여러 부집단을 이용한 새로운 진화 프로그래밍 기법

임종화, 황찬식, 한대현, 최두현
경북대학교 전자전기공학부
대구광역시 북구 산격동 1370 번지

A New Evolutionary Programming Technique Using Multiple Populations

Jong-Hwa Lim, Chan-Sik Hwang, Dae-Hyun Han, and Doo-Hyun Choi
School of Electronic and Electrical Engineering
Kyungpook National University Taegu 702-701, Korea
dhc@ee.kyungpook.ac.kr

Abstract: A new evolutionary programming technique using multiple subpopulations with completely different evolution mechanisms is proposed to solve the optimization problems. Three subpopulations, each has different evolution characteristics and uses different EP algorithms such as SAEP, AEP, and FEP, are cooperating with synergy effect in which it increases the possibility to quickly find the global optimum of continuous optimization problems. Subpopulations evolve in different manner and the interaction among these leads to global minimum quickly.

I. 서론

진화 연산은 계획(planning), 설계(design), 모사(simulation and identification), 제어(control), 분류(classification) 등의 다양한 최적화 분야에 강건한 성능을 발휘하는 알고리듬의 개발을 위해 꾸준히 계속되어 왔다[1]. 진화 알고리듬들은 전통적인 방법에 비해 비선형적이고 미분이 불가능하거나 힘들며, 많은 극값을 가지는 문제에 대해서도 적합도 함수만 정의할 수 있다면 모든 최적화 문제에 효과적으로 적용될 수 있다. 그러나 진화 알고리듬은 조기 수렴 등의 문제를 가지고 있다. 이 때문에 여러 알고리듬을 적절히 결합함으로써 상승효과(synergism)를 기대할 수 있는 하이브리드(hybrid) 방법은 탐색 알고리듬(search algorithm) 뿐만 아니라 다른 분야에서도 문제 해결의 좋은 방법이 된다. 본 논문에서는 서로 다른 진화 형태를 가지는 알고리듬의 하이브리드를 위해 병렬 유전 알고리듬(Parallel Genetic Algorithm; PGA)의 부집단(subpopulation) 개념을 도입한 부집단을 이용한 새로운 하이브이드 방법을 제안한다. 각 부집단들은 각기 다른 조건하에서 독립적으로 진화를 수행하고 일정 시간(isolation time) 후에는 개체 이주(migration)를 통해서 부집단 간에 정보를 교환한다. 본 논문에서는 자기 적응 진화 프로그래밍(ASEP), Accelerated 진화 프로그래밍(AEP), Fast 진화 프로그래밍(FEP)을 부집단으로 하는 알고리듬으로 잘 알려져 있는 표준 테스트 함수(benchmark test function)에 대해서 제안된 알고리듬의 우수성을 검증한다.

II. 부집단을 이용한 진화 프로그래밍 기법

본 논문에서는 서로 다른 진화 연산자를 가지는 세부집단을 이용한 다 집단 진화 프로그래밍(Multiple Population Evolutionary Programming; MPEP) 기법을 제안한다. MPEP 에 사용된 부집단은 다음과 같다.

1. Self-Adaptive Evolutionary Programming (SAEP)
2. Accelerated Evolutionary Programming(AEP)
3. Fast Evolutionary Programming(FEP)

SAEP 에서 개체 $a = (x, \sigma)$는 식 (1)에 따라 돌연 변이한다[2].

$$\begin{aligned} x_i' &= x_i + N(0,\sigma) \\ \sigma_i' &= \sigma_i \cdot \exp(\tau \cdot N(0,1) + \tau' \cdot N_i(0,1)) \end{aligned} \tag{1}$$

식 (1)에서 $\tau \propto (\sqrt{2\sqrt{n}})^{-1}$, $\tau' \propto (\sqrt{2n})^{-1}$ 라 정해지나 최적화 목적 함수의 특성에 따라 다른 값을 가질 수도 있다.

다양성을 감소시키지 않고 수렴 속도를 향상시키기

위해 개체에 방향(direction) 연산자와 나이(age) 변수를 사용한 AEP 알고리듬이 제안되었다[3]. AEP 에서 개체는 식 (2.2)와 같이 표현된다.

$$x^i = \left[x_1^i, \cdots, x_n^i, dir(x_1^i), \cdots, dir(x_1^i), age^i\right]^T \quad (2)$$

AEP 에서의 돌연 변이는 SAEP 와는 다른 방법으로 진행된다. SAEP 가 모든 변수에 대해 동일한 방법으로 돌연 변이하는 반면, AEP 는 개체의 나이에 따라 돌연 변이 방법을 달리한다. AEP 의 방향 연산자와 나이 변수의 변화는 그림 (1)에 따라 수행되고, 돌연 변이는 그림 (2)의 규칙에 따라 수행된다.

If $f(x^i) < f(x^i[k-1])$,

then: $dir(x_j^i[k]) = \mathrm{sgn}(x_j^i[k] - x_j^i[k-1])$;

$age^i[k] = 1$;

else: $age^i[k] = age^i[k-1] + 1$;

$\forall i \in \{1,2,\cdots,N_p\}, \forall j \in \{1,2,\cdots,n\}$

그림 1. AEP 의 방향 연산자와 나이 변수의 변화.
Fig 1. Change of directions and ages for the AEP.

If $age^i[k] = 1$

$$\text{then:} \begin{cases} \sigma^i = \beta_1 \cdot f(x^i[k]), \\ x_j^i[k] = x_j^i[k-1] \\ \quad + dir(x_j^i[k-1]) \cdot \left|N(0,\sigma^i)\right|; \end{cases} \quad (3)$$

$$\text{else:} \begin{cases} \sigma^i = \beta_2 \cdot f(x^i[k]) \cdot age^i, \\ x_j^i[k] = x_j^i[k-1] + N(0,\sigma^i); \end{cases} \quad (4)$$

$\forall i \in \{1,2,\cdots,N_p\}$, $\forall j \in \{1,2,\cdots,n\}$

where β_i, i=1,2.: positive constants

그림 2. AEP 의 돌연 변이 규칙
Fig 2. Mutation rule for the AEP

FEP 도 AEP 와 마찬가지로 개체의 나이와 방향 연산자를 사용한다[4]. 그러나 AEP 의 방향 연산자가 -1 혹은 1 의 값만 가지는데 반해, FEP 는 방향 연산자의 값으로 식 (7)과 같이 실수값을 가진다. FEP 의 개체 표현은 식 (2)와 동일하다. 그리고 돌연 변이 방법은 적합도를 스케일링(scaling)해서 표준 편차를 구한다는 것을 제외하고는 동일하다. FEP 의 돌연 변이 방법은 그림 (3)과 같다.

본 논문에서는 각기 다른 특성을 가지는 세 알고리듬을 결합하여 모든 문제에 대해서 우수한 성능을 나타내는 알고리듬을 제안한다. 제안된 알고리듬은 각 부집단이 독립적으로 진화를 수행 후 개체 이주(migration)를 통해서 상호간의 정보를 교환한다. 각 부집단에서 최고인 두 개체들로 최고 집합(best pool)을 형성 하고, 최고 집합에서 최고(best-of-the-best)인 두 개체와 모든 부집단에서 적합도가 가장 낮은 두 개체를 교체함으로써 개체 이주가 이루어진다. 이때 개체의

If $age^i = 1$,

$$\text{then:} \begin{cases} \sigma^i = \beta_1 \cdot scale\left(f(x^i)\right), \\ x_j^{i+N_p} = x_j^i + dir(x_j^i) \cdot N(0,\sigma^i); \end{cases} \quad (5)$$

$$\text{else:} \begin{cases} \sigma^i = \beta_2 \cdot scale(f(x^i)) \cdot age^i \\ x_j^{i+N_p} = x_j^i + N(0,\sigma^i); \end{cases} \quad (6)$$

$$dir\left(x_j^{i+N_p}\right) = \frac{\left(x_j^{i+N_p} - x_j^i\right)}{\sqrt{\sum_j \left(x_j^{i+N_p} - x_j^i\right)^2}}; \quad (7)$$

$age^{i+N_p} = 1$; $age^i = age^i + 1$;

$\forall i \in \{1,2,\cdots,N_p\}$, $\forall j \in \{1,2,\cdots,n\}$

where β_i, i=1, 2.: positive constants

그림 3. FEP 의 돌연 변이 규칙.
Fig 3. Mutation rule for the FEP.

나이를 1 로 하여 이주시키면 같은 알고리듬에 대해서 동일한 개체가 다른 방법으로 진화가 이루어지기 때문에 각 부집단에서 작은 개체수로 인한 발생할 수도 있는 다양성의 감소를 보완할 수 있다. 제안된 알고리듬의 수행 과정은 그림 (4)와 같다.

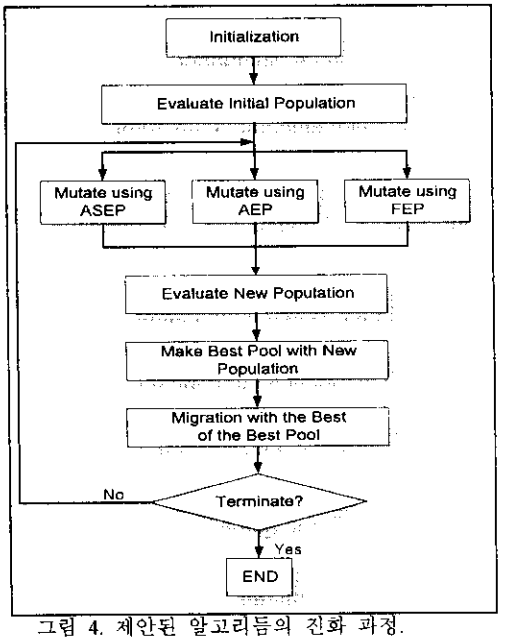

그림 4. 제안된 알고리듬의 진화 과정.
Fig 4. Evolution procedures of the proposed technique.

제안된 알고리듬에 사용된 부집단의 특성은 표 1 에 요약하였다.

표 1. 부집단의 특성 비교

	선택 방법	적합도 스케일링	방향 연산자 $dir(x_i')$
SAEP	$(\mu+\mu)$	-	×
AEP	$(1+1)$	×	$dir(x_i') \in \{-1,1\}$
FEP	$(\mu+\mu)$	O	$dir(x_i') \in R$

III. 실험 결과 및 고찰

제안된 알고리듬의 성능을 확인하기 위해 잘 알려져 있는 표준 테스트 함수에 적용해 보았다[5].

Problem 1. De Jong Function 1 (Sphere)

$$f(x_1,x_2,x_3)=\sum_{i=1}^{3}x_i^2,\ -6.0\le x_i\le 6.0$$

Problem 2. De Jong Function 2 (Rosenbrock)

$$f(x_1,x_2)=100\left(x_1^2-x_2\right)^2+\left(1-x_1\right)^2,\ -6.0\le x_i\le 6.0$$

Problem 3. De Jong Function 4

$$f(x_1,\cdots,x_{30})=\sum_{i=1}^{30}x_i^4+N(0,1),\ -1.28\le x_i\le 1.28$$

Problem 4. De Jong Function 5 (Foxholes)

$$f(x_1,x_2)=\frac{1}{1/500+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}}-0.998,$$

$$-65.536\le x_i\le 65.536$$

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

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

Problem 5. Colville Function

$$f(x_1,\cdots,x_4)=100(x_2-x_1^2)^2+(1-x_1)^2+90(x_4-x_3^2)^2$$
$$+(1-x_3)^2+10.1\left((x_2-1)^2+(x_4-1)^2\right)+19.8(x_2-1)(x_4-1)$$

$$-10\le x_i\le 10$$

Problem 6. Griewangk Function

$$f(x_1,\cdots,x_{10})=\sum_{i=1}^{n}\left(x_i^4/4000\right)-\prod_{i=1}^{n}\cos(x_i/\sqrt{i})+1$$

$$-600\le x_i\le 600$$

랜덤한 초기치의 효과를 무시하고 알고리듬의 성능을 확인하기 위해 50 번의 반복 수행 후 평균값을 비교하였다. 모든 테스트 함수에 대해서 가능하면 같은 조건으로 실험을 행하였다. 다음에 실험 조건을 나열하였다.

1) 개체수는 45 개로 모든 알고리듬에 동일. 단, MPEP 에서는 각 부집단을 15 로 하였다.
2) 초기 표준 편차는 0 ~ 1 사이의 값을 선택.
3) 개체 초기 나이는 1.
4) AEP: β_1 =5.0, β_2 =0.005, FEP: β_1 =0.5, β_2 =0.005
5) AEP: 초기 방향 연산자: -1 혹은 1.
   FEP : 초기 방향 연산자: -1 ~ 1 사이의 실수값.

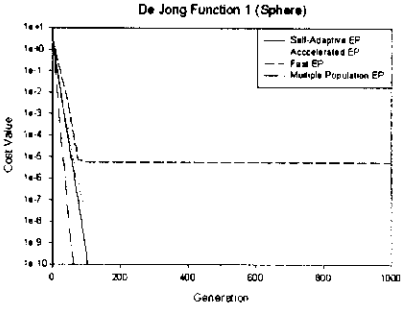

그림 5. Problem 1 에 대한 평균 최소 적합도의 진화.
Fig 5. Evolution of the averaged minimum cost for Problem 1.

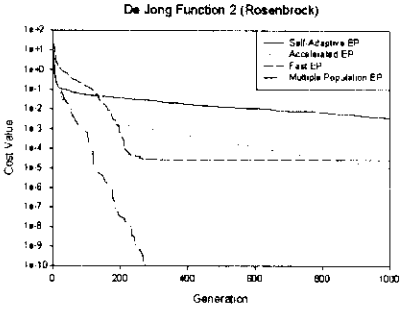

그림 6. Problem 2 에 대한 평균 최소 적합도의 진화.
Fig. 6. Evolution of the averaged minimum cost for Problem 2.

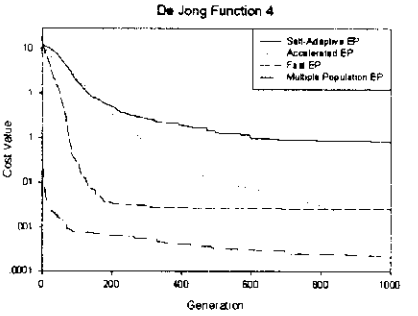

그림 7. Problem 3 에 대한 평균 최소 적합도의 진화.
Fig. 7. Evolution of the averaged minimum cost for Problem 3.

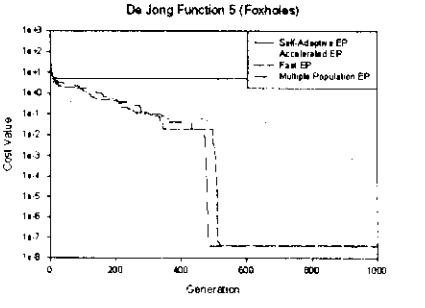

그림 8. Problem 4 에 대한 평균 최소 적합도의 진화.
Fig. 8. Evolution of the averaged minimum cost for Problem 4.

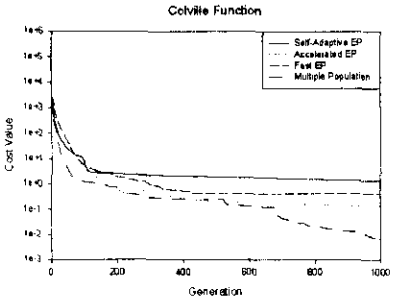

그림 9. Problem 5 에 대한 평균 최소 적합도의 진화.
Fig. 9. Evolution of the averaged minimum cost for Problem 5.

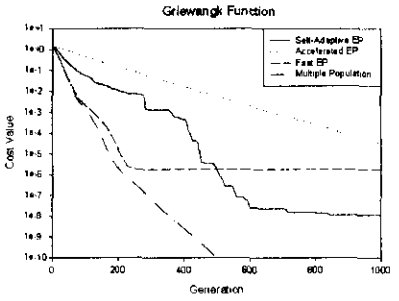

그림 10. Problem 6 에 대한 평균 최소 적합도의 진화.
Fig 10. Evolution of the averaged minimum cost for Problem 6.

각 부집단들이 특정 함수에 대해 우수한 성능을 보여줌을 그림 5 에서 그림 10 의 결과 그래프에서 쉽게 확인할 수 있다. 즉, SAEP 는 적합도 함수의 모양이 급격한 기울기를 이루고 있으면서 하나의 전역해를 가지는 함수나 전역 최적해의 깊이에 비해 상대적으로 낮은 지역 최적해를 가지는 함수에서 우수한 성능을 보인다. AEP 는 적합도 함수의 차수가 낮고, 차원이 낮은 함수에서 우수한 성능을 보이고, FEP 는 적합도 함수의 차수가 높고 차원이 높은 함수에서 우수한 성능을 보였다.

MPEP 는 실험에 사용한 모든 함수에 대해서 다른 알고리듬보다 우수한 성능을 보였다. 이는 MPEP 가 각 부집단의 장점을 유지하면서 각 부집단으로 이주한 우수한 개체들이 각기 다른 알고리듬으로 진화함으로 인해 다양성의 향상과 빠른 수렴 속도를 얻을 수 있기 때문이다.

IV. 결론

모든 최적화 문제에 적합한 알고리듬을 개발하려는 많은 연구가 있었지만 어떤 알고리듬도 모든 문제를 해결하지는 못했다(No Free Lunch(NFL) theorem)[1]. 그래서 본 논문에서는 특정 함수에 대해 우수한 성능을 보이는 알고리듬들을 부집단으로 사용한 새로운 알고리듬을 개발함으로써 진화 연산의 성능을 개선하려고 시도했다. 즉, 모든 문제에 잘 적응하는 알고리듬은 없지만, 제안된 MPEP 는 본 논문에서 사용한 다른 알고리듬보다는 훨씬 우수한 성능을 보였다. 이는 각 부집단에서 가장 뛰어난 개체들에 의해서 이주가 일어나게 되므로, 동일한 개체가 각기 다른 특징을 가지는 알고리듬에 의해 진화하기 때문에 다양성을 향상시킬 뿐만 아니라 가장 뛰어난 개체 주위를 자주 탐색하게 되므로 빠른 수렴 속도를 얻을 수 있다.

앞으로 제안된 MPEP 의 성능을 향상시키기 위한 최선의 개체 이주 방법을 찾고, 최상의 부집단을 구현하기 위한 연구가 행해져야 할 것이다. 아울러 더 많은 시험 함수에 대해서도 실험이 이루어져야 할 것이다. 그리고 제안된 알고리듬이 실제 문제에 대해서도 표준 테스트 함수에서 보인 성능을 나타낼 수 있는지도 확인해야 할 것이다.

참고문헌

[1] T. Back, D. B. Fogel, and Z. Michalewicz, "Handbook of Evolutionary Computation", Oxford University Press, 1997.

[2] T. Back and H. P. Schewefel, "Evolutionary Computation: An Overview", IEEE Int. Conf. on Evolutionary Computation, pp. 20-29, 1996.

[3] Jong-Hwan Kim, Hong-Kook Chae, Jeong-Yul Jeon, and Seon-Won Lee, "Identification and Control of Systems with Friction Using Accelerated Evolutionary Programming", IEEE Control Systems, pp. 38-47, 1997.

[4] Hyeon-Joong Cho, Se-Young Oh, and Doo-Hyun Choi, "Fast Evolutionary Programming Through Search Momentum and Multiple Offspring Strategy", ICEC, 1998.

[5] Z. Michalewicz, "*Genetic Algorithms + Data Structures = Evolutionary Programmings*", 3rd rev. and extended ed., Springer-Verlag, USA, pp. 349-352, 1996.