 초기해의 예가 그림 6에 나와 있다.

표 2에는 제안된 회귀분석 문제에서 GP 연산자에 의한 세대별 개체수의 분포수를 계산한 것으로, 20 세대 이후부터는 모든 개체들이 영역 (I) 에 포함됨을 확인할 수 있다.

실험에 사용된 진화연산자의 조합은 실험조건에서 제시한 네 가지 조합을 이용하였으며, 각 조합별로 여섯 단계의 난이도를 부합하여 각각 20회씩 실험을 실시하였다. 실험결과는 표 3에 나와 있으며, 제안된 진화연산자를 이용한 조합 실험 방법들이 표준 GP 연산자에 비해서 성공률에서는 비슷하거나 약간의 개선을 보였으며, 제일 중요한 연산량 측면에서는 대폭적인 개선이 이루어졌음을 알 수 있다.

그림 7은 ERC[-100,100]의 경우에서 트리구조기반 진화연산자들로 구성된 진화연산의 수행 중 나타나는 개체의 분포 이동을 표현한 그래프 이다. 초기에는 랜덤하게 구성된 500개의 개체 분포가 영역 (I)의 왼쪽 편에 편중되어 나타나며, 소수의 개체가 영역 (IIb)의 경계 부근에 나타나고 있다.

이러한 분포가 진화연산이 진행됨에 따라 160세대에 이르러 해를 찾게 될 때 까지 교배와 돌연변이 연산을 통해 재구성된 개체들의 분포가 오른쪽으로 평탄하게 이동하고

표 3. 3차 회귀 문제 실험결과.

Table 3. Regression problem test results.

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
standard crossover & proposed mutation	noERC	100%	4,875.0
	[-1,1]	100%	79,650.0
	[-2,2]	90%	29,583.3
	[-3,3]	95%	37,526.3
	[-10,10]	95%	69,394.7
	[-100,100]	95%	70,210.5
proposed crossover only	noERC	100%	6,742.5
	[-1,1]	100%	59,840.0
	[-2,2]	95%	39,226.3
	[-3,3]	90%	67,977.8
	[-10,10]	85%	104,161.8
	[-100,100]	85%	85,121.9
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

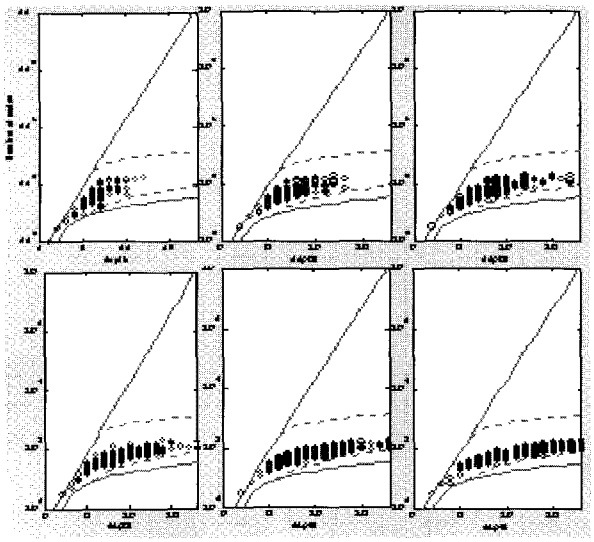


그림 7. 구조기반의 진화연산자에 의한 500개체의 세대별 이동 패턴 분석 (3차 이항식 회귀문제 ERC[-100,100]).

Fig. 7. Distribution of tree structures generated by proposed operators for 500 individuals by generation (binomial-3 regression with ERC[-100,100]).

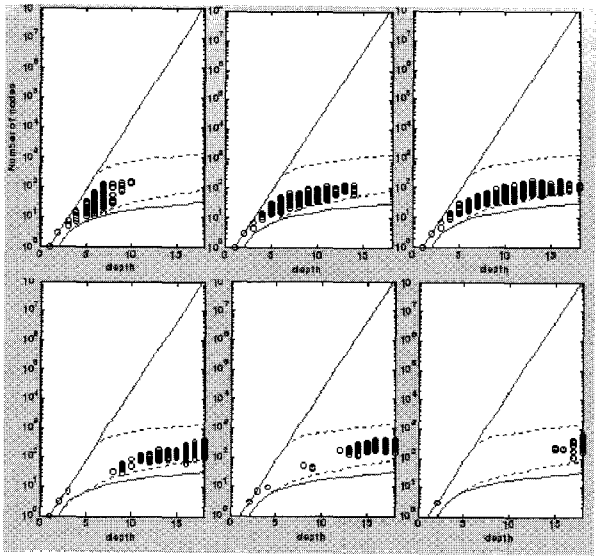


그림 8. 표준 진화연산자에 의한 500 개체의 이동패턴 분석 (3 차 이항식 회귀문제 ERC[-100,100]).

Fig. 8. Distribution of tree structures generated by standard crossover & mutation for 500 individuals by generation (binomial-3 regression with ERC[-100,100]).

있는 모습을 관찰 할 수 있으며, 해를 찾은 마지막 세대에서도 개체들이 다양하게 분포되어 있음을 알 수 있다. 결과적으로 제안된 진화연산자가 제대로 수행되어지는 모습을 확인할 수 있으며, 부가적으로 개체의 다양성도 획득하고 있음을 알 수 있다.

참고적으로, 그림 8은 표준 GP 진화연산자, 즉, 임의선택법에 의한 서브트리 교환 방식과 돌연변이 생성법에 의한 비교 결과를 보여주고 있다. 트리 개체의 이동 및 분포가 세대가 지날수록 편중되며, 마지막 세대에서는 소수의 개체로 수렴되는 현상을 볼수 있다. 겉 모습으로는 일부를 제외한 대부분의 개체들이 영역 (I)에 분포되어 있어서 제안된 진화연산자를 사용치 않더라도 표준 연산자로도 같은 분포를 얻은것 처럼 생각될 수 있지만, 영역 (I)을 벗어난 개체들이 낮은 적합도 값을 가짐으로써 선택에서 배제되어 탈락된 것으로 보이며 이른 개체들이 세대가 지날수록 소수의 개체로 편중되고 수렴되는 현상으로 설명될 수 있다.

3. 멀티플렉서

멀티플렉서 문제는 n 비트 멀티플렉서 기능을 가지는 논리회로를 구성하는 문제이다. 6비트 멀티플렉서의 경우, 두 개의 주소 비트 a0와 a1을 가지고, 데이터 비트가 4개로 총 64개의 입력조합이 존재하는 문제이다. 11비트 문제는 주소 비트가 a0, a1 그리고 a2로써 3개이며, 데이터 비트가 8개로, 총 2038개의 입력조합이 존재하는 문제이다.

함수 집합은 {and, or, not, if}를, 터미널 집합으로 {a0, ···,an, d0, d1, d2, ···, dn}를 각각 사용하였다. 6비트와 11비트 두 가지 문제에 대하여 회귀분석 문제와 동일한 진화연산자 조합을 이용하여 20회 테스트 하였으며 그 결과를 비교 하였다.

표 4에 두 가지 경우에 대한 실험결과가 정리되어 있다.

표 4. 멀티플렉서 실험결과.

Table 4. Multiplexer problem test results.

multiplexer - 20회 실험 평균값			
Operator	no. of bits	success rate	No. of Evals
standard crossover & mutation	6bit	45%	979,444.4
	11bit	0%	-
standard crossover & proposed mutation	6bit	60%	293,892.9
	11bit	5%	4,518,500.0
proposed crossover only	6bit	60%	156,208.3
	11bit	0%	-
proposed crossover & mutation	6bit	70%	164,807.7
	11bit	60%	617,214.3

6비트 멀티플렉서의 경우, 제안된 진화연산자를 이용한 조합에서 해의 성공률과 연산량에서 모두 표준 진화연산자에 비하여 성능이 향상되었음을 확인할 수 있다. 성공률의 경우, 표준 연산자가 45% 인데 비해 제안된 연산자의 조합들은 각기, 60%, 60%, 75% 로 우수함을 보인다. 특히, 제안된 교배&돌연변이 연산자의 경우, 성공률도 75%로 가장 높으

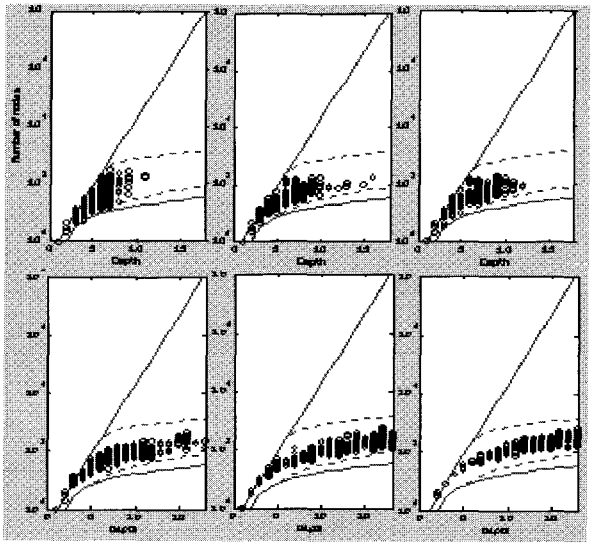


그림 9. 구조기반의 진화연산자에 의한 500개체의 이동패턴 분석 (6비트 멀티플렉서).

Fig. 9. Distribution of tree structures generated by proposed crossover & mutation for 500 in dividuals by generation (6-multiplexer).

면서 연산량에 있어서는 표준 진화연산자의 연산량(약 98만)의 단지 17%정도의 낮은 연산량(약16만5천)으로 해를 찾아내는 결과를 얻고 있다.

11비트 멀티플렉서의 문제(가장 난이도가 높은 문제이며 ADF 없이는 해결하기 어려운 문제로 알려져 있다)에서는 표준 진화연산자로는 20회 모두 전혀 해결하지 못하였으나, 트리구조기반 진화연산자 조합에 의하여 60%의 해 발견율을 보여주었다. 실험 결과, 해 탐색의 높은 성공률과 연산량 면에서 표준 연산자가 6bit 멀티플렉서 문제 해결시에 사용한 98만의 연산량보다도 적은 61만의 연산량을 보여 큰 성능 차이가 있음을 확인할 수 있다.

그림 9는 6비트 멀티플렉서 문제에 대해서 가장 우수한 성능을 보인 트리구조기반 진화연산자에 의한 개체의 분포 이동 그래프로서, 4.2절의 회귀분석 다항식 문제와 유사하게 영역 (I)에 고르게 분포되면서 다양성도 상당히 유지되고 있음을 나타낸다.

4. 짝수 패리티

세 번째 실험은 짝수 패리티 문제로서 표준적으로 제공되는 함수 집합의 제한으로 인해, 문제를 해결하기 위해 꼭 필요한 EQ(=)와 XOR 논리 연산자를 조합하여야 하는 요건 때문에 GP 벤치마크에 문제에서도 난이도가 높은 편이다.

함수 집합으로 {and, or, nand, nor}를, 터미널 집합으로 {d0, d1, d2, ···, dn}를 각각 사용하였다.

3, 4, 5비트의 입력을 가지는 문제에 대하여 각 20회의 테스트를 하였으며, 평균 수치를 비교한 결과가 표 5에 나

표 5. 짝수 패리티 문제 실험결과.

Table 5. Even parity problem test results.

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%	-
standard crossover & proposed mutation	3bit	100%	11,578.9
	4bit	90%	602,500.0
	5bit	20%	2,037,625.0
proposed crossover only	3bit	100%	14,454.5
	4bit	100%	479,575.0
	5bit	30%	2,428,083.3
proposed crossover & mutation	3bit	100%	15,266.7
	4bit	95%	237,631.6
	5bit	75%	2,924,100.0

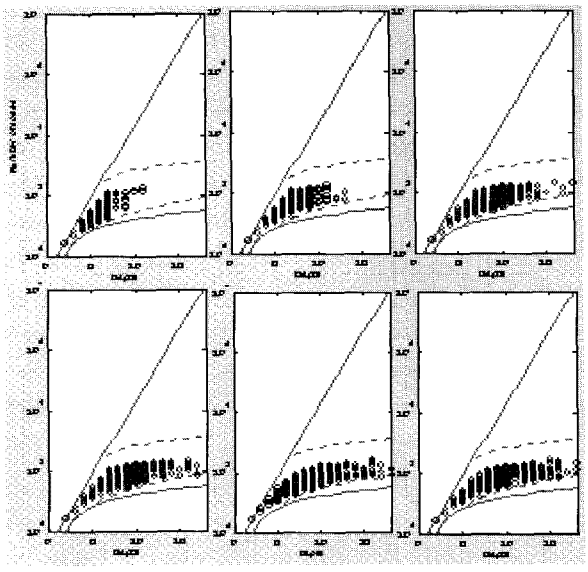


그림10. 구조기반의 진화연산자에 의한 500개체의 이동패턴 분석 (3비트 짝수 패리티).

Fig. 10. Distribution of tree structures generated by proposed crossover & mutation for 500 in dividuals by generation (even 3bit parity).

와있다. 3비트 문제의 경우 모든 비교 방법이 해를 100% 찾아내었으나, 제안된 진화연산자 방법들이 연산량에서 모두 반 이하임을 보여준다.

두 번째, 4비트의 경우는 표준 진화연산자보다 제안된 진화연산자 조합(네번째 방법)이 표준 연산자의 연산량(198만)의 단지 12% 정도의 낮은 연산량(24만) 만으로 문제를 해결하고 있음을 알 수 있다. 해 탐색의 성공률 면에서도 표준 진화 연산자의 75%에 비하여 95%의 높은 성능을 보여주고 있다.

마지막으로, 5비트 문제의 경우에서는 표준 진화연산자의 경우는 전혀 해를 찾지 못하였고, 두 번째 방법인 표준 교배연산자와 제안된 돌연변이 방법의 경우 20%, 제안된 교배연산자만으로는 30% 성공률이었지만, 제안된 교배연산자와 돌연변이 연산자를 같이 사용한 경우 75%의 확률로 문제를 해결하였다.

짝수 3비트 패리티 문제에 대한 진화과정의 트리 개체 분포 형태가 그림 10에 나와 있으며, 앞의 두 문제와 유사하게 분포의 다양성이 존재함을 확인할 수 있다.

V. 결론

GP가 가지고 있는 트리구조의 특성 중에서, 우수 해가 특정 영역에 주로 분포되어 있는 현상에 착안하여, 트리의 분포를 가능한한 특정영역으로 배치시키는 새로운 재조합 진화연산자를 설계하였다. 즉, 교배와 돌연변이 연산을 통해 생성되는 개체의 위치를 Daida의 트리 분포 그래프상의 영역 (I)로 유도하여, 우수해의 발생 확률을 높이고자 하였다.

제안된 트리분포 기반 진화연산자의 검증을 위해 세 가지 벤치마크문제, 3차 이항식 회귀분석, 멀티플렉서, 짝수 패리티에 대하여 실험 하였고, 해 탐색의 성공률과 연산량

을 기준으로 표준 진화연산자의 성능과 비교하였다.

3가지 벤치마크 문제의 결과를 통해, 제안된 트리분포기반 진화연산자의 조합이 표준 진화 연산자보다 해의 탐색 성공률과 연산량 측면에서 대부분 우수하였고, 특히 제안된 교배연산자와 돌연변이 연산자를 조합했을 때가 더욱 높은 성능을 보였다.

부가적으로, 트리 구조 공간위에서 진화과정 중 나타나는 트리 분포 결과를 관찰할 때, 표준 진화연산자에서는 세대가 지남에 따라 개체의 다양성이 급격히 상실되고 있음에 비해서, 제안된 진화연산자에 의한 분포는 마지막 세대까지 해의 분포가 다양성을 유지하는 모습을 확인할 수 있었다.

향후 트리 구조공간에 대한 보다 심층적인 분석을 통해, 현재의 진화 연산자를 개선하는 것과 진화연산자와 다양성과의 유용성 있는 관계를 도출하는 연구가 필요하다고 사료된다.

VI. 참고문헌

[1] J. R. Koza, Genetic Programming : On the Programming of Computers by Natural Selection, MIT Press, Cambridge, MA, USA, 1992.

[2] J. R. Koza, F. H. Bennett, D. Andre and M. A. Keane, III, Darwinian Invention and Problem Solving, Morgan Kaufmann Publishers, USA, 1999.

[3] S. Silva and E. Costa, "Resource Limited Genetic Programming : The Dynamic Approach," *in Proceeding of the Genetic and Evolutaionary Computation Conference (GECCO'05)*, pp.1673-1680, Washington, DC, USA, June 2005.

[4] N. F. McPhee, A. Jarvis and E. F. Crane, "On the Strength of Size Limits in Linear Genetic Programming," *in Proceeding of the Genetic and Evolutaionary Computation Conference (GECCO2004)*, LNCS 3103, pp.593-604, Seattle, WA, USA, June 2004.

[5] T. Ito, H. Iba and S. Sato, "Depth dependent Crossover for Genetic Programming," *in Proceeding of the IEEE*, pp.775-780, Anchorage, AK, USA, May 1998.

[6] H. Majeed and C. Ryan, "On the Constructiveness of Context Avare Crossover," *in proceeding of the Genetic and Evolutionary Computation Conference (GECCO-07)*, pp.1659-1666, London, England, United Kingdom, July 2007.

[7] Nguyen, X. Hoai, B. McKay and D. Essam, "Representation and structural Difficulty in Genetic Programming," *Evolutionary computation*, IEEE Transactions on Volume 10, Issue 2, pp.157-166, April 2006.

[8] J. Page, R. Poli and W. B. Langdon, "Smooth Uniform Crossover with Smooth Point Mutation in Genetic Programming : A Preliminary Study," *EuroGP'99, LNCS* 1598, pp.39-48, Göteborg, Sweden, May 1999.

[9] R. Poli and J. Page, "Solving High Order Boolean Parity Problems with Smooth Uniform Crossover, Sub Machine Code GP and Demes," *Genetic Programming and Evoluvable Machines*, Volume 1, Issue 1/2, pp.37-56, April 2000.

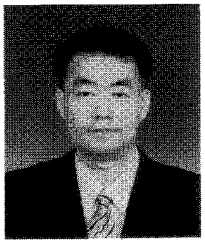
[10] J. M. Daida, J. A. Polito, S. A. Stanhope, R. R. Berttam and J. C. Khoo, "What Makes a Problem GP Hard? Analysis of a Turably Difficult Problem in Genetic Programming," *in proceedings of the Genetic Programming and Evolvable Machine*, ISSN 1389-2576, Volume 2, Issue 2, pp.165-191, Hingham, MA, USA, June 2001.

[11] J. M. Daida and A. M. Hilss, "Identifying Structural Mechanisms in Standard Genetic Programming," *in Proceeding of the Genetic and Evolutaionary Computation Conference (GECCO2003)*, LNCS 2724, pp.1639-1651, Chicago, IL, USA, July 2003.

[12] J. M. Daida and A. M. Hilss, "What Makes a Problem GP Hard? Validating a Hypothesis of Structural Causes," *in Proceeding of the Genetic and Evolutaionary Computation Conference (GECCO2003)*, LNCS 2724, pp.1665-1677, Chicago, IL, USA, July 2003.

[13] S. Luke, "Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat," PhD of University of Maryland, 2000.

[14] D. Zongker and B. Punch, lil-gp User's Manual, Michigan State University, July 1995.



서 기 성

1986년 연세대학교 전기공학과 공학사. 1988년 연세대학교 전기공학과 공학석사. 1993년 연세대학교 전기공학과 공학박사.

1993～1998년 : 서경대학교 산업공학과 조교수

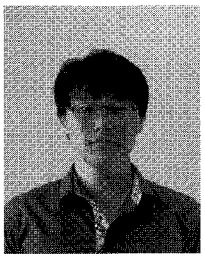
1999～2003년 : Michigan State University, Genetic Algorithms Research and Applications Group, Research Associate.

2002～2003년 : Michigan State University, Electrical & Computer Engineering, Visiting Assistant Professor.

2003～2004년 : 서경대학교 전자공학과 조교수.

2004～현재 : 서경대학교 전자공학과 부교수.

관심분야는 GA, GP, 진화 디자인, 지능로봇.



방 철 혁

2007년 서경대학교 전자공학과 공학사. 2007～현재 서경대학교 전자공학과 석사과정.