코시 분포의 축척 매개변수를 추정하여 돌연변이 연산에 적용한 진화 프로그래밍

(Evolutionary Programming of Applying Estimated Scale Parameters of the Cauchy Distribution to the Mutation Operation)

이 창 용 †

(Chang-Yong Lee)

요 약 진화 프로그래밍은 실수형 최적화 문제에 널리 사용되는 알고리즘으로 돌연변이 연산이 중요한 연산이다. 일반적으로 돌연변이 연산은 확률 분포와 이에 따른 매개변수를 사용하여 변수값을 변화시키는데, 이 때 매개변수 역시 돌연변이 연산의 대상이 됨으로 이를 위한 또 다른 매개변수가 필요하다. 그러나 최적의 매개변수 값은 주어진 문제에 전적으로 의존하기 때문에 매개변수 개수가 많은 경우 매개변수 값들에 대한 최적 조합을 찾기 어렵다. 이러한 문제를 부분적으로나마 해결하기 위하여 본 논문에서는 변수의 돌연변이 연산을 위한 매개변수를 자기 적응적 관점에서 이론적으로 추정한 돌연변이 연산을 제안하였다. 제안한 알고리즘에서는 코시 확률 분포의 축척 매개변수를 추정하여 돌연변이 연산에 적용함으로 축척 매개변수에 대한 돌연변이 연산이 필요하지 않다는 장점이 있다. 제안한 알고리즘을 벤치마킹 문제에 적용한 실험 결과를 통해 볼 때, 최적값 측면에서는 제안한 알고리즘의 상대적 우수성은 벤치마킹 문제에 의존하였으나 계산 시간 측면에서는 모든 벤치마킹 문제에 대하여 제안한 알고리즘이 우수하였다.

키워드 : 진화 연산, 진화 프로그래밍, 코시 확률 분포, 돌연변이 연산, 최적화 문제

Abstract The mutation operation is the main operation in the evolutionary programming which has been widely used for the optimization of real valued function. In general, the mutation operation utilizes both a probability distribution and its parameter to change values of variables, and the parameter itself is subject to its own mutation operation which requires other parameters. However, since the optimal values of the parameters entirely depend on a given problem, it is rather hard to find an optimal combination of values of parameters when there are many parameters in a problem. To solve this shortcoming at least partly, if not entirely, in this paper, we propose a new mutation operation in which the parameter for the variable mutation is theoretically estimated from the self-adaptive perspective. Since the proposed algorithm estimates the scale parameter of the Cauchy probability distribution for the mutation operation, it has an advantage in that it does not require another mutation operation for the scale parameter. The proposed algorithm was tested against the benchmarking problems. It turned out that, although the relative superiority of the proposed algorithm from the optimal value perspective depended on benchmarking problems, the proposed algorithm outperformed for all benchmarking problems from the perspective of the computational time.

Key words : evolutionary computation, evolutionary programming, Cauchy probability distribution, *mutation operation, optimization problem*

† 정 회 원 : 공주대학교 산업시스템공학과 교수
clee@kongju.ac.kr
논문접수 : 2010년 4월 26일
심사완료 : 2010년 7월 19일

정보과학회논문지: 소프트웨어 및 응용 제37권 제9호(2010.9)

1. 서 론

진화 연산(evolutionary computation)은 지능적 계산(computational intelligence)이라고도 불리며, 생물체의 '자연 선택'과 '환경에 대한 적응'이라는 두 가지 개념을 컴퓨터에서 구현하여 실제 문제에 응용하고자 하는 계산 방법론이다. 메타 휴리스틱스에 속하는 진화 연산은

그 방법론에 있어 보편성을 띄고 있음으로 광범위한 적용 범위를 가지고 있으며 최근 병렬 연산과 연계되어 활발한 연구가 진행되고 있다. 특히 진화 연산은 후보 해(解)들의 집합인 모집단(population) 개념을 사용하여 여러 후보 해들 사이에 다양한 유형의 연산을 적용하여 집단적으로 최적해를 찾는 방법을 사용한다. 진화 연산의 대표적인 방법론으로는 유전 알고리즘(Genetic Algorithms)[1], 진화 전략(Evolutionary Strategies)[2], 진화 프로그래밍(Evolutionary Programming)[3], 유전 프로그래밍(Genetic Programming)[4], 그리고 분포 추정 알고리즘(*Estimation of Distribution Algorithm*)[5] 등을 들 수 있다. 이러한 방법론들은 주로 최적화 문제에 성공적으로 적용되어 탐색 공간 전체를 조사하지 않고 근사 최적해를 찾아주는 알고리즘으로 널리 사용되고 있다. 이 중에서도 진화 프로그래밍은 초기에 인공지능의 한 분야로 예측 문제의 해를 구하기 위하여 유한 상태 기계(finite state machine)를 진화시키는 방법[6]으로 시작하였으나, 80년대 후반에 D. Fogel에 의하여 더욱 개발되어 최근에는 실수형 다변수 함수의 최적화 문제나 공학적 문제에 많이 적용되고 있다.

진화 연산에서 사용되는 주요한 연산으로는 선택 연산(*selection operation*), 교차 연산(*crossover operation*), 그리고 돌연변이 연산(mutation operation) 등을 들 수 있는데, 이러한 연산들은 주로 생물체 혹은 생태계의 자연 선택과 진화 등의 개념들을 연산화 한 것이다. 교차 연산이 주된 연산인 유전 알고리즘 및 유전 프로그래밍과 달리 진화 전략 및 진화 프로그래밍에서는 돌연변이 연산이 주된 연산이며, 돌연변이 연산을 통하여 생성된 자손(offspring) 개체(individual, 혹은 후보 해)와 그 부모(parents) 개체들 중에서 적합도(fitness)에 의한 선택 연산을 적용하여 새로운 모집단을 생성하게 된다. 이렇게 생성된 새로운 모집단은 다음 세대를 구성하며 돌연변이 및 선택 연산을 각 세대에 반복적으로 수행하여 최적화의 과정으로 진화하게 된다.

진화 프로그래밍은 초기 60년대 L. Fogel 등에 의하여 인공 지능의 한 분야로 예측 문제를 해결하기 위하여 문제의 기초가 되는 구조를 진화시키기 위한 '유한 상태 기계'(finite state machine)에서 시작하였다[6]. 그 이후 진화 프로그래밍은 연속 함수 최적화 및 이산 조합 최적화 문제 등과 같은 다양한 데이터 구조를 처리할 수 있도록 확대되었으며[7], 이러한 연구에 힘입어 진화 프로그래밍은 실변수 함수의 최적화[8,9] 및 실용적 최적화 문제[10]에 성공적으로 적용되었다. 보다 정교한 진화 프로그래밍에 대한 연구가 90년대 D. Fogel 등에 의해 수행[11,12]되었는데, D. Fogel 등은 진화 프로그래밍의 돌연변이 연산을 위해 사용하는 정규 분포의 분산을 자기 적응적(self-adaptive)으로 변화시키는 방법을 제안하였다. 또한 N. Saravanan[13] 등은 진화 전략에서 사용하는 자기 적응적 연산[14]을 진화 프로그래밍에 도입하였으며, 자기 적응적 진화 프로그래밍은 현재 가장 널리 사용되는 방법론으로 자리 잡고 있다. 이러한 측면에서 볼 때, 진화 프로그래밍과 진화 전략은 비록 출발점은 서로 다르나 비슷한 알고리즘으로 간주될 수 있다.

일반적인 진화 프로그래밍에서는 돌연변이 연산을 위하여 정규 분포(Gaussian probability distribution)를 사용한다. 그러나 정규 분포는 유한한 분산(finite second moment)을 가짐으로 돌연변이 연산을 수행한 후 변수의 변화가 크지 않기 때문에 탐색 범위가 넓은 경우 정규 분포에 기초한 돌연변이 연산은 비효율적일 수 있다. 이러한 문제를 보완하기 위하여 코시 확률 분포(Cauchy probability distribution)와 코시 분포의 일반화된 형태인 레비 확률 분포(Lévy probability distribution)를 사용한 돌연변이 연산을 통한 진화 프로그래밍이 제안되었고[15,16], 또한 코시 분포를 사용한 진화 전략에 대한 연구도 진행되었다[17]. 코시와 레비 확률 분포는 분포의 이차 모멘트(second moment, 혹은 분산)가 무한인 분포임으로 돌연변이 연산 후 변수의 변화가 크게 될 확률이 정규 분포에 비하여 상대적으로 크다. 따라서 탐색해야할 변수의 영역이 넓은 경우(주로 지역 최적해가 많은 문제가 여기에 해당됨)에 효율적인 탐색 방법을 제공한다. 코시와 레비 분포를 사용한 돌연변이 연산을 적용한 실험 결과를 통해 볼 때, 정규 분포보다 코시 혹은 레비 분포를 통한 돌연변이 연산이 더 효율적임이 알려졌다[15,16]. 또한 진화 프로그래밍과 타부 탐색을 결합한 알고리즘에 대한 연구가 M. Ji 등에 의하여 수행[18]되었는데, 이 알고리즘은 진화 프로그래밍의 골격은 유지한 상태에서 타부 리스트를 추가적으로 도입하여 보다 효율적으로 변수 영역을 탐색하도록 고안된 알고리즘이다.

진화 프로그래밍의 응용 분야에 적용한 최근 연구들을 살펴보면 제약 조건이 주어진 경우 혹은 그렇지 않은 경우 모두에 대하여 경제 분야의 부하 발송 문제(load dispatch problem)에 적용된 연구[19,20]를 들 수 있다. 이 문제는 부하의 규모에 따라 복잡도가 비선형적으로 증가하는 비선형 최적화 문제로 기존의 기울기에 기초한 방법론(gradient-based method)이 적용되기 힘든 문제이다. 또한 진화 프로그래밍은 전력 시스템 최적화에도 적용되었는데, 전력 시스템 네트워크에서 최대 부하량(loadability)을 예측하기 위하여 임계치를 계산하는데 진화 프로그래밍을 사용한 연구[21]와 최대 부하 간선(buses)을 고려한 반응 전력 계획 수립을 위하여

진화 프로그래밍을 사용한 연구[22] 등을 들 수 있다. 또한 진화 프로그래밍 방법론은 다른 학습 방법론과 접목되어 연구되기도 하였는데, 대표적인 연구로 소비자의 행위를 예측하기 위한 한 방법으로 진화 프로그래밍을 제안한 연구[23]를 들 수 있다. 이 연구는 학습을 위한 방법론으로 진화 프로그래밍을 사용하고, 그 결과를 베이지안 네트워크(Bayesian network)에 적용하여 소비자의 행위를 예측한 연구이다.

진화 프로그래밍을 포함한 대부분의 메타 휴리스틱스 알고리즘은 매개변수를 포함하고 있고, 최적화 결과는 사용된 매개변수의 값에 의존한다. 또한 매개변수의 값은 최적화의 대상이 되는 함수에 따라 경험을 통해서 결정되는 경우가 일반적임으로, 매개변수가 많은 알고리즘일수록 매개변수 값들의 최적 조합을 찾는 것은 어려운 문제이다. 이러한 측면에서 볼 때, 비슷한 성능인 경우에는 적절한 가정 하에 상대적으로 적은 개수의 매개변수를 사용하는 알고리즘이 보다 유용하고 범용성 있는 알고리즘이라 할 수 있다.

진화 프로그래밍에서도 여러 매개변수가 사용되고 있는데, 이 중에서 중요한 매개변수로 돌연변이 연산을 위한 매개변수를 들 수 있다. 진화 프로그래밍에서는 두 가지 돌연변이 연산이 사용된다. 첫 번째 돌연변이 연산은 개체의 값을 변화시키는 돌연변이 연산으로 식 (12) 참고), 새로운 개체를 생성하기 위하여 기존 개체에 확률 분포에 따른 매개변수를 포함한 돌연변이 연산을 수행한다. 이 때 매개변수는 정규 분포를 사용하는 돌연변이 연산에서는 정규 분포의 표준편차에 해당하며, 코시 분포와 레비 분포의 경우에는 축척 매개변수(scale parameter)에 해당한다. 두 번째 돌연변이 연산은 매개변수 자체에 대한 돌연변이 연산으로, 일반적인 진화 프로그래밍에서는 이 매개변수의 값을 고정시키지 않고 돌연변이 연산을 적용하여 매개변수의 값을 변화시키는 방법을 사용한다(식 (13) 참고). 따라서 이 매개변수에 대한 돌연변이 연산을 위해 또 다른 매개변수가 필요하다.

본 논문에서는 기존 진화 프로그래밍에서 사용되는 매개변수의 개수를 최소화하여 보다 범용성 있는 알고리즘을 구현하기 위하여, 코시 확률 분포를 사용한 돌연변이 연산에서 사용되는 매개변수를 이론적으로 추정한 진화 프로그래밍을 제안한다. 제안한 알고리즘은 개체의 돌연변이 연산을 위한 매개변수를 추정하기 때문에, 이 매개변수에 대한 돌연변이 연산을 위해 또 다른 매개변수를 필요로 하지 않는다. 또한 제안한 알고리즘은 연산이 반복(iteration)됨에 따라 매개변수를 적절한 가정 하에 추정하기 때문에 주어진 최적화 문제에 대하여 실험을 통하여 최적의 매개변수 값을 구할 필요가 없는 특징을 가지고 있다. 제안한 진화 프로그래밍을 지역 최적해가 많은 벤치마킹(benchmarking) 문제에 적용하여 기존의 진화 프로그래밍 알고리즘과 성능 면에서 비교 분석하였다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 코시 확률 분포의 특성과 돌연변이 연산에서 사용되는 매개변수 추정을 위한 이론적 고찰을 하였고, 3장에서는 제안한 진화 프로그래밍 알고리즘을 기존 알고리즘과 비교하여 기술하였다. 4장에서는 제안한 알고리즘의 성능 평가를 위해 잘 알려진 벤치마킹 문제를 사용하여 기존 알고리즘과 성능을 비교하였으며, 5장에서는 본 논문의 요약과 결론을 맺었다.

2. 코시 확률 분포와 돌연변이 매개변수 추정

2.1 코시 확률 분포

로렌츠 분포(Lorentz distribution)라고도 알려진 코시 확률 분포[24]는 공명 혹은 공진(resonance) 현상을 설명하는데 많이 사용되는 분포로 아래와 같은 확률 밀도 함수(probability density function)를 가지고 있다.

$$f(x;x_0,\gamma)=\frac{1}{\pi}\left(\frac{\gamma}{(x-x_0)^2+\gamma^2}\right) \tag{1}$$

여기서 x_0는 위치 매개변수(location parameter)로 분포의 최대 확률 위치를 나타내며 중앙값(median)과 최빈값(mode)에 해당한다. 또한 γ는 축척 매개변수(scale parameter)로 반진폭 너비(half-width)를 나타낸다. 코시 분포의 가장 큰 특징 중 하나는 분포의 모든 모멘트(moment)가 무한대이기 때문에 평균과 분산 등이 유한한 값으로 정의되지 않다는 것이다. 이것은 유한의 평균과 분산을 가지는 정규 분포와 구별되는 것으로, 그림 1에서 볼 수 있듯이 분포의 꼬리 부분에서 확률이 정규 분포에 비해 상대적으로 크다. 특히 $x \gg \gamma$인 경우, 위의 식 (1)은 근사적으로 $f(x) \propto x^{-2}$으로 표현할 수 있으며, 이때 $f(x)$는 멱급수(power-law) 형태로 주어지기 때문에 $x \rightarrow bx$로 축척을 변화시켜도 분포의 모양인 $f(x)$는 변하지 않는, 축척 없는 분포(scale-free distribution)임을 쉽게 알 수 있다. 특히 $x_0=0$이고 $\gamma=1$인 경우를 표준 코시 확률 분포라 부르고 이 때 확률 밀도 함수는

$$f(x)=\frac{1}{\pi}\left(\frac{1}{x^2+1}\right) \tag{2}$$

로 주어진다.

코시 분포를 따르는 확률 변수는 일양 분포(uniform distribution)를 따르는 확률 변수를 생성한 후, 적절한 변환을 통하여 구할 수 있다. 즉, $[-\pi/2, \pi/2]$ 범위에서 일양 분포를 따르는 확률 변수 X를 생성하여 $Y=\gamma \tan(X)$의 변환을 하면 확률 변수 Y는 축척 매개변수가 γ이고 위치 매개변수 $x_0=0$인 코시 분포를

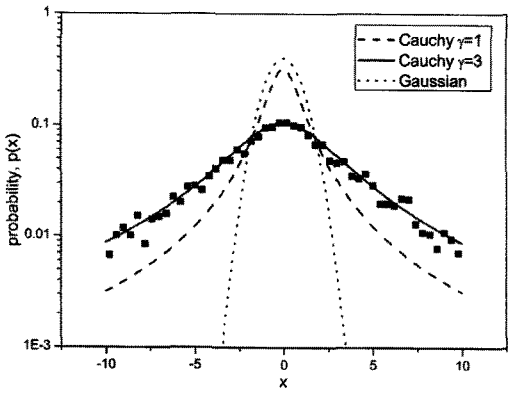

그림 1 표준 정규 분포(dotted line), 표준 코시 분포(dashed line), 그리고 $\gamma=3$인 경우의 코시 분포(solid line)의 확률 분포 비교. 코시 분포의 꼬리 부분의 확률이 정규 분포에 비해 상대적으로 큼을 알 수 있다. 확률 $p(x)$는 로그(log) 축척을 사용하였고, 네모 상자(■)는 $\gamma=3$인 코시 분포를 따르는 확률 변수를 생성하여 히스토그램(histogram)을 사용하여 구한 결과이다.

따른다[24]. 코시 분포를 사용한 돌연변이 연산에서는 후보 해를 중심으로 새로운 해를 생성하기 때문에 일반적으로 $x_0=0$으로 둔다. 그림 1의 네모 상자(■)는 위의 변환을 사용하여 $\gamma=3$인 경우 생성한 확률 변수를 통해 분포를 추정한 결과이다.

2.2 돌연변이 연산을 위한 매개변수 추정

돌연변이 연산은 크게 변수에 대한 돌연변이 연산과 확률 분포의 매개변수에 대한 돌연변이 연산 등 두 가지로 구성된다. 또한 기존의 돌연변이 연산에서는 코시 분포를 위한 축척 매개변수에 대한 돌연변이 연산을 사용하고 있다. 본 논문에서는 축척 매개변수에 돌연변이 연산을 적용하지 않고, 이론적으로 추정하는 방법을 제안한다. 제안한 방법은 개체의 돌연변이 연산에 필요한 축척 매개변수를 추정하여 사용하기 때문에 축척 매개변수에 대한 돌연변이 연산을 위해 경험적인 값을 필요로 하지 않는다.

진화 프로그래밍에서는 N개의 부모 개체(혹은 후보 해) $\vec{x_i}$, $(i=1,2,\cdots,N)$로 구성된 모집단에서 각 개체를 구성하는 m개의 변수 $x_i(j)$, $(j=1,2,\cdots,m)$에 대하여 돌연변이 연산을 적용하여 N개의 자손 개체를 생성한다. 개체에 대한 돌연변이 연산은 일반적으로 아래와 같이 표현할 수 있다.

$$x_i^*(j)=x_i(j)+\gamma_j\,C_j(0,1) \tag{3}$$

여기서 $x_i^*(j)$는 돌연변이 연산 후 생성된 자손 개체 i의 j번째 변수를 나타내고, $C_j(0,1)$는 돌연변이 연산을 위한 표준 코시 확률 변수를 나타내며, γ_j는 돌연변이 크기를 나타내는 축척 매개변수이다. 기존 진화 프로그래밍에서는 돌연변이 크기를 나타내는 축척 매개변수 γ_j의 초기값은 실험적 혹은 경험적으로 결정되나, 반복이 진행되면 γ_j에 대한 돌연변이 연산을 수행하여 그 값을 결정한다. 제안한 알고리즘에서는 돌연변이 크기를 나타내는 매개변수 γ_j를 다음과 같이 이론적인 고찰을 통하여 추정하고자 한다.

진화 프로그래밍에서 초기 모집단은 각 변수가 취할 수 있는 값의 영역 내에서 N개의 개체를 무작위로 생성하여 구성된다. 따라서 개체 i의 j번째 변수 $x_i(j)$가 가질 수 있는 값의 영역을 $[L_j, U_j]$로 표현하면 (여기서 L_j는 하한, U_j는 상한을 나타냄), 이 변수에 대하여 N개의 변수 값 $x_i(j)$, $i=1,2,\cdots,N$ 이 무작위로 $[L_j, U_j]$ 내에서 생성된다. 만약 N개의 변수 값이 $[L_j, U_j]$에 골고루 분포되어 있다고 가정하면, 각 변수 값은 평균적으로 같은 간격을 유지할 것이고, 그 간격을 $2\alpha_j$이라하면

$$2\alpha_j=\frac{U_j-L_j}{N} \tag{4}$$

으로 주어진다. 따라서 평균적으로 볼 때 각 개체의 변수는 그림 2에서 도식적으로 나타낸 것처럼 중복되지 않는 영역 $2\alpha_j$의 중심에 위치한다.

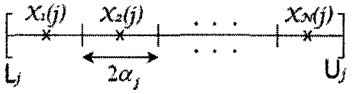

그림 2 개체를 구별하는 i가 클수록 변수 값이 크다는 가정하에 $x_i(j)$의 평균적 위치를 도식적으로 나타낸 그림

식 (3)의 매개변수 γ_j를 추정하기 위하여 돌연변이 연산을 통해서 생성되는 자손 개체의 변수값을 고려하자. 매개변수가 γ_j인 코시 분포를 사용한 돌연변이 연산을 통해서 생성되는 자손 $x_i^*(j)$가 부모 $x_i(j)$를 중심으로 영역 $2\alpha_j$내에 생성되는 경우는 두 가지가 있다. 첫째는 부모 $x_i(j)$에 대한 돌연변이 연산을 통하여 생성되는 경우이며, 둘째는 다른 부모 개체 $x_k(j)$ $(k\neq i)$에 의해 생성되는 경우이다. 첫 번째 경우는 자손 $x_i^*(j)$가 $x_i(j)$ 영역 내에 생성됨으로 돌연변이 연산 후 변수의 변화가 적은 경우에 해당하며, 이 경우가 발생할 확률을 p라 하면

$$p=Prob\{|x_i^*(j)-x_i(j)|\le\alpha_j\}=\frac{1}{\pi}\int_{x_i(j)-\alpha_j}^{x_i(j)+\alpha_j}\frac{\gamma_j\,dx}{(x-x_i(j))^2+\gamma_j^2} \tag{5}$$

로 표현할 수 있다. 두 번째 경우는 부모 개체 $x_k(j)$ $(k \neq i)$에 대한 돌연변이 연산으로 생성되는 경우로, $x_k(j)$ 입장에서 볼 때 자손 $x_i^*(j)$가 $x_k(j)$ 영역 내에서 생성되지 않음으로 연산 후 변수의 변화가 큰 경우에 해당한다. 두 번째 경우가 발생할 확률을 q라 하면 이 확률은

$$q = \sum_{k \neq i}^{N} Prob\left\{\left|x_k^*(j) - x_i(j)\right| \leq \alpha_j\right\} \quad (6)$$
$$= \frac{1}{\pi}\sum_{k \neq i}^{N}\int_{x_i(j)-\alpha_j}^{x_i(j)+\alpha_j} \frac{\gamma_j\, dx}{(x - x_k(j))^2 + \gamma_j^2}$$

로 나타낼 수 있다. 개체의 개수 N이 큰 경우, 위 식 (6)은 근사적으로

$$\sum_{k \neq i}^{N} Prob\left\{\left|x_k^*(j) - x_i(j)\right| \leq \alpha_j\right\} \approx 1 - p \quad (7)$$

임을 증명할 수 있다(Appendix에 자세한 증명이 있음). 즉, 개체 $x_i(j)$의 영역 내에 자손 개체 $x_i^*(j)$가 생성될 확률은 개체 $x_i(j)$에 의해 생성될 확률이 p이고 다른 개체들 $x_k(j)$ $(k \neq i)$에 의해 생성될 확률이 $1-p$이다.

매개변수 γ_j를 추정하기 위해서는 식 (5)의 확률 p를 결정해야 한다. 이 확률 p는 돌연변이 연산을 통하여 생성되는 자손 $x_i^*(j)$가 부모 $x_i(j)$를 중심으로 '반경' α_j 내에 존재할 확률이다. 이것은 개체 i의 입장에서 볼 때, $x_i(j)$에서 비교적 가까운 공간을 탐색하는 것으로 간주할 수 있다. 반면 식 (7)의 확률 $1-p$는 $k \neq i$인 모든 부모 $x_k(j)$에 의하여 생성되는 자손 $x_k^*(j)$가 부모 $x_i(j)$를 중심으로 반경 α_j 내부에 존재할 확률이다. 따라서 $k \neq i$인 개체 k 입장에서 보면, 이 확률은 돌연변이 연산 후 생성되는 자손 $x_k^*(j)$가 부모 $x_k(j)$를 중심으로 '반경' α_j 외부에 존재할 확률에 해당하기 때문에, 이 경우는 돌연변이 연산을 통해 생성되는 자손 $x_k^*(j)$가 부모 $x_k(j)$에서 멀리 떨어진 공간을 탐색하는 것으로 간주할 수 있다. 즉, 모집단 전체 관점에서 볼 때, p는 부모 개체에서 비교적 가까운 공간을 탐색할 확률이고, $1-p$는 부모 개체에서 비교적 먼 공간을 탐색할 확률에 해당한다.

탐색 공간을 골고루 탐색하기 위하여 변수의 변화가 큰 경우와 작은 경우 모두 동일한 확률로 자손 개체를 생성한다고 가정하면, $p \approx 1-p$가 되어야 함으로 $p \approx \frac{1}{2}$이 된다. 또한 이 확률은 변수의 변화가 큰 경우와 작은 경우가 동시에 일어날 확률, 즉 $p(1-p)$를 최대로 한다. $p = \frac{1}{2}$일 때, 식 (5)는 다음과 같이 표현된다.

$$\frac{1}{2} = \frac{1}{\pi}\int_{x_i(j)-\alpha_j}^{x_i(j)+\alpha_j} \frac{\gamma_j\, dx}{(x - x_i(j))^2 + \gamma_j^2} = \frac{2}{\pi}\tan^{-1}\left(\frac{\alpha_j}{\gamma_j}\right) \quad (8)$$

식 (8)을 만족하기 위해서 $\tan^{-1}\left(\frac{\alpha_j}{\gamma_j}\right) = \frac{\pi}{4}$, 즉 $\gamma_j = \alpha_j$가 되어야 함으로 축척 매개변수는 식 (4)에 의하여

$$\gamma_j = \frac{U_j - L_j}{2N} \quad (9)$$

으로 추정할 수 있다. 따라서 무작위로 생성된 초기 모집단을 사용하여 반복 $g=1$에서 돌연변이 연산을 위한 코시 분포의 매개변수는 식 (9)로 주어진다.

g번 반복(iteration)에서 축척 매개변수 $\gamma_j(g)$는 다음과 같이 추정할 수 있다. g번 반복 동안 생성된 총 개체(부모와 자손을 포함) 수는 gN이다. 이 때 각 개체가 변수의 영역 내에서 차지하는 중복되지 않는 영역을 $2\alpha_j(g)$라 하면, $2\alpha_j(g)$는 평균적으로

$$2\alpha_j(g) = \frac{U_j - L_j}{gN} \quad (10)$$

로 주어진다. 따라서 위의 $g=1$인 경우에 γ_j를 구한 것과 동일한 논리를 적용하면 $\gamma_j(g) = \alpha_j(g)$임으로 $\gamma_j(g)$는 아래와 같이 주어진다.

$$\gamma_j(g) = \frac{U_j - L_j}{2gN} \quad (11)$$

식 (11)을 통해 볼 수 있듯이 추정된 축척 매개변수는 변수의 상한과 하한의 차이에 비례하고, 반복과 모집단 개수에 반비례한다. 일반적으로 모집단의 개수 N은 알고리즘이 진행되는 동안에 고정된 값임으로 축척 매개변수는 반복에 반비례한다. 따라서 추정된 매개변수 $\gamma_j(g)$를 사용하여 돌연변이 연산을 수행하면 반복의 초기에는 돌연변이 연산을 통한 변수의 변화가 클 확률이 상대적으로 높고, 반복이 진행되는 동안 변수의 변화가 적을 확률이 점차적으로 커지게 된다.

돌연변이 연산 후 변수의 변화 정도와 반복 사이의 상관관계에 대한 연구[15,16]을 통해 볼 때, 반복의 초기에는 변수의 변화가 큰 개체가 보다 우수한 반면, 반복의 후반에는 변수의 변화가 적은 개체가 보다 우수함을 알 수 있다. X. Yao 등[15]은 각 부모 개체에 대하여 정규 분포와 코시 분포를 사용한 돌연변이 연산 각각을 수행하여 2개의 자손 개체를 생성한 후, 그 중에서 우수한 개체를 선택하는 방법을 취하였다. 이 방법을 사용한 실험 결과를 통해 볼 때, 반복의 초기에는 코시 분포가 보다 우수한 개체를 생성하였고 반복의 후반으로 갈수록 정규 분포가 우수한 개체를 생성함을 보였다. 또한 C. Lee 등[16]은 레비 확률 분포를 사용한 돌연변이 연산에서 변수의 변화 정도를 결정하는 매개변수 α(α 값이 클수록 변수의 변화가 적음)에 대하여 4개의 값 ($\alpha = 1.0, 1.3, 1.7, 2.0$)을 적용하여 그 중에서 가장 우수한 자손을 선택하는 방법을 취하였다. 이 경우에도 반복

의 초기에는 $\alpha=1.0, 1.3, 1.7$과 같은 값들이 선택될 확률이 높았으나 반복이 진행되면서 $\alpha=2.0$을 사용한 돌연변이 연산이 선택될 확률이 높았다. 이러한 성향은 본 논문에서 추정한 척도 매개변수 값이 반복에 반비례적으로 작아지는 것과 일치한다.

3. 매개변수를 추정한 진화 프로그래밍 알고리즘

본 논문에서 제안한 축척 매개변수 추정의 유용성을 검정하기 위하여 X. Yao[15] 등이 제안한 코시 분포를 사용한 진화 프로그래밍을 기초로 본 논문에서 제안한 축척 매개변수를 사용한 돌연변이 연산을 적용하였다. 일반적으로 실함수 최적화 문제는 다음과 같이 표현할 수 있다.

$$\text{find } \min f(\vec{x}) \text{ where}$$
$$\vec{x}=\{x(1), x(2), \cdots, x(m)\} \in S \text{ and}$$
$$S=\{x(j) \in R, L_j \le x(j) \le U_j, j=1, 2, \cdots, m\} \subseteq R^m$$

여기서 $f(\vec{x})$는 최적화 대상이 되는 실함수이고, $\vec{x}$는 m개의 변수로 구성된 벡터이며, $x(j)$는 j번째 변수를 나타내며 $j=1, 2, \cdots, m$이다. 또한 S는 각 변수 $x(j)$에 대한 상한(upper limit) U_j와 하한(lower limit) L_j 값으로 구성된 변수 영역의 집합이다. 진화 프로그래밍에서는 각 변수에 대하여 N개의 독립된 개체를 생성하여 돌연변이 및 선택 연산을 수행함으로, 각 개체를 구별하기 위하여 첨자 (subscript) i를 사용한다. 즉, 각 변수 $x(j)$가 생성한 N개의 개체를 $x_i(j)$ $i=1, 2, \cdots, N$으로 표현한다.

또한 T. Back 등은 돌연변이 연산 수행 시 확률 분포의 매개변수를 고정시키는 것 보다 매개변수 자체에 돌연변이 연산을 적용하여 환경에 적응하도록 변화시키는 것이 일반적으로 더 좋은 결과를 낳음을 보였다[25]. 따라서 본 연구에서도 환경에 적응하는 매개변수를 사용한다. 본 논문에서 제안한 방법론의 유용성을 검증하기 위하여 기존 방법론과 비교가 필요한데, 이를 위하여 X. Yao 등[15]이 제안한 알고리즘 중 코시 분포를 사용한 빠른 진화 프로그래밍(Fast Evolutionary Programming)을 사용한다. X. Yao 등[15]이 제안한 빠른 진화 프로그래밍 알고리즘은 아래와 같다.

1 단계: N개의 개체로 구성된 초기 모집단을 무작위로 형성하고, 반복(iteration) 횟수 $g=1$로 정한다. 각 개체는 실수 값을 가지는 벡터 쌍인 $(\vec{x_i}, \vec{\eta_i})$ $(i=1, 2, \cdots, N)$로 표현한다. 여기서 $\vec{x_i}$는 변수들을 벡터로 표현한 것으로 $x_i(j), j=1, 2, \cdots, m$이며, $x_i(j)$는 $[L_j, U_j]$ 영역 내에서 무작위로 생성한다. $\vec{\eta_i}$는 각 변수에 대하여 돌연변이 연산을 위한 매개변수이며 $\eta_i(j), j=1, 2, \cdots, m$이고, 초기 $\eta_i(j)$ 값은 고정된 값으로 $\eta_i(j)=SD$로 정한다.

2 단계: 각 부모 개체 $(\vec{x_i}, \vec{\eta_i})$에 대하여 그 자손 $(\vec{x_i^*}, \vec{\eta_i^*})$을 아래와 같이 벡터 쌍을 구성하는 각 변수 $(j=1, 2, \cdots, m)$에 대하여 생성한다.

$$x_i^*(j)=x_i(j)+\eta_i(j)\, C_j(0,1) \tag{12}$$

$$\eta_i^*(j)=\eta_i(j)\exp(\tau' N(0,1)+\tau N_j(0,1)) \tag{13}$$

여기서 $x_i(j)$와 $x_i^*(j)$은 각각 벡터 $\vec{x_i}$와 $\vec{x_i^*}$의 j번째 성분(즉, j번째 변수)을 나타내고, $\eta_i(j)$와 $\eta_i^*(j)$은 각각 벡터 $\vec{\eta_i}$와 $\vec{\eta_i^*}$의 j번째 변수를 나타낸다. 여기서 $C_j(0,1)$은 성분 j에 대하여 새롭게 생성된 표준 코시 분포를 따르는 확률 변수를 나타내고, $N(0,1)$은 표준 정규 분포를 따르는 확률 변수를 나타내며, $N_j(0,1)$는 각 성분 j에 대하여 새롭게 생성된 확률 변수를 나타낸다. 그리고 매개변수 τ와 τ'은 일반적으로 다음과 같이 경험적으로 정의된다[25].

$$\tau=\frac{1}{\sqrt{2\sqrt{m}}},\quad \tau'=\frac{1}{\sqrt{2m}} \tag{14}$$

3 단계: 각 부모 개체 $(\vec{x_i}, \vec{\eta_i})$와 자손 개체 $(\vec{x_i^*}, \vec{\eta_i^*})$에 대하여 최적화 대상이 되는 적합도 함수 $f_1, f_2, \cdots, f_{2N}$을 계산한다. 여기서 적합도 함수는 최적 실함수 값에 해당한다.

4 단계: 생성된 $2N$개의 적합도 함수에 대하여 N개의 개체로 구성된 다음 세대를 위한 선택 연산을 수행한다. 먼저 각 개체에 대하여 승리 함수(winning function) $w_i=0$, $i=1, 2, \cdots, 2N$으로 초기화한다. 각 개체의 적합도 함수 f_i, $i=1, 2, \cdots, 2N$에 대하여 1개의 다른 적합도 함수를 임의로 선택하여 두 적합도 함수를 비교한다. 만약 고려하는 개체의 적합도 함수가 비교되는 적합도 함수보다 작으면 $w_i=w_i+1$으로 정의한다. 이러한 비교를 Q번하게 되면 각 개체의 승리함수 w_i는 0에서 Q사이의 정수가 된다. 이러한 과정을 $2N$개의 모든 적합도 함수에 대하여 실행한다.

5 단계: 승리 함수 w_i, $i=1, 2, \cdots, 2N$에 대하여 함수의 값이 큰 것부터 N개를 선택하고 선택된 개체들을 다음 세대의 부모로 구성된 모집단 $(\vec{x_i}, \vec{\eta_i})$ $(i=1, 2, \cdots, N)$으로 정하고, 가장 최적의 적합도 함수를 저장한다.

6 단계: $g \leftarrow g+1$로 하고, 정해진 횟수 혹은 정한 기준이 만족될 때까지 2 단계에서 5단계까지 되풀이한다.

본 연구에서 제안한 돌연변이 연산을 사용한 진화 프로그래밍 알고리즘은 위의 기존 알고리즘의 골격은 유지하면서 식 (12)의 개체에 대한 돌연변이 연산에 추정

한 매개변수를 사용한다. 또한 기존 진화 프로그래밍에서 사용하던 매개변수에 대한 돌연변이 연산을 수행하지 않기 때문에 식 (13)에 해당하는 부분이 없고, 따라서 계산 시간이 줄어드는 장점이 있다. 따라서 매개변수를 추정한 진화 프로그래밍의 순서는 기존 진화 프로그래밍과 유사하고 다만 위 알고리즘의 2 단계를 아래와 같이 변경한다.

2 단계: 각 부모 개체 $\overrightarrow{x_i}$ 에 대하여 그 자손 $\overrightarrow{x_i^*}$ 을 추정한 매개변수를 사용하여 각 성분$(j=1,2,\cdots,m)$에 대하여 다음과 같이 생성한다.

$$x_i^*(j) = x_i(j) + \gamma_j(g)\, C_j(0,1) \qquad (15)$$

여기서 매개변수 $\gamma_j(g)$는 식 (11)을 만족한다.

4. 실험 결과 및 분석

제안한 돌연변이 연산의 효과를 살펴보기 위하여 지역 최적해가 많은 벤치마킹 문제들을 선정하여 기존 진화 프로그래밍 방법과 비교하였다. 선정한 벤치마킹 함수들은 진화 프로그래밍에서 표준적으로 사용되는 함수들이며, 지역최적해가 많은 함수$(f_1, f_2, f_3, f_4, f_5)$와 적은 함수$(f_6, f_7, f_8)$ 등 두 가지 유형으로 구분하였다. 표 1은 본 실험에서 사용한 벤치마킹 함수 및 각 함수에 대한 변수 개수와 그 영역을 나타낸 것이다. 일반적으로 볼 때 어느 특정 알고리즘이 모든 최적화 문제에 우수한 성능을 나타낸다고 주장하기 어렵다[26]. 본 논문에서 제안한 알고리즘 역시 모든 함수에 대하여 기존의 진화 프로그래밍 알고리즘에 비하여 상대적으로 우수하다고 할 수 없다. 다만 벤치마킹 문제를 통하여 어떤 경우에 제안한 알고리즘이 보다 나은 효과를 나타내며, 또한 제안한 알고리즘의 장점은 무엇인가를 분석하고자 한다. 본 실험에서 사용한 매개변수는 아래와 같다.

- 돌연변이 연산의 초기 분산 $SD=3.0$
- 모집단 개수 $N=100$
- 승리 함수를 위한 토너먼트 크기 $Q=10$
- 반복 횟수 $g=5000$

위의 매개변수 값들은 진화 프로그래밍에서 통상적으로 사용하는 값들이며, 반복 횟수는 두 알고리즘을 사용한 결과가 충분히 수렴할 수 있는 정도로 정하였다.

그림 3은 지역최적해의 개수가 많은 벤치마킹 함수들에 대하여 제안한 돌연변이 연산과 기존 돌연변이 연산에서 반복에 따른 함수의 최소값 변화를 나타낸 것이다. 그림 3에서 볼 수 있듯이 벤치마킹 함수 f_1과 f_2의 경우에는 기존 돌연변이 연산이 나은 결과를 보였으며, 함수 f_3, f_4, 그리고 f_5인 경우에는 반복에 기초한 돌연변이 연산이 보다 우수한 결과를 나타내었다. 특히 함수 f_3, f_4, 그리고 f_5의 경우, 반복의 초기에는 기존 돌연변이 연산이 빠른 수렴 속도를 보이나, 반복이 거듭될수록 제안한 돌연변이 연산이 보다 나은 최적해를 찾는 것을 알 수 있다. 이것은 반복의 초기 단계에는 보다 넓은 변수 영역을 탐색하는 것이 중요하고, 반복이 거듭될

표 1 실험에 사용한 벤치마킹 함수들. f_1, f_2, f_3, f_4, f_5은 지역 최적해가 많은 함수들이고, f_6, f_7, f_8은 지역 최적해가 적은 함수들이다. m은 변수의 개수를 나타내고 S는 변수의 영역[하한, 상한]을 나타낸다.

벤치마킹 함수	m	S
$f_1 = -\sum_{i=1}^{m} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	$[-500, 500]^m$
$f_2 = \sum_{i=1}^{m} \left\{x_i^2 - 10\cos(2\pi x_i) + 10\right\}$	30	$[-5.12, 5.12]^m$
$f_3 = -20\exp\left(-0.2\sqrt{\frac{1}{m}\sum_{i=1}^{m} x_i^2}\right) - \exp\left(\frac{1}{m}\sum_{i=1}^{m}\cos(2\pi x_i)\right) + 20 + e$	30	$[-32, 32]^m$
$f_4 = \frac{1}{4000}\sum_{i=1}^{m} x_i^2 - \prod_{i=1}^{m}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^m$
$f_5 = 0.1\left\{\begin{array}{l}\sin^2(3\pi x_1) + \sum_{i=1}^{m-1}(x_i-1)^2\left\{1+\sin^2(3\pi x_{i+1})\right\} + (x_m-1)^2 \\ \left\{1+\sin^2(2\pi x_m)\right\}\end{array}\right\} + \sum_{i=1}^{m} u(x_i, 5, 100, 4)$	30	$[-50, 50]^m$
$f_6 = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5.0, 5.0]^m$
$f_7 = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-2.0, 2.0]^m$
$f_8 = -\sum_{i=1}^{5}\left\{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i\right\}^{-1}$	4	$[0.0, 10.0]^m$

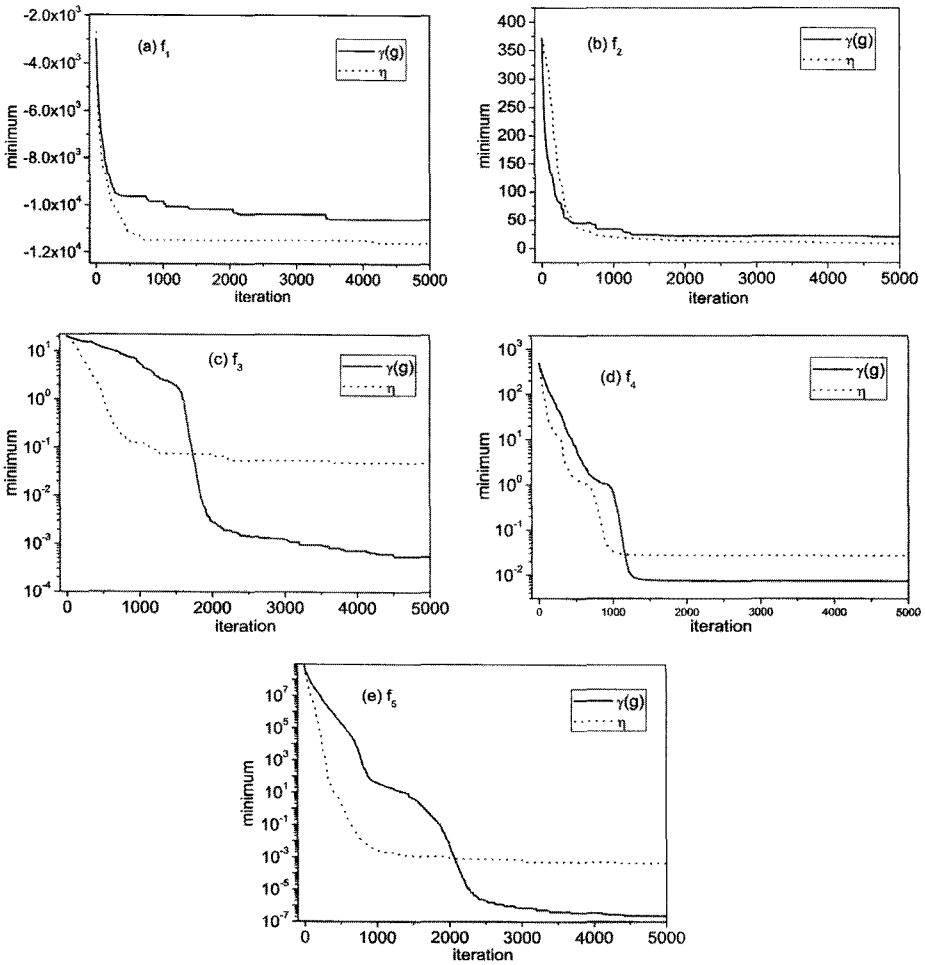

그림 3 표 1의 지역최적해가 많은 함수에 대하여 구한 반복에 따른 최적값의 변화. 그림 (a), (b), (c), (d), (e)는 각각 표 1의 함수 f_1, f_2, f_3, f_4, f_5 에 대하여 구한 최적값의 결과이다. 실선(solid line)은 제안한 돌연변이 연산을 사용한 경우를 나타내고 점선(dotted line)은 기존 돌연변이 연산의 경우를 나타낸다.

수록 우수한 해 근처를 집중적으로 탐색하는 것이 나음을 입증하는 것이라 할 수 있다.

벤치마킹 함수에 따른 실험 결과의 차이는 다음과 같이 이해할 수 있다. 벤치마킹 함수 f_1 과 f_2 의 경우에는 지역 최적해들이 산발적으로 존재하는 반면, 함수 f_3, f_4 와 f_5 의 경우에는 지역최적해가 많이 존재하나, 함수의 전반적인 모양을 볼 때 지역 최적해들은 하나의 전역최적해로 수렴하는 특징을 보이고 있다. 또한 함수 f_1 과 f_2 의 경우에는 지역 최적해의 '깊이'가 함수 f_3, f_4 와 f_5 에 비하여 상대적으로 깊은 특징을 가지고 있다. 지역 최적해의 깊이가 깊은 경우에는 지역 최적해를 극복하기 위한 장벽(barrier)이 크기 때문에 변수의 변화가 큰 돌연변이 연산이 보다 효율적이고, 장벽이 크지 않는 경우에는 변수의 변화가 적은 연산이 상대적으로 더 효율적이다.

제안한 돌연변이 연산에서 축척 매개변수는 반복에 반비례적으로 감소함으로, 평균적으로 볼 때, 반복이 진행될수록 돌연변이 연산 후 변수의 변화가 적다. 따라서 제안한 돌연변이 연산은 반복이 진행되면서 변수의 가까운 주변을 탐색할 확률이 기존 돌연변이 연산에 비하여 크기 때문에 지역 최적해의 장벽이 상대적으로 작은 f_3, f_4 와 f_5 에 보다 효과적으로 적용될 수 있으며, 상대적으로 큰 장벽을 가진 f_1 과 f_2 에 대해서는 비효율적임을 알 수 있다.

그림 4는 지역최적해의 개수가 상대적으로 적은 함수 f_6, f_7, f_8 에 대하여 제안한 돌연변이 연산과 기존 돌연

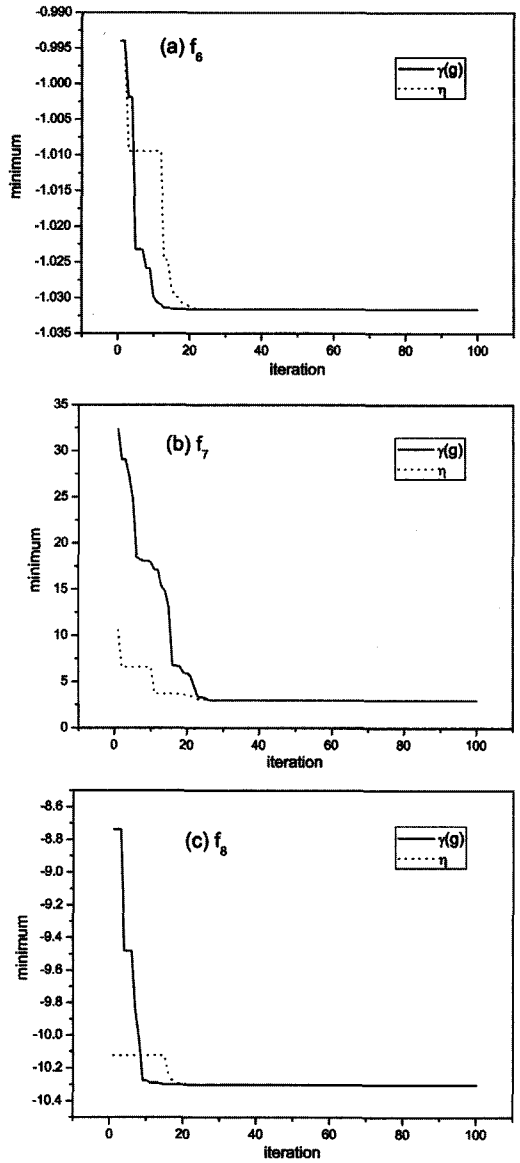

그림 4 표 1의 지역최적해가 적은 함수에 대하여 구한 반복에 따른 최적값의 변화. 그림 (a), (b), (c)는 각각 표 1의 함수 f_6, f_7, f_8에 대하여 구한 최적값의 결과이다. 실선(solid line)은 제안한 돌연변이 연산을 사용한 경우를 나타내고 점선(dotted line)은 기존 돌연변이 연산의 경우를 나타낸다.

변이 연산에서 반복에 따른 함수의 최소값 변화를 나타낸 것이다. 그림 4에서 볼 수 있듯이 지역 최적해의 개수가 적은 함수의 경우에는 제안한 돌연변이 연산과 기존 돌연변이 연산간의 차이가 거의 없다. 이것은 고려하는 벤치마킹 함수가 소수의 지역 최적해를 가짐으로 인하여 지역 최적해가 많은 함수에 비하여 상대적으로 전역최적해로 수렴할 확률이 높기 때문이다.

제안한 돌연변이 연산과 기존 돌연변이 연산을 각 벤치마킹 함수에 대하여 30번 독립적으로 최적해를 구하였으며, 그 결과를 표 2에 요약하였다. 표 2를 통해 볼 수 있듯이, 제안한 돌연변이 연산의 최적해에 대한 성능은 벤치마킹 함수에 의존적이다. 평균적으로 볼 때 함수 f_1과 f_2의 경우에는 기존 돌연변이 연산이 나은 결과를 보였으며, 함수 f_3, f_4, 그리고 f_5인 경우에는 제안한 돌연변이 연산이 보다 우수한 결과를 나타내었다. 또한 함수 f_6, f_7, f_8의 경우에는 두 방법 모두 유사한 결과를 나타내었다. 이 함수들은 지역최적해가 적은 함수로 전역 최적해를 비교적 쉽게 찾을 수 있기 때문에 두 방법 간의 차이가 거의 없음을 알 수 있다. 그러나 제안한 알고리즘은 매개변수를 위한 돌연변이 연산을 수행하지 않기 때문에 두 알고리즘이 비슷한 성능이라고 가정한다면 매개변수를 적게 사용하는 알고리즘이 보다 보편적이고, 최적화를 위한 계산 시간 측면에서도 장점을 가지고 있다고 할 수 있다.

표 2에 나타난 두 알고리즘의 성능을 비교 분석하기 위하여 통계적 검정을 실시하였다. 통계적 검정은 각 벤치마킹 함수에 대하여 독립적으로 실시하였고, 동일한 함수에 대하여 각기 다른 돌연변이 연산을 사용하여 구한 최적값의 평균에 대한 검정을 실시하였다. 검정을 위하여 설정한 귀무가설(null hypothesis) H_0와 대립가설(alternative hypothesis) H_1은 다음과 같다.

$$H_0: \mu_{iteration} = \mu_{parameter} \quad H_1: \mu_{iteration} < \mu_{parameter} \tag{16}$$

여기서 $\mu_{iteration}$와 μ_{gauss}는 각각 제안한 돌연변이 연산과 기존 돌연변이 연산을 사용하여 구한 최적값의 평균을 나타낸다. 즉, 귀무가설은 두 방법의 결과가 통계적으로 차이가 없다는 것이고, 대립가설은 벤치마킹 함수를 최적화하는데 제안한 돌연변이 연산을 사용하는 것이 기존의 방법 보다 우수하다는 것이다.

본 실험에서는 각 벤치마킹 함수에 대하여 30번 독립적인 실행을 하였음으로 표본의 개수가 30개에 해당하기 때문에 t-검정[27]을 실시하여야 한다. t-검정은 일반적으로 두 모집단의 분산이 동일한 경우와 그렇지 않는 경우 등 두 가지로 구분한다. 표 2를 통해 볼 때 f_1, f_6, f_7, f_8은 두 방법의 결과에 대한 표본 표준 편차에 큰 차이가 없으나, $f_2 - f_5$는 두 방법의 결과에 대한 표본 표준 편차에 큰 차이가 있기 때문에, f_1, f_6, f_7, f_8은 '모집단의 분산이 동일하다'는 가정 하에 검정을 실시해야 하고 $f_2 - f_5$는 '모집단의 분산이 동일하지 않다'는 가정 하에 검정을 실시하여야 한다.

표 2 각 함수에 대한 최적해와 통계 분석 결과. 실험 결과는 각 돌연변이 연산에 대하여 30번의 독립적 시행 후 구한 최적해의 평균과 표본 표준편차(괄호안), 그리고 t-검정을 위해 구한 검정 통계량 t_0, 자유도 df 및 유의 수준 $\alpha=0.05$에서 기각역 $t_\alpha(df)$ 등으로 구성되어 있다.

벤치마킹 함수	제안한 돌연변이 연산 평균 (표준편차)	기존 돌연변이 연산 평균 (표준편차)	t_0	df	기각역 $t_\alpha(df)$
f_1	$-10363.49(396.91)$	$-11621.97(284.69)$	13.54	58	-1.67
f_2	$26.70(8.04)$	$9.21(1.86)$	11.60	32	-1.70
f_3	$5.55\times10^{-4}(3.13\times10^{-5})$	$5.12\times10^{-2}(4.13\times10^{-3})$	-67.17	29	-1.69
f_4	$1.15\times10^{-2}(1.39\times10^{-2})$	$2.09\times10^{-2}(2.65\times10^{-2})$	-1.71	43	-1.68
f_5	$1.92\times10^{-7}(3.27\times10^{-8})$	$5.47\times10^{-4}(7.95\times10^{-5})$	-37.67	29	-1.69
f_6	$-1.03(<10^{-9})$	$-1.03(2.40\times10^{-9})$	0.0	58	-1.67
f_7	$3.00(<10^{-9})$	$3.00(7.37\times10^{-8})$	0.0	58	-1.67
f_8	$-7.80(3.02)$	$-8.22(2.85)$	0.55	58	-1.67

모집단의 분산이 동일하다는 가정을 한 경우, 검정통계량은

$$t_0=\frac{\overline{X_1}-\overline{X_2}}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} \tag{17}$$

로 주어진다. 여기서 $\overline{X_1}$, $\overline{X_2}$는 각각 제안한 돌연변이 연산과 기존 돌연변이 연산을 사용한 결과에 대한 표본 평균이며, n_1, n_2는 표본 개수 ($n_1=n_2=10$), 그리고 S_p는 등분산 가정에 대한 합동추정량으로 $S_p=\sqrt{\frac{(n_1-1)S_1^2+(n_2-1)S_2^2}{n_1+n_2-2}}$ 이고, S_1, S_2는 각각의 표본 표준 편차를 나타낸다. 또한 이 경우 자유도(degree of freedom) $df=n_1+n_2-2=18$로 주어진다. 모집단의 분산이 동일하지 않다는 가정 하에 검정통계량은

$$t_0=\frac{\overline{X_1}-\overline{X_2}}{\sqrt{\frac{S_1^2}{n_1}+\frac{S_2^2}{n_2}}} \tag{18}$$

로 주어지며, 이 경우 검정통계량 t_0는 자유도가

$$df=\frac{\left(\frac{S_1^2}{n_1}+\frac{S_2^2}{n_2}\right)^2}{\left(\frac{S_1^2}{n_1}\right)^2/(n_1-1)+\left(\frac{S_2^2}{n_2}\right)^2/(n_2-1)} \tag{19}$$

인 t-분포로 근사시킬 수 있음으로 이를 이용하여 t-검정을 시행할 수 있다.

유의 수준 $\alpha=0.05$에서 검정을 실시한 결과를 표 2에 나타내었다. 표 2에서 볼 수 있듯이 함수 f_1과 f_2의 경우, 검정통계량 t_0는 기각역(critical region) t_α보다 큼으로(즉, $t_0>t_\alpha$) 식 (16)의 귀무가설을 채택할 수 있다. 이것은 제안한 돌연변이 연산이 기존 돌연변이 연산에 비하여 우수한 성능을 가진다고 주장할 수 있는 근거가 없다는 것을 의미한다. 이와 대조적으로 함수 f_3, f_4와 f_5의 경우에는 검정통계량 t_0가 기각역 t_α보다 작음으로 (즉, $t_0<t_\alpha$), 식 (16)의 귀무가설 H_0를 기각하고 대립가설 H_1을 채택할 수 있다. 즉, 제안한 돌연변이 연산이 기존 돌연변이 연산보다 성능 면에서 우수함을 통계적으로 입증할 수 있음을 의미한다. 또한 지역 최적해가 적은 f_6, f_7, f_8의 경우에는 검정통계량 t_0가 $t_0=0$이거나 $t_0=0$에 가까움으로 이 경우 역시 제안한 돌연변이 연산이 기존 돌연변이 연산에 비하여 우수한 성능을 가진다고 주장할 수 있는 근거가 없다.

제안한 돌연변이 연산의 장점 중 하나는 같은 실험 조건인 경우에는 제안한 돌연변이 연산을 실행하는데 소요되는 계산 시간이 기존 돌연변이 연산에 비하여 적게 걸린다는 점이다. 그 이유는 기존 알고리즘에서 식 (13)으로 표현되는 매개변수에 대한 돌연변이 연산이 필요하지 않기 때문이다. 같은 실험 조건(반복 횟수, 승리 함수를 위한 토너먼트 크기, 모집단 개수 등) 하에서 두 돌연변이 연산을 시간복잡도(time complexity) 측면에서 비교하면 다음과 같다. 기존의 축척 매개변수에 대한 돌연변이 연산은 N개의 개체를 구성하는 축척 매개변수 $\eta_i(j)$ ($i=1,2,\cdots,N$)에 대해서도 돌연변이 연산을 수행하기 때문에 표준 정규 분포를 따르는 확률 변수를 생성하는데 걸리는 시간 복잡도를 제외하더라도 $O(N)$의 시간 복잡도인 반면, 제안한 방법에서는 축척 매개변수를 위해 식 (11)을 사용하고 돌연변이 연산을 수행하지 않기 때문에 시간 복잡도는 $O(1)$이 된다. 따라서 반복 횟수가 G인 경우, 기존 방법론은 $O(GN)$의 시간 복잡도를 추가로 가지게 된다.

5. 요약 및 결론

본 논문에서는 일반적인 진화 프로그래밍에서 돌연변

이 연산에 사용되는 매개변수 값이 주어진 최적화 문제에 의존하는 단점을 해결하기 위하여 돌연변이 연산을 위한 매개변수를 적절한 가정을 통하여 추정하는 새로운 진화 프로그래밍을 제안하였다. 제안한 진화 프로그래밍 알고리즘은 돌연변이 연산을 위해 코시 확률 분포를 사용하며, 돌연변이 연산의 크기를 결정하는 코시 분포의 축척 매개변수를 이론적으로 추정한 알고리즘이다. 제안된 알고리즘은 축척 매개변수를 적절한 가정 하에 이론적으로 추정하여 사용하기 때문에 매개변수를 경험적으로 결정할 필요가 없는 장점을 가지고 있다. 또한 제안한 알고리즘은 기존 알고리즘과 달리 매개변수 자체에 대하여 돌연변이 연산을 수행하지 않기 때문에 최적해를 찾는데 걸리는 계산 시간이 단축되는 장점도 가지고 있다.

제안한 진화 프로그래밍의 성능을 평가하기 위하여 지역 최적해가 많은 벤치마킹 문제를 선정하여 기존의 진화 프로그래밍 알고리즘과 비교 분석하였다. 실험 결과를 통해 볼 때, 제안한 진화 프로그래밍 방법이 모든 벤치마킹 함수에서 우수한 성능을 보인 것은 아니나 일정 부분 유용함이 있음을 알 수 있었다. 또한 모든 벤치마킹 함수에 대하여 제안한 알고리즘은 기존 알고리즘에 비하여 함수의 최적화 과정에서 걸리는 계산 시간을 상당 부분 단축할 수 있는 장점이 있었다.

돌연변이 연산에 사용되는 매개변수를 추정하는 진화 프로그래밍 방법론에 관한 연구는 향후 많은 연구가 필요한 분야인데, 다음 두 가지 측면은 언급할 만하다. 첫째, 본 연구에서는 돌연변이 연산을 위한 매개변수를 주로 반복에 의존하도록 고안하였는데, 이 매개변수는 변수의 변화를 주도하는 중요한 요소임으로 반복외의 다른 환경에도 적응할 수 있도록 고안하는 연구가 필요하다. 둘째, 본 연구에서는 코시 확률 분포에만 국한하여 매개변수를 추정하였는데, 향후 정규 분포 및 레비 분포 등 다른 확률 분포에 같은 원리를 적용할 수 있는 이론적인 연구가 추가적으로 필요하다. 이러한 연구들은 향후 추정한 매개변수를 사용하는 최적화 알고리즘에 대한 연구를 활성화시킬 것으로 예상된다.

참 고 문 헌

[1] D. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*, Reading, MA, Addison-Wesley, 1989.

[2] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford Univ. Press, New York, 1996.

[3] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, New York, IEEE Press, 1995.

[4] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[5] J. Lozano, et al. (Eds.), *Towards a new evolutionary computation. Advances in estimation of distribution algorithms*, Springer, 2006.

[6] L. Fogel, A. Owens, and M. Walsh, *Artificial Intelligence Through Simulated Evolution*, New York, Wiley, 1966.

[7] D. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, New York, 1995.

[8] D. Fogel and J. Atmar, "Comparing Genetic Operators with Gaussian Mutation in Simulated Evolutionary Processes Using Linear Systems," *Biological Cybernetics*, vol.63, pp.111-114, 1990.

[9] D. Fogel and L. Stayton, "On the Effectiveness of Crossover in Simulated Evolutionary Optimization," *BioSystems*, vol.32, pp.171-182, 1994.

[10] A. Sebald and J. Schlenzig, "Minimax Design of Neural Net Controllers for Highly Uncertain Plants," *IEEE Trans. Neural Networks*, vol.5, pp.73-82, 1994.

[11] D. Fogel, L. Fogel, W. Atmar, and G. Fogel, "Meta-evolutionary programming," in R. Chen editor, *Proc. 25th Asilomar Conference on Signals, Systems and Computers*, pp.540-545, 1991.

[12] D. Fogel, *Evolving Artificial Intelligence*, Ph. D thesis, University of California, San Diego, CA, 1992.

[13] N. Saravanan and D. Fogel, "Learning of Strategy Parameters in Evolutionary Programming: An Empirical Study," In *Proc. of the third Ann. Conf. on Evolutionary Programming*, edited by A. Sebald and L. Fogel, River Edge, NJ, World Scientific, pp. 269-280, 1994.

[14] H.-P. Schwefel, *Numerical Optimization of Computer Models*, Chichester, UK, John Wiley, 1981.

[15] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol.3, pp.82-102, 1999.

[16] C.-Y. Lee and X. Yao, "Evolutionary programming using mutations based on the Lévy probability distribution," *IEEE Trans. Evol. Comput.*, vol.8, pp.1-13, 2004.

[17] X. Yao and Y. Lin, "Fast evolution strategies," *Contr. Cybern.*, vol.26, pp.467-496, 1997.

[18] M. Ji and J. Klinowski, "Taboo evolutionary programming: a new method of global optimization," *Proc. R. Soc. A*, vol.462, pp.3613-3627, 2006.

[19] N. Sinha, R. Chakrabarti, and P. Chattopadhyay, "Evolutionary programming techniques for economic load dispatch," *IEEE Trans. Evol. Comput.*, vol.7, pp.83-94, 2003.

[20] P. Somasundaram and K. Kuppusamy, "Application of evolutionary programming to security constrained economic dispatch," *International Journal of*

Electrical Power & Energy Systems, vol.27, pp.343-351, 2005.

[21] A. Minhat, I. Musirin, and M. Othman, "Evolutionary programming based technique for secure operating point identification in static voltage stability assessment," *J. of Artificial Intelligence*, vol.1, pp.12-20, 2008.

[22] S. Kumar1, R. Kumar, K. Thanushkodi, and P. Renuga., "Reactive Power Planning considering the highest load buses using Evolutionary Programming," *International Journal of Recent Trends in Engineering*, vol.2, pp.37-39, 2009.

[23] G. Cui, M. Wong, and H.-K. Lui, "Machine Learning for Direct Marketing Response Models: Bayesian Networks with Evolutionary Programming," *Management Science*, vol.52, pp.597-612, 2006.

[24] R, Hogg and A. Craig, *Introduction to mathematical statistics*, 4th Ed., Macmillan Publishing Co., Inc. New York, 1978.

[25] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evol. Comput.*, vol.1, pp.1-23, 1993.

[26] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol.1, pp.67-82, 1997.

[27] 김우철 등, *통계학 개론*, 영지문화사, 2000.

이 창 용

1983년 서울대학교 계산통계학과 졸업(이학사). 1995년 미국 텍사스 주립대학교(Univ. of Texas at Austin) 물리학과 졸업(이학박사). 1996년~1998년 한국전자통신연구원 선임연구원. 1998년~2007년 공주대학교 산업정보학과 교수. 2007년~현재 공주대학교 산업시스템공학과 교수. 관심분야는 진화 연산 알고리즘 및 최적화 문제, 복잡계 네트워크(complex networks), 생물정보학(bioinformatics) 등

Appendix:

$$\sum_{k\neq i}^{N} Prob\left\{\left|x_k^*(j)-x_i(j)\right|\le\alpha_j\right\}\approx 1-p \text{ 의 증명} \quad \text{(A1)}$$

증명. 편이 상 짝수의 개체로 모집단을 구성하고, 무작위로 선택된 N개의 변수를 크기에 따라 올림차순으로 나열하며, 각 개체를 x_k ($k=-N/2,-N/2+1,\cdots,N/2-1$, $N/2$)로 표현한다. 또한 편의상 변수를 나타내는 인덱스 j는 생략한다.

코시 확률 분포의 정의에 의하면 개체 $x_k(k\neq i)$에 의해 돌연변이 연산 후 생성된 개체 x_k^*가 개체 x_i의 반경 α내에 존재할 확률은 다음과 같이 주어진다.

$$Prob\left\{\left|x_k^*-x_i\right|\le\alpha\right\}=\frac{1}{\pi}\int_{x_i-\alpha}^{x_i+\alpha}\frac{\gamma\,dx}{(x-x_k)^2+\gamma^2} \quad \text{(A2)}$$

따라서 식 (A1)을 다음과 같이 표현할 수 있다. 즉,

$$\sum_{k\neq i}^{N} Prob\left\{\left|x_k^*-x_i\right|\le\alpha\right\}=\frac{1}{\pi}\sum_{k=-N/2}^{i-1}\int_{x_i-\alpha}^{x_i+\alpha}\frac{\gamma\,dx}{(x-x_k)^2+\gamma^2}+\frac{1}{\pi}\sum_{k=i+1}^{N/2}\int_{x_i-\alpha}^{x_i+\alpha}\frac{\gamma\,dx}{(x-x_k)^2+\gamma^2} \quad \text{(A3)}$$

식 (A3)에 대하여

$$y=x-x_k+x_i=x+2(i-k)\alpha \quad \text{(A4)}$$

로 치환하면, 식 (A3)의 우변 첫 번째 항은 변수 y에 대하여 위치 매개변수가 x_i 인 코시 분포의 적분이 된다. 즉,

$$\sum_{k=-N/2}^{i-1}\int_{x_i-\alpha}^{x_i+\alpha}\frac{\gamma\,dx}{(x-x_k)^2+\gamma^2} \quad \text{(A5)}$$

$$=\int_{x_i+\alpha}^{x_i+(N+1)\alpha+2i\alpha}\frac{\gamma\,dy}{(y-x_i)^2+\gamma^2}$$

이 된다. 식 (A3)의 우변 첫 번째 항을 계산한 방식과 유사하게 식 (A3)의 우변 두 번째 항도 계산할 수 있다. 즉,

$$\sum_{k=i+1}^{N/2}\int_{x_i-\alpha}^{x_i+\alpha}\frac{\gamma\,dx}{(x-x_k)^2+\gamma^2} \quad \text{(A6)}$$

$$=\int_{x_i-(N+1)\alpha+2i\alpha}^{x_i-\alpha}\frac{\gamma\,dy}{(y-x_i)^2+\gamma^2}$$

따라서 고정된 α에 대하여 N이 매우 클 때, 식 (A5)와 (A6)을 사용하면 식 (A3)은 근사적으로

$$\sum_{k\neq i}^{N} Prob\left\{\left|x_k^*-x_i\right|\le\alpha\right\} \quad \text{(A7)}$$

$$\approx\frac{1}{\pi}\int_{-\infty}^{x_i-\alpha}\frac{\gamma\,dx}{(x-x_i)^2+\gamma^2}+\frac{1}{\pi}\int_{x_i+\alpha}^{\infty}\frac{\gamma\,dx}{(x-x_i)^2+\gamma^2}$$

$$=1-\frac{1}{\pi}\int_{x_i-\alpha}^{x_i+\alpha}\frac{\gamma\,dx}{(x-x_i)^2+\gamma^2}=1-p$$

가 된다. 여기서 코시 분포를 따르는 확률밀도함수의 성질인

$$\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{\gamma\,dx}{(x-x_k)^2+\gamma^2}=1 \quad \text{(A8)}$$

과 본문의 식 (5)인

$$p=Prob\left\{\left|x_i^*(j)-x_i(j)\right|\le\alpha_j\right\} \quad \text{(A9)}$$

$$=\frac{1}{\pi}\int_{x_i-\alpha_j}^{x_i+\alpha_j}\frac{\gamma_j\,dx}{(x-x_i(j))^2+\gamma_j^2}$$

을 이용하였다.