Proceedings of KFIS 2002 Spring Conference, 2002. 5. 25

진화 연산의 성능 개선을 위한 하이브리드 방법

A Hybrid Method for Improvement of Evolutionary Computation

정진기, 오세영
포항공과대학교 전자컴퓨터공학부

Jinki Chung, Se-Young Oh
Dept. of Electrical Engineering, Pohang University of Science and Technology (POSTECH)
E-mail : hipearl@postech.ac.kr

ABSTRACT

진화연산에는 교배, 돌연변이, 경쟁, 선택이 있다. 이러한 과정 중에서 선택은 새로운 개체를 생산하지는 않지만, 모든 해중에서 최적의 해가 될만한 해는 선택하고, 그러지 않은 해는 버리는 판단의 역할을 한다. 따라서 아무리 좋은 해를 만들었다고 해도, 취사 선택을 잘못하면, 최적의 해를 찾지 못하거나, 또 많은 시간이 소요되게 된다. 따라서 본 논문에서는 stochastic한 성질을 갖고 있는 Tournament selection에 Local selection개념을 도입하여, 지역 해에서 벗어나 전역 해를 찾는데, 개선이 될 수 있도록 하였고 Fast Evolutionary Programming의 mutation과정을 개선하고, Genetic Algorithm의 연산자인 crossover와 mutation을 도입하여 Parallel search로 지역 해에서 벗어나 전역 해를 찾는 하이브리드 알고리즘을 제안하고자 한다.

Key words : evolutionary programming, crossover, mutation, tournament selection, genetic algorithms

I. 서 론

1960년대부터 다양한 최적화 문제를 해결하기 위하여 자연현상들을 흉내낸 알고리즘들이 개발되어 왔다. 진화 연산(Evolutionary Computation)이란 생물학적 진화 이론을 개념으로 하는 통계적 최적화 알고리즘을 통틀어 말한다. 진화 연산은 크게 유전 알고리즘(Genetic Algorithms: GAs), 유전 프로그래밍(Genetic Programming), 진화 알고리즘(Evolutionary Algorithm), 진화 프로그래밍(Evolutionary Programming)과 진화 전략(Evolutionary strategies)으로 구분할 수 있다. 최적 탐색을 위하여 진화 연산은 탐색 공간의 새로운 영역에 대한 전역 탐색과 현재 개체군에 유용한 정보의 지역 탐색이 균형을 이루어야 한다. 유전 알고리즘의 목표는 해 공간(Solution Space)에서 통계적 전역 탐색의 평균에 의해 최대의 적합도를 가진 개체를 찾는 것이다. 유전 알고리즘은 일반적으로 가중치 공간의 전체적인 탐색을 수행하기 때문에 그 결과 지역 최소값에 빠질 수 없는 반면 한번 찾은 해를 계속적으로 개선해 나가는, 즉 부근을 탐색해서 해의 정밀도를 높이는 능력이 부족하다[1]. 지역탐색과 전역탐색은 분명히 서로 상충되는 특성[2]으로서, 이 둘 사이에 적절한 균형이 필요하다. 이러한 문제를 해결하기 위하여 흔히 유전알고리즘과 다른 지역탐색 알고리즘의 장점을 결합하는 하이브리드 형태의 알고리즘들이 제안되고 있다[3][4]. 하이브리드 알고리즘에서 유전 알고리즘의 역할은 전역 해에 가까이 도달하게 해주고, 지역탐색 알고리즘의 역할은 이점을 초기 점으로 하여 정밀한 해를 찾는 것이다.

본 논문에서는 가능 해에 대한 미세한 조정에 좋은 성능을 나타내는 Neural Network의 Backpropagation을 기반으로 하는 mutation rule과 문제 공간에서 전역 탐색에 좋은 성능을 보이는 GA의 (genetic operator)를 결합하여 많은 문제에 대하여 좋은 성능을 발휘하는

강건한 시스템을 창출하고자 한다.

II. 본 론

많은 문제에 대하여 좋은 성능을 발휘하는 강건한 시스템을 창출하기 위하여 본 논문에서는 GA의 교배, 돌연변이 연산자와 Neural Network의 Backpropagation 개념을 기반으로 하는 3세대 돌연변이 연산자를 융합하는 알고리즘을 제안하였다. 또 그룹에 의한 지역 토너먼트 알고리즘을 선택 알고리즘으로 제안하였다.

제안하는 알고리즘의 전체적인 구조는 다음과 같다.

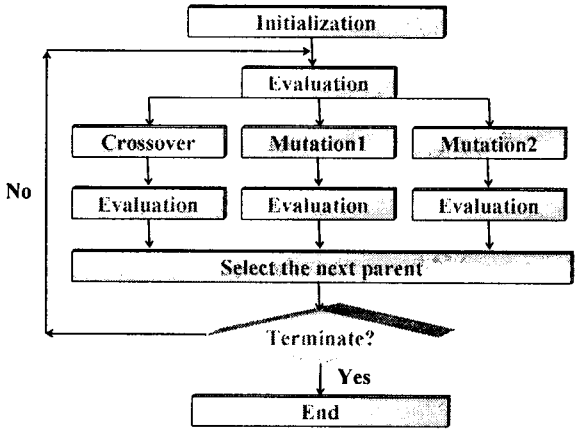

그림1. Flowchart of the proposed algorithm

2.1 GA의 교배 연산자

GA의 교배 연산자는 염색체에 함유된 부분 정보가 개체군에 다른 염색체로 전달되도록 한다. 이러한 정보의 혼합은 최적 염색체가 형성되도록 한다.

본 논문에서는 산술적 교배(arithmetic crossover)[5]를 사용하였다. 이 연산자는 두 부모 염색체를 일차결합(linear combination)하여 자손을 생성하고 있는데 각 요소는 다음 식으로 계산된다.

$$\tilde{x}_j^u = \lambda \tilde{x}_j^v + (1-\lambda)\tilde{x}_j^u$$

$$\tilde{x}_j^v = \lambda \tilde{x}_j^u + (1-\lambda)\tilde{x}_j^v \quad (1 \le j \le n)$$

여기서 λ 는 곱인수(multiplier)로서 고정되거나 아니면 각 요소마다, 염색체마다 독립적으로 결정될 수 있다. 본 논문에서는 λ 값이 [0,1]사이의 난수 값으로 제한되어 발생되는 볼록 교배(convex crossover)를 사용하였다.

2.2 GA의 돌연변이 연산자

탐색 공간을 완전히 탐색하기 위해서는 개체 집단의 다양성이 결정적이다. 유전 알고리즘의 돌연변이 연산자의 목적은 지역 최적 점을 극복하고 전역 최적 점에 도달하기 위해서 충분한 개체군의 다양성을 유지하는 역할을 한다.

본 논문에서는 동적 돌연변이(dynamic mutation)[5][6]를 사용하였다. 일명 불균등 돌연변이(non-uniform mutation)라고도 불리는 이 연산자는 정밀도를 높이기 위하여 미세 조정이 가능하도록 고안된 것이다. j번째 유전자에서 돌연변이가 일어나면 x_j 는 다음 식으로부터 결정된다.

$$x_j = \begin{cases} \tilde{x}_j + \Delta(k, x_j^{(U)} - \tilde{x}_j), & \tau = 0 \text{ 일때} \\ \tilde{x}_j - \Delta(k, \tilde{x}_j - x_j^{(L)}), & \tau = 1 \text{ 일때} \end{cases}$$

여기서 τ 는 0또는 1 둘 중에서 하나를 취하는 난수이다. $\Delta(k, y)$ 로 다음 함수가 이용될 수 있다.

$$\Delta(k, y) = y \cdot r \cdot (1 - \frac{k}{T})^b$$

여기서 r 은 0과 1사이의 실수 난수이고, T 는 알고리즘이 실행되는 최대수이고, b 는 불균등 정도를 나타내는 매개변수로서 사용자에 의해 결정된다. 이 함수의 영역 [0, y]에서 값을 가지는데, 세대 수 k 가 증가함에 따라 0에 가까운 수를 발생할 확률이 높아지도록 접근하게 된다. 이 연산자의 특징은 초기에는 전 공간을 균등한 확률로 탐색하다가 세대수가 증가하면 지역적으로 탐색하도록 해준다.

2.3 Neural Network의 Backpropagation을 기반으로 하는 3세대 돌연변이

제안된 3세대 돌연변이 생성 방식[7]은 부

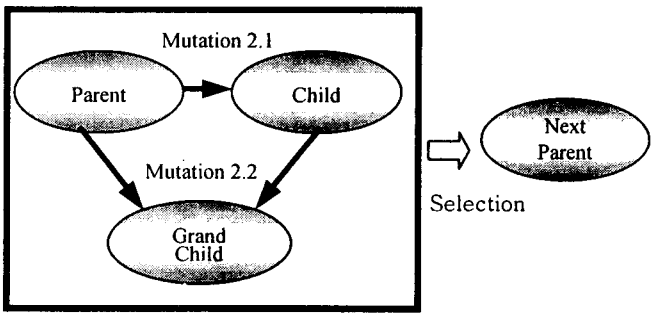

그림2. The proposed 3-generation mutation

모개체에서, 자손을 생성시키고, 다시 자손에서 손자를 생성시키는 직렬적인 방식이다. 그림2. 에서 두 가지 돌연변이 생성 방법은 부모개체에서 자손을 생성시킬 때와 자손에서 손자를 생성시킬 때이며, 두 가지 방법을 동시에 적용한다[8]. 자손을 생성시킬 때는 기존의 돌연변이 방법과 동일하나, 자손에서 손자를 생성시

Child : Mutation 2.1

$$x_j^i(k+1) = x_j^i[k] + \eta \cdot \nabla x_j^i(k) + \alpha \cdot sx_j^i(k)$$

$$\nabla x_j^i(k+1) = (x_j^{best}(k) - x_j^i(k)) \cdot |N(0,1)|$$

$$sx_j^i(k+1) = \eta \cdot acc^i(k) \cdot \nabla x_j^i(k+1) + \alpha \cdot sx_j^i(k)$$

$$acc^i(k+1) = \begin{cases} 1, & \text{if cost of } x(k+1) < \text{cost of } x(k) \\ 0, & \text{otherwise} \end{cases}$$

Grandchild : Mutation 2.2

if cost of $x(k+1)$ < cost of $x(k)$ then

$$x_j^i(k+2) = x_j^i[k+1] + \eta \cdot \nabla x_j^i(k+1) + \alpha \cdot sx_j^i(k+1)$$

$$\nabla x_j^i(k+2) = (x_j^{best}(k+1) - x_j^i(k+1)) \cdot |N(0,1)|$$

$$sx_j^i(k+2) = \eta \cdot acc^i(k+1) \cdot \nabla x_j^i(k+2) + \alpha \cdot sx_j^i(k+1)$$

else

$$x_j^i(k+2) = x_j^i[k] - \eta \cdot \nabla x_j^i(k) - \alpha \cdot sx_j^i(k)$$

$$\nabla x_j^i(k+2) = (x_j^{best}(k) - x_j^i(k+1)) \cdot |N(0,1)|$$

$$sx_j^i(k+2) = \eta \cdot acc^i(k) \cdot \nabla x_j^i(k+2) + \alpha \cdot sx_j^i(k)$$

$$acc^i(k+2) = \begin{cases} 2 & \text{if cost of } x(k+2) < \text{cost of } x(k) \\ 1 & \text{else if cost of } x(k+2) < \text{cost of } x(k+1) \\ 0 & \text{otherwise} \end{cases}$$

그림3. Detailed 3-generation mutation

킬 때는 자손의 정보뿐 아니라, 그 전 세대 즉 손자의 입장에서 본다면, 할아버지 세대의 정보를 이용할 수 있도록 하여야 한다. 기존의 Multiple Offspring방법이 하나의 부모 개체로부터 많은 수의 자손을 만들어 내는 방식인데 비하여[9], 부모가 하나의 자손을 생성시키고, 그리고 그 자손이 또 하나의 자손을 생성시키는 것으로 그 방법이 다르다고 할 수 있다. 여기에서 손자를 생성시킬 때는 기본적으로 부모의 정보를 이용하지만, 부모 세대와 조부모 세대의 적합도를 비교하여, 부모 세대를 그대로 물려 받을 것인지 아니면, 새롭게 생성 할 것인지를 결정하게 된다. 새롭게 생성할 때는 부모의 정보를 버리고, 다시 조부모 세대에서 탐색 영역을 반대 방향으로 찾게 된다. 해를 찾는 속도를 개선하기 위하여, 자손이 부모세대보다 적합도가 개선되면, 그 방향으로 '1'을 주고, 손자가 조부모 세대보다도, 개선된 경우는 그 방향으로 최적의 해가 존재할 가능성이 높으므로 '2'의 값을 부여하게 된다.

기타의 경우에 대해서는 개선이 되지 않은 경우이므로 '0'의 값으로 한다.

제안된 돌연변이에서는 기존의 알고리즘 중에서 상대적으로 우수한 특성을 보여주는 Neural Network의 일반적인 학습 방법인 Backpropagation 개념을 적용 하였다[10]. 학습률과 모멘텀의 개념을 그대로 사용하여 해를 찾는데 사용하고 있다. 여기에서 알고리즘의 성능이 학습률과 모멘텀 값에 의존하게 되므로[11], 학습률을 시간에 따른 변화 함수로 줌으로써, 학습률에 의한 영향을 줄이고, 탐색 영역을 시간에 따라 좀더 세부적으로 관찰 할 수 있게 하였다. 그림3에서 $x_j^i[k]$는 k번째 세대에서의 i번째 개체의 j번째 변수이다. η는 학습률 그리고 α는 모멘텀(momentum) 계수가 된다. 실제 문제에서 초기에는 넓은 영역을 탐색하다가 점점 작은 영역을 세분해서 탐색 해야 하므로 학습률을 어닐링(Annealing) 하여 사용하여 이러한 문제점에 적응하도록 하였고, $\nabla x_j^i(k)$는 개체의 변화율이다. 변화율은 매 세대마다 계산되는 오차이며, 다음세대에서 최적해에 가깝도록 하는 방향의 값으로 준다. $sx_j^i(k)$ 는 전 세대의 진화에 대한 모멘텀(Momentum)을 의미한다[10]. 부모개체로부터 자손을 생성 시키고, 그 자손으로부터, 손자를 생성시킬 때, 자손이 부모보다 열등한 결과를 낳게 되면, 손자는 자손이 아니라, 부모의 정보로부터, 자손과 반대 방향으로 해를 생성시킬 수 있도록 함으로써, 해의 방향에 따른 진화가 가능하다.

2.4 지역 토너먼트 선택 알고리즘

선택은 새로운 세대를 생산하지는 않지만, 모든 해중에서 최적의 해가 될만한 것은 선택하고, 그러지 않은 해는 버리는 판단의 역할을 한다. 따라서 아무리 좋은 해를 만들었다고 해도, 취사 선택을 잘못하면, 최적의 해를 찾지 못하거나, 또 많은 시간을 소요되게 된다.

본 논문에서는 토너먼트 선택 방법과 지역 선택 방법의 장점을 결합한 새로운 선택알고리즘을 제안하고자 한다.

대부분의 선택법에서는 집단 내의 모든 부모 개체들이 선택 급원(selection pool)이 되지만 지역선택에서는 이들은 지역이웃이라는 제약이 가해진 환경 안에 머물게 된다. 한 지역 내에 있는 개체들은 그 안에서만 경쟁하게 된다. 적합도의 순서에 따라 지역을 그룹화하고 각 그룹 안에서는 경쟁적으로 개체를 선택하는 토너먼트 선택방법을 채택하였다. 일반적인 토너먼트 선택 방법은 전체의 집단에서 임의로 추출된 인자와 경쟁을 통해서 살아남은 인자가 후손에 물려지지만, 때로는 지역적으로 높은 지대에 있는 인자들은 선택될 확률이 적어진다.

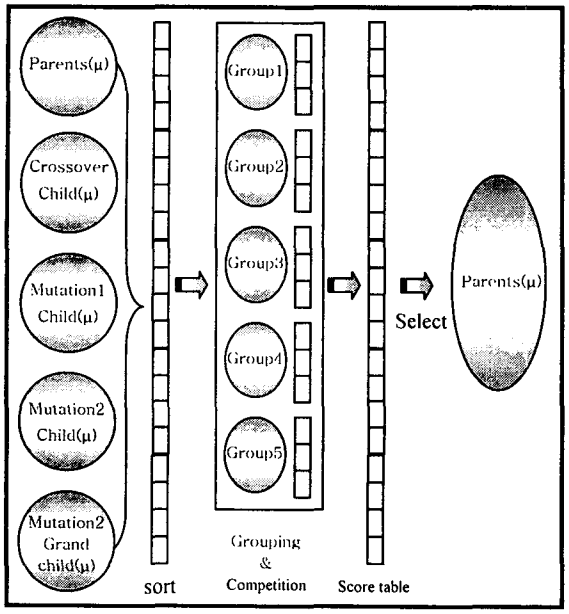

그림4. Proposed Selection Scheme

물론 선택강도(Selection Pressure)를 작게 하면, 확률적으로 열등인자도 선택될 확률이 높지만, 그러면 지역적으로 우수한 인자들이 선택되지 않을 확률이 높아진다[12] [13].

따라서 지역을 적합도 순으로 나누어 지역에서 먼저 토너먼트 경쟁방법을 통하여, 일차적인 경쟁을 하고, 이때 이긴 횟수를 가지고, 전체영역에서 이긴 횟수의 순위에 의거한 선택을 하게 되면, 매 지역에서 우수하게 결정된 인자는 반드시 선택이 될 수 밖에 없다. 즉 지역 우승자는 다음 세대에 선택이 되며, 나머지는 그 서열에 의거하여, 다음 세대로의 인자로 결정되게 된다. 이것은, 지역적으로 열등한 인자들도 진화에 참여할 수 있게 하는 것이므로 하게 되므로, 해의 다양성의 측면에 상당한 영향을 기여하게 된다.

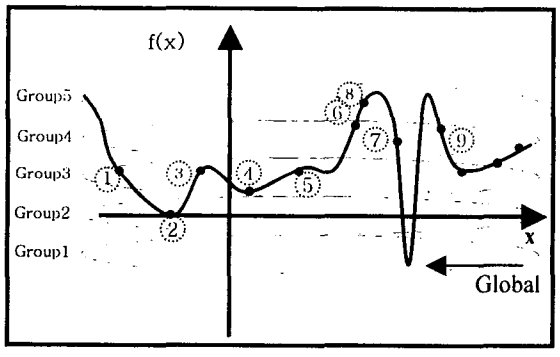

그림5. Scheme of the Grouping

위의 그림5에서 전체공간에서 선택을 한다면, 7번과 같은 해가 선택될 확률은 감소하게 된다. 따라서 해집합을 5개의 지역으로 분리하고 각 그룹에서 우선적인 경쟁을 하게 된다. 그것은 전체에서 토너먼트 선택을 할 경우에는 전체 중에서 우수한 인자가 선택될 확률이 높기 때문에 전체의 최적해로 가는 인자는 선택될 확률이 줄어들게 된다.

2.5 Simulation Results

제안된 알고리즘의 성능을 비교하기 위하여 Standard EP와 FEP를 Multiple Offspring으로 동일한 조건에서 10번의 반복 실험을 통해 평균 결과를 제시하였다. 성능 테스트를 위하여 6개의 Bench function을 사용하였고, 함수는 Sphere, Griewangk, Rosenbrock, Colville, Bohachevsky and Foxhole [14]을 통하여 제안한 알고리즘의 성능 결과를 제시하고자 한다.

Problem 1: Sphere function

$$f_1(x_1,x_2,x_3)=\sum w_i \times x_i^2$$

subject to: $-6.0 \le x_i \le 6.0$ w_i=[100 1 1], i = 1,2,3

Problem 2: Rosenbrock function

$$f_2(x_1,x_2)=100(x_1^2-x_2)^2+(1-x_2)^2$$

subject to: $-6.0 \le x_i \le 6.0$, i = 1,2

Problem 3. Coville function

$$f_3(x_1,x_2)=100(x_2-x_2)^2+(1-x_1)^2+90(x_4-x_3^2)^2+(1-x_3)^2+10.1((x_2-1)^2+(x_4-1)^2)+19.8(x_2-1)^2+(x_4-1)^2$$

subject to: $-10 \le x_i \le 10$, i = 1,2

Problem 4. Foxholes function

$$f_4(x_1,x_2)=\frac{1}{500+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}}-0.9980038$$

subject to: i = 1,2

a_{1j} = {-32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, 16,32}

a_{2j} = {-32, -32, -32, -32, -32, -16, -16, -16, -16, -16, 0, 0, 0, 0, 0, 16, 16, 16, 16,16, 32, 32, 32, 32, 32}

Problem 5. Bohachevsky function

$$f_5(x_1, x_2) = \frac{\left(x_1^2 + x_2^2\right)}{2} - \cos\left(20 \cdot \Pi x_1\right) \cdot \cos\left(20\Pi x_2\right) + 2$$

subject to: $-10 \le x_i \le 10$, $i = 10$

Problem 6. Griewangk function

$$f_6(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^4 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad n = 30$$

실험에 사용한 EP의 개체 수는 다음 표와 같다. 동일조건을 유지하기 위하여 자손개체수의 합이 같도록 하였다

Table. 1. Population Size

	SEP	FEP	BPEP	HEP		
Parents	50	50	50	50		
Children	200	200	200	Crossover	Children	50
				Mutation1	Children	50
				Mutation2	Children	50
					Grandchild	50

Standard EP나 FEP, BPEP에서는 자손을 생성시킴에 있어서 하나의 부모에서 네 개의 자손을 생성시키는 Multiple Offspring 방법을 사용하였으며, HEP는 자손과 손자의 개체 수를 합하면, 다른 EP의 자손의 개체 수와 동일하게 적용하였다. 또한 각 EP의 변수는 각각의 논문에 주어진 변수를 사용하였다 [9][10][12][15]. 위의 그림6-그림11까지 결과를 보면, 그림6-그림8까지 간단한 함수의 경우는 제안한 EP가 월등히 우수하게 해를 찾고 있음을 볼 수 있다. 또한, 그림9-그림11의 지역 해가 많은 문제의 경우에도 다른 EP보다 대체적으로 성능이 좋게 나오지만, 그림10을 보면, 초기 성능은 좋은데 시간이 흐를수록, FEP에 비하여 성능이 조금 저하되는 결과를 볼 수 있다. 하지만 다른 EP에 비해서는 상당히 좋은 결과를 나타낸다. 전체적으로 비교 했을 때 성능이 기존의 알고리즘보다, 우수하게 나타남을 확인할 수 있다.

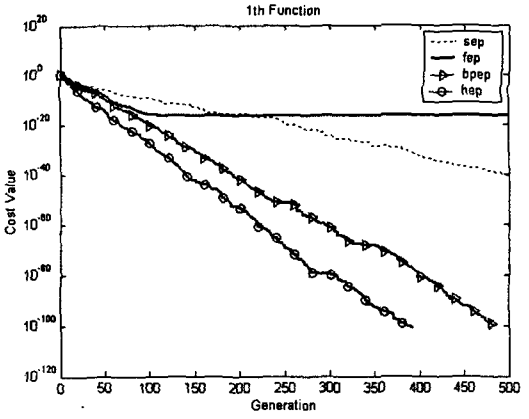

그림6. Comparison of evolution curves for Problem 1

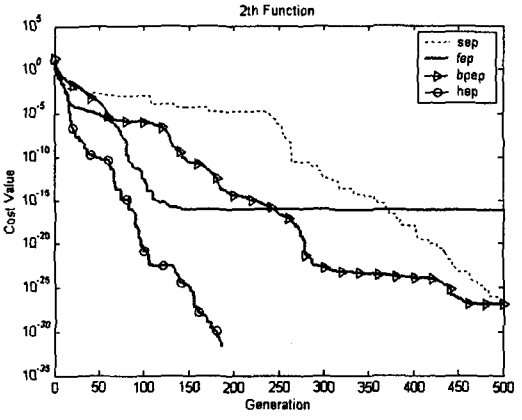

그림7. Comparison of evolution curves for Problem 2

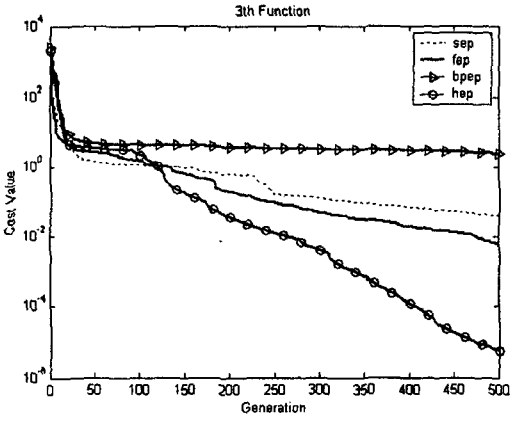

그림8. Comparison of evolution curves for Problem 3

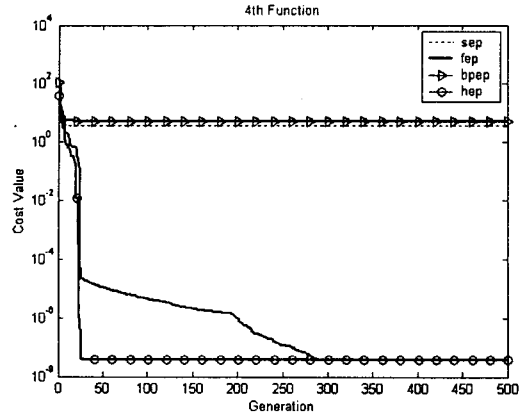

그림9. Comparison of evolution curves for Problem 4

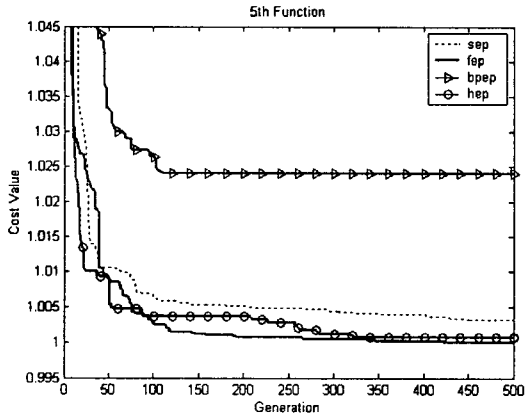

그림10. Comparison of evolution curves for Problem 5 (Bohachevsky function)

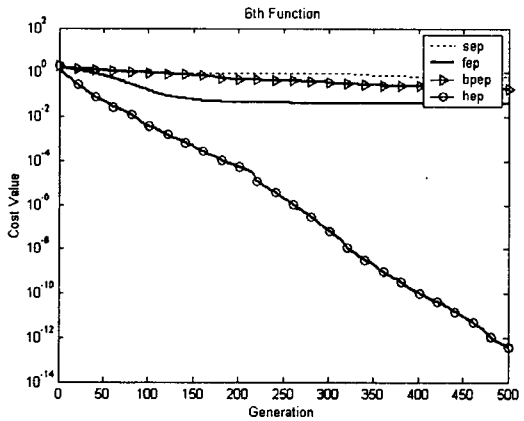

그림11. Comparison of evolution curves for Problem 6 (Griewangk function)

III. 결 론

본 논문에서는 지역탐색에 강한 특성을 가진 알고리즘과 전역 탐색에 강한 특성을 가진 알고리즘을 적절히 조합하여 최적 탐색을 이루는 하이브리드 알고리즘을 제안하였다. Neural Network의 Backpropagation Rule을 이용한 3세대 mutation 알고리즘이 지역 또는 전역 최적해를 찾기 위해 빠져 나와야 하는 오차 영역에서, GA의 crossover와 mutation의 두 연산자는 3세대 알고리즘이 항상 최적해에 수렴하는 지역 또는 전역 최소점 주위의 영역을 일컫는 'basins of attraction'을 찾는 데에 사용되었다. 또 지역 토너먼트 선택방법을 제안하여 개체군의 다양성(population diversity)과 선택 강도(selective pressure)사이의 균형을 이루어 최적 해에 빠르게 도달할 수 있게 하였다.

앞으로는 연산자들의 조합을 최적화하는 하이브리드 알고리즘을 만드는 연구와 하이브리드 알고리즘의 계산량과 수행시간을 줄이는 연구가 수행되어야 할 것이다.

IV. 참고문헌

[1] M. Mitchell, j. h. Holland and S. Forrest, "when will a Genetic Algorithm Outperform Hill Climbing?," Advances in Neural Information Processing Systems, J. D. Cown, G. Tesauro and J. Alspector(Eds), Morgan Kaufmann, San Mateo, CA, 1993

[2] L. B. Booker, "Improving Search in Genetic Algorithms," Genetic Algorithms and Simulated Annealing, L. Davis (Ed), Morgan Kaufmann Publishers, Los Altos, CA, pp.61-73, 1987

[3] D. M. Etter and M. M. Masukawa, "A Comparison Algorithms for Adaptive Estimation of the time Delay Between Sampled Signals," Proc. '81 IEEE Int. Conf. on Genetic Acoustics, Speech and Signal Processing, Vol. 3, pp.1253-1256, 1981

[4] T. Kido, H. Kitano, and M. Nakanishi, "A Hybrid Search for Genetic Algorithms: Combining Genetic Algorithms, Tabu Search, and Simulated Annealing," Proc. 5th Int. Conf. on Genetic Algorithms, S. Forrest(Ed.) pp. 641, 1993

[5] Z. Michalewicz, Genetic Algorithms + Data Structures= Evolution Programs, Springer-Verlag , Berlin Heidelberg, 1996

[6] C. Z. JaniKow and Z. Michalewicz, "An Experimental Comparison of Binary anf Floating Point Representations in Genetic

Algorithms," Proc. 4th Int. Conf. on Genetic Algorithms, R. Belew and L. B. Booker(Eds), Morgan Kaufmann Publishers, CA, 1991

[7] Hyeon-Kuk Jeong and Se-Young Oh, "Evolutionary Programming Integrating 3-Generation Based Mutation and Local Competition Based Selection," Submitted to IEEE Conference on Evolutionary Computation, 2002.

[8] Jinn-Moon Yang, "Integrating Adaptive Mutations and Family Competition with Differential Evolution for Flexible Ligand Docking", IEEE Transactions On Evolutionary Computation, pp. 473-480, 2001.

[9] Hyeon-Joong Cho, Se-Young Oh and Doo-Hyun Choi, "Fast Evolutionary Programming Through Search Momentum and Multiple Offspring Strategy", IEEE Transactions On Evolutionary Computation, pp. 805-809. 1998.

[10] Doo-Hyn Choi and Se-Young Oh, "A New Mutation Rule for Evolutionary Programming Motivated from Backpropagation Learning", IEEE Transactions On Evolutionary Computation, Vol.4, No.2, 2000

[11] Simon Haykin, "Neural Networks: A Comprehensive Foundation", MACMILLAN, 1999.

[12] Thomas Bäck, Ulrich Hammel, and Hans-Paul Schwefel, "Evolutionary Computation: Comments on the History and Current State," IEEE Trans. on Evolutionary Computation, vol. 1. no 1, pp. 3-17, 1997.

[13] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, "Handbook of Evolutionary Computation", Oxford University Press, 1997.

[14] Z.Michalewicz, Genetic Algorithms + Data Structures = Evolutionary Programs, 3rd ed. New York:Springer-Verlag, 1996

[15] Jong-Hwan Kim, Hong-Kook Chae, Jeong-Yul Jeon, and Seon-Woo Lee, "Identification and Control of Systems with Friction Using Accelerated Evolutionary Programming", IEEE International Conference on Intelligent Control and Instrumentation, pp.188-191, 1995.