다양성을 유지하는 새로운 진화 프로그래밍 기법

신 정 환(申 正 煥)†, 진 성 일(秦 成 一)†, 최 두 현(崔 斗 鉉)*
†경북대학교 전자전기공학부, *경북대학교 차세대 정보통신연구소/정보통신학과
전화 : (053) 940-8570 / 팩스 : (053) 950-5505

A New Diversity Preserving Evolutionary Programming Technique

Jung-Hwan Shin†, Sung-Il Chien†, and Doo-Hyun Choi*
†School of Electronics and Electrical Engineering, Kyungpook National University,
*Center for Next Generation Information Technology/Dept. of Information & Communication
E-mail : tobewidu@palgong.knu.ac.kr

Abstract

In this paper, a new algorithm has been presented that helps to preserve diversity as well as to enhance the convergence speed of the evolutionary programming. This algorithm is based on the cell partitioning of search region for preserving the diversity. Until now, the greater part of researches is not concerned about preserving the diversity of individuals in a population but improving convergence speed. Although these evolutions are started from multi-point search at the early phase, but at the end those search points are swarming about a one-point, the strong candidate. These evolutions vary from the original idea in some points such as multi-point search. In most case we want to find the only one point of the best solution not several points in the vicinity of that. That is why the cell partitioning of search region has been used. By restricting the search area of each individual, the diversity of individual in solution space is preserved and the convergence speed is enhanced. The efficiency of the proposed algorithm has been verified through benchmark test functions.

Ⅰ. 서론

지난 30 여 년 동안 적응과 최적화 문제를 풀기 위해서 생물학적인 진화 체계를 모방하려는 많은 시도들이 있었다. 이런 시도들을 일컬어 진화 연산(evolutionary computation: EC)이라 한다. 진화연산은, 주어진 문제의 해를 많은 수의 개체 (individual)라 생각하고, 이 개체로 이루어진 집단을 진화시키면서 더 나은 개체를 찾기 위해서 확률 연산자를 사용한다. 이 진화연산은 개체 표현 방법과 진화 연산자의 종류, 선택 방법, 그리고 재생산 방법에 따라서 유전 알고리즘(genetic algorithm: GA)과 진화 전략(evolution strategy: ES), 그리고 진화 프로그래밍(evolutionary programming: EP) 등으로 구분된다[1,2]. 유전 알고리즘은 1960년대에 미국의 Holland[3]가 처음 제안하였고, De Jong이 이를 변수 최적화 문제에 적용할 수 있음을 주장하였다[4]. 진화 전략은 1960년대 독일의 Rechenberg[5]가 처음 제안했고, 그 후 Schwefel[6]에 의해 개선 및 발전되었으며 실수 함수의 최적화 문제에 적용되었다. 진화 프로그래밍은 Fogel[7]이 제안하였고, 진화 전략과 마찬가지로 실수 함수 최적화에 적용되었다. 이런 진화 연산은 계획, 설계, 모사, 제어, 분류 등의 다양한 최적화 분야에서 원하는 목적을 달성할 수 있는 강건한 알고리즘을 개발하기 위해서 계속 연구되고 있다[8].

Ⅱ. 기존의 탐색 알고리즘

최적화 문제를 해결하기 위한 방법으로는 수학적 해석에 의존하는 전통적인 방법과 자연의 생물학적 진화를 모방하는 진화 연산으로 크게 나눌 수 있다.

진화 연산은 자연의 진화 과정, 즉 복잡하고 잘 적응된 생명 유기체의 출현 과정을 모방하고 있다[1]. 이런 알고리즘들은 복잡한 여러 문제에 대해서 뛰어난 강인성을 발휘하고 있다. 그리고 이런 진화 연산은 개체를 표현하는 방법, 변화 연산자의 종류, 그리고 선택 및 재생산 방법에 따라 구분할 수 있다[2].

이 장에서는 그 중 진화 프로그래밍과 그 기법들에 대해 알아본다.

1. 진화 프로그래밍

1960년에 Fogel에 의해서 고안된 진화 프로그래밍은 유전 알고리즘과 유사한 확률적인 최적화 전략(stochastic optimization strategy)이다[11]. 그러나, 유전 알고리즘이 자연에서 관찰되는 특정 진화 연산자에 초점이 맞추어진 반면, 진화 프로그래밍은 부모와 자손 사이의 행위적인 관계에 초점을 두고 있다. 진화 프로그래밍은 수학적인 해석에 의존하는 전통적인 방법이 잘 풀지 못하는 비선형적이고 미분이 불가능하거나 힘들며, 많은 지역 최적해를 가지는 문제에 좋은 성능을 보이는 것으로 알려져 있다[2]. 특히 주어진 최적화 문제에 대한 사전 지식이 없는 경우에도 적합도 함수만 정의된다면 모든 최적화 문제에 효과적으로 적용할 수 있는 장점이 있다. 이러한 장점에도 불구하고 진화 프로그래밍은 조기 수렴 현상 등의 여러 가지 문제를 가지고 있다. 이러한 문제를 해결하기 위해 새로운 돌연변이 연산자를 이용하는 방법과 여러 알고리즘을 하이브리드(hybrid)하여 각 알고리즘의 장점을 취하고 단점을 보완하도록 하는 방법들이 제안되어 왔다[9, 10].

2. 자기 적응 진화 프로그래밍

진화 프로그래밍의 단점을 보완하기 위해 제안된 대표적인 프로그래밍 알고리즘이 자기 적응 진화 프로그래밍(self-adaptive evolutionary programming: SAEP)이다[8]. 이 자기 적응 진화 프로그래밍은 Schwefel에 의해 제안된 것인데, 이는 개체 공간 뿐만 아니라 파라미터 공간도 같이 탐색해서 진화 프로그래밍의 성능을 향상시킨다. 이 알고리즘에서의 개체는 식 (1)과 같이 표현된다.

$$a=[x_1,\cdots,x_n,\sigma_1,\cdots,\sigma_n]^T \tag{1}$$

자기 적응 진화 프로그래밍의 돌연변이 연산은 이전 스텝 사이즈에 로그-노말(log-normal) 형태의 확률 분포를 가지는 난수를 곱해서 스텝 사이즈를 먼저 돌연변이 시킨 후, 평균이 0이고 변환된 스텝 사이즈를 표준 편차로 하는 정규 분포를 따르는 난수를 개체에 더해서 이루어진다. 돌연변이 방법은 식 (2)와 같다.

$$\begin{aligned}\sigma_i' &= \sigma_i \cdot \exp(\tau \cdot N(0,1)+\tau' \cdot N_i(0,1))\\ x_i &= x_i+\sigma_i' \cdot N_i(0,1)\end{aligned} \tag{2}$$

여기에서 학습률 $\tau' \propto (\sqrt{2n})^{-1}$과 $\tau \propto (\sqrt{2\sqrt{n}})^{-1}$는 Schwefel에 의해 경험적으로 정해졌지만, 이 값들은 최적화 목적 함수의 특성에 따라서 다른 값을 가질 수도 있다. 그리고 스텝 사이즈에는 지수형의 값이 곱해지므로 항상 양의 값을 유지할 수 있다.

Fogel은 위의 방법과는 다른 진화 프로그래밍의 자기 적응 알고리즘을 제안했다[8]. Fogel의 알고리즘에서는 스텝 사이즈의 돌연변이에 로그-노말 형태의 난수 대신 정규 분포를 따르는 난수를 더해 주는 돌연변이 방법을 사용한다. 돌연변이는 식 (3)과 같다.

$$\begin{aligned}\sigma_i' &= \sigma_i+\alpha \cdot \sigma_i \cdot N_i(0,1)\\ x_i' &= x_i+\sigma_i' \cdot N_i(0,1)\end{aligned} \tag{3}$$

여기서 α는 스케일링 계수로 양의 상수이다. 만약 스텝사이즈 σ_i가 음수가 되는 경우에는 음수인 스텝 사이즈 대신에 작은 임의의 값 ε으로 대신한다.

Ⅲ. 제안된 알고리즘

지금까지 진행되어온 연구는, 다양성과 수렴성의 측면에서 다양성보다는 수렴성에 초점을 맞추어 진행되어왔다고 해도 과언이 아니다. 물론 초기 집단군에서는 개체가 탐색 영역 전체에 골고루 퍼져 있으나, 진화가 진행됨에 따라 이들 개체는 그 집단 내에서 가장 유력한 개체의 주변으로 몰리게 된다. 이것은 전체 영역 중 유력한 개체가 존재하는 근처 영역을 집중적으로 찾음으로써 해의 수렴 속도를 높일 수는 있으나, 자칫 지역 최적해에 빠질 위험을 가지고 있다.

그래서 이 논문에서는 이런 위험을 피하면서 동시에 해의 수렴 속도도 보장할 수 있는 알고리즘(Diversity Preserving EP: DPEP)을 제안하고자 한다.

우리는, 여러 사람으로 이루어진 한 모임이 어려운 문제 또는 복잡한 문제에 접했을 때, 그 문제를 풀기 위해 다룰 수 있는 작은 문제로 나누어 좀 더 문제를 쉽게 해결하고자 한다. 이것은, 각 구성원이 서로 맡은 일에 방해를 받지 않고 자신이 맡은 일을 함으로써 문제를 해결하려는 의도와 어려운 일을 보다 쉬운 작은 일로 나누어 풀려는 의도(Divide and Conquer)를 담고 있다. 이 논문에서 우리가 제안하고자 하는 알고리즘도 이런 생각을 포함하고 있다.

이 알고리즘은, 먼저 각 개체가 진화해 나갈 영역 즉, 탐색 영역을 분할하여 각 개체가 탐색할 영역을 정해주고, 그 제한된 영역을 탐색함으로써 그 집단의 다양성을 유지하고자 한다.

이 논문에서 다양성이라는 것은, 한 세대에서 가장 높은 적합도를 갖는 개체와 그 밖의 개체들 사이의 유클리드 거리(Euclidean distance)를 구하고, 초기 개체군에서의 유클리드 거리와 각 세대에서의 유클리드 거리의 비로 정의하고 있다. 다양성 척도(Diversity measure)는 식 (4)와 같다.

$$\text{Diversity} = \frac{D_i{}^j(gen)}{D_i{}^j(init)} \tag{4}$$

$$D_i{}^j(gen) = \sqrt{\sum_j (x_i{}^j - x_{best}{}^j)^2} \tag{5}$$

식 (5)는 개체 사이의 유클리드 거리이고, x_{best}는 그 세대에서 가장 높은 적합도를 갖는 개체이다.

Ⅳ. 실험 결과 및 고찰

이 논문에서 제안한 다양성을 유지하는 진화 프로그래밍의 성능 평가를 위해 표준 테스트 함수에 대해서 실험을 수행하여 그 결과를 고찰해 본다.

먼저 본 논문에서 사용된 표준 테스트 함수와 그 특징을 살펴보자[12].

함수1: Foxholes function(De Jong F5)

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

where $-65 \le x_i \le 65$

이 함수는 $(x_1, x_2) = (-32, -32)$에서 전역 최소값 0.9980038을 가진다.

함수2: Rosenbrock function(De Jong F2)

$f(x_1, x_2) = 100 \cdot (x_1^2 - x_2)^2 + (1 - x_1)^2$,
where $-2.048 \le x_i \le 2.048$

이 함수는 $(x_1, x_2) = (0, 0)$에서 전역 최소값 0을 가진다.

위에 열거한 함수 중에서 Rosenbrock은 오직 하나의 전역해를 가지지만 완만한 기울기를 가지는 모양으로 조기 수렴 현상 여부를 시험할 수 있다. 그리고 Foxholes function은 많은 지역해를 가지는 함수로, 본 알고리즘의 많은 지역해에 어떤 성능을 발휘하는 지를 가늠할 수 있다.

본 논문에서 비교하고 있는 두 알고리즘의 개체의 수는 36개로 하여 실험을 수행하였으며, 임의적인 초기치 효과를 무시하며 성능을 평가하기 위해 10번의 반복 수행을 통한 평균값을 비교하였다. 그리고 제안한 알고리즘은, 전체 영역을 작은 영역으로 나누고(cell partitioning), 그 작은 영역(cell)에 하나의 개체를 위치시켜서 탐색을 하는 것이므로, 기존의 자기 적응 진화 프로그래밍에서 이루어진 $(\mu + \mu)$ 전략을 사용하지 않고, 각 작은 영역에서 부모 세대보다 나은 자손이 나올 때까지 돌연변이 연산을 수행하여 보다 나은 개체를 다음 세대의 부모로 정하는 선택 전략인 $(1, 1)$ 전략을 사용하였다.

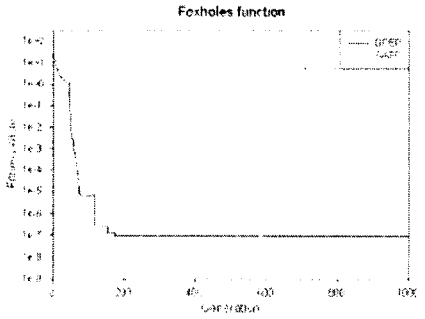

그림 1. 함수 1에 대한 평균 최소 적합도의 진화

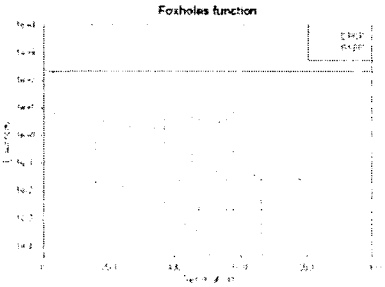

그림 2. 함수 1에 대한 각 세대의 다양성의 진화

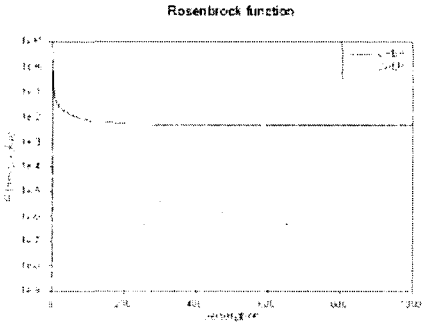

그림 3. 함수 2에 대한 평균 최소 적합도의 진화

여기서는 자기 적응 진화 프로그래밍과의 성능 비교를 위해 식 (3)에서와 같이 기존의 자기 적응 진화 프로그래밍에서 사용한 돌연변이 방법을 사용하였다.

그림 1에서 보면, 제안한 알고리즘이 기존의 자기 적응 진화 프로그래밍보다 빠르게 수렴하는 것을 보이고 있다. 그리고 그림 2에서는 그 결과에 대한 다양성의 변화를 보이고 있는데, 자기 적응 진화 프로그래밍은 지역해에 빠졌을 때 스텝 사이즈를 변화시켜 다양성을 유지하려 하지만 결국에는 다양성이 감소하는 결과를 보이고 있다. 이로 인해서 지역해에 빠지는 결과가 발생하였다. 그러나 제안한 알고리즘은 다양성을 유지함으로써 보다 나은 결과를 얻고 있다.

그러나 그림 3에 보면, 제안한 알고리즘이 자기 적응 진화 프로그래밍보다 오히려 못한 결과를 보이고 있다. 이는 자기 적응 진화 프로그래밍은 우등한 개체 근처로 모여서 해를 찾아감으로써 보다 나은 해를 찾아갈 확률이 그만큼 큰 반면, 제안한 알고리즘은 지역 탐색 영역(local area)에서 지역 최적해(local optimum)을 찾는 메카니즘이므로, 함수 2와 같이 완만한 기울기를 갖는 경우에는 한 세대의 개체 수에 반비례하는 확률을 가지고 탐색을 하기 때문에 보다 좋지 못한 결과를 낳고 있다.

V. 결론 및 추후 연구

지금까지 이루어진 많은 연구는, 진화가 진행됨에 따라 개체들이 우등한 개체 주변으로 몰려 함께 탐색하는 방식으로 이루어졌다. 이런 연구는 다점 탐색법(multi-point search mechanism)이라기 보다는 단점 탐색법(one-point search mechanism)이라 할 수 있다. 그런 결과, 해의 다양성 측면보다는 수렴성에 보다 많은 관심을 가지며 진행되어 왔다. 이런 연구는 해의 다양성이라는 점을 간과한 나머지, 자칫 지역 최적해에 빠져버리는 결과를 가지는 경우가 많았다. 그래서 본 논문에서는 해의 다양성을 유지하여 지역 최적해에 빠질 확률을 줄임으로써 보다 빠른 수렴성을 가지는 알고리즘을 제시하였다. 테스트 함수에 의한 실험 결과, 지역 최적해를 많이 가지고 있는 함수에 대해서 다양성을 유지하며 기존 알고리즘보다 빠르게 해에 수렴하는 결과를 얻었다.

그러나 지역 최적해가 없이 완만한 기울기를 갖는 문제에 대해서는 우수한 성능을 나타내지 못하는 문제가 나타났다. 이 문제를 해결하기 위해서, 진화를 하면서 최적해가 될 확률이 적은 소영역(cell)을 미리 제거하고 그 영역의 개체를 유력한 개체가 있는 영역으로 옮김으로써 보다 빨리 수렴하도록 하는 연구 등, 기타 여러 가지 방법에 대한 연구를 모색하고 있다.

참고문헌

[1] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary Computation: Comments on the History and Current State", IEEE Trans. On Evolutionary Computations pp. 3-17, 1997.

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

[3] J. H. Holland, "Outline for a Logical Theory of Adaptive Systems", Journal of the Association for Computing Machinary, 3:297-314, 1962.

[4] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison Wesley, Reading, MA, 1996.

[5] I. Rechenberg, "Cybernetic Solution Path of an Experimental Problem", Royal Aircraft Establishment, Library Translation, No. 1122, Farnborough, Hants, UK, Aug. 1965.

[6] H.-P. Schwefel, "Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strömungstechnik", Diplomarbeit, Technische Universität Berlin, 1965.

[7] D. B. Fogel, "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence", IEEE Press, Piscataway, NJ, 1995.

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

[9] H. J. Cho, S. Y. Oh, and D. H. Choi, "Fast Evolutionary Programming Through Search Momentum and Multiple Offspring Strategy", IEEE Int. Conf, on Evolutionary Computation, pp. 805-809, 1998.

[10] P. Adamidis and V. Petridis, "Co-operating Population with different evolution behaviors", IEEE Int. Conf. on Evolutionary Computation, pp. 188-191, 1996.

[11] J. Heitkoetter and D. Beasley, "The Hitch-Hiker's Guide to Evolutionary Computation", FAQ for comp.ai.genetic, 1997.

[12] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolutionary programmings", 3rd rev. and extended ed. Spring-Verlag, 1996.