J. Soc. Korea Ind. Syst. Eng Vol. 40, No. 1 : 95-104, March 2017
https://doi.org/10.11627/jkise.2017.40.1.095
ISSN : 2005-0461(print)
ISSN : 2287-7975(online)

The Effect of Sample and Particle Sizes in Discrete Particle Swarm Optimization for Simulation-based Optimization Problems

Dong-Soon Yim†

Department of Industrial and Management Engineering, Hannam University

시뮬레이션 최적화 문제 해결을 위한 이산 입자 군집 최적화에서 샘플수와 개체수의 효과

임 동 순†

한남대학교 산업경영공학과

This paper deals with solution methods for discrete and multi-valued optimization problems. The objective function of the problem incorporates noise effects generated in case that fitness evaluation is accomplished by computer based experiments such as Monte Carlo simulation or discrete event simulation. Meta heuristics including Genetic Algorithm (GA) and Discrete Particle Swarm Optimization (DPSO) can be used to solve these simulation based multi-valued optimization problems. In applying these population based meta heuristics to simulation based optimization problem, samples size to estimate the expected fitness value of a solution and population (particle) size in a generation (step) should be carefully determined to obtain reliable solutions. Under realistic environment with restriction on available computation time, there exists trade-off between these values. In this paper, the effects of sample and population sizes are analyzed under well-known multi-modal and multi-dimensional test functions with randomly generated noise effects. From the experimental results, it is shown that the performance of DPSO is superior to that of GA. While appropriate determination of population sizes is more important than sample size in GA, appropriate determination of sample size is more important than particle size in DPSO. Especially in DPSO, the solution quality under increasing sample sizes with steps is inferior to constant or decreasing sample sizes with steps. Furthermore, the performance of DPSO is improved when OCBA (Optimal Computing Budget Allocation) is incorporated in selecting the best particle in each step. In applying OCBA in DPSO, smaller value of incremental sample size is preferred to obtain better solutions.

Keywords : Simulation-Based Optimization, Genetic Algorithm, Particle Swarm Optimization, Optimal Computing Budget Allocation

1. 서 론

다수의 이산적 값을 갖는 문제(multi-valued discrete problem)는 결정변수 값이 0부터 $M-1$까지 정수로 구성된 최적화 문제이다. 이러한 문제의 해결 방법에 대한 많은 연구들은 유전 알고리즘(Genetic Algorithm : GA)이나 입자 군집 최적화(Particle Swarm Optimization : PSO) 등의 집단 개체에 기초한 메타 휴리스틱을 대상으로 하고 있다[8, 11, 13, 14, 17]. 이들 알고리즘들은 이산적인 해를 갖는 문제에 용이하게 적용될 수 있고, 목적함수가 정형적으로 정의될 수 없는 실제적인 많은 문제들에 효과적

Received 9 February 2017; Finally Revised 17 March 2017; Accepted 20 March 2017
† Corresponding Author : dsyim@hnu.kr

으로 적용될 수 있는 장점을 가지고 있다.

목적함수로 정의된 해의 평가값이 확정적이 아니라 확률적인 경우에 평가값은 노이즈를 포함한다. 몬테카를로 시뮬레이션이나, 이산사건 시뮬레이션 등의 컴퓨터 실험에 기반한 최적화에서 노이즈는 시뮬레이션 모델에 포함된 랜덤 효과에 기인한다[7]. 노이즈를 포함하는 평가함수에 대한 최적화에서 중요한 것은 올바른 평가값의 추정이다. 올바르지 않은 평가값으로 최적화를 할 경우 잘못된 최적해 뿐만 아니라 해가 수렴하지 않는 등의 좋지 않은 결과를 초래한다. 올바른 평가값을 얻기 위해서는 충분한 횟수의 평가를 시행하여 정확한 기대 평가값을 추정하여야 한다. 그러나, 시뮬레이션 실험에 의한 한 번의 평가에 적지 않은 노력과 시간이 요구된다는 점을 고려하면 충분한 횟수의 평가는 실현 불가능할 수 있다.

시뮬레이션 최적화에 GA나 PSO 등의 집단 개체에 기반한 메타 휴리스틱을 적용하기 위해서는 해의 평가횟수(샘플수)뿐만 아니라 개체수와 세대수(또는 단계수)를 결정하여야 한다. 이들 수량은 허용되는 시뮬레이션 실행시간의 제약을 받으므로 수량간 상충효과가 발생한다. 정확한 추정을 위하여 샘플수를 증가시키면 단계수나 개체수를 감소시킬 수 밖에 없어 해의 질이 떨어지고, 샘플수를 감소시키면 올바른 평가값의 추정이 되지 않아 잘못된 최적해를 생성할 수 있다.

이러한 상충효과에 대한 집단 개체 기반 메타 휴리스틱의 접근 방법은 샘플 평균(explicit averaging)과 함축적 평균(implicit averaging)으로 설명될 수 있다[7]. 샘플 평균은 한 해에 대해 다수의 샘플 평가값을 구하여 이들의 평균으로 평가값을 추정하는 일반적인 방법이다. 샘플수가 n일 때 추정에 대한 분산은 $\sqrt{n}$에 반비례하여 n이 클수록 분산은 작아진다. 샘플수 n은 평가횟수와 추정 분산 간의 상충효과를 고려하여 사전에 구한 후 모든 해의 평가에 동일하게 적용시키는 방법이 일반적이다. 고정된 샘플수 보다는 상황에 따라 변동되는 샘플수를 활용하는 여러 방안들이 제안되었다. Aizawa와 Wah[1]는 GA 실행 중 샘플수를 조절하는 방안을 제안하였다. 세대수가 증가될수록 샘플수를 증가시키고, 또한 해에 대한 추정 분산이 크면 샘플수를 증가시켰다. Zhai[18]는 세대수의 증가에 따라 샘플수를 선형적으로 증가시키는 것이 좋은 결과를 가져왔다고 보고하였다. 한 해에 대한 평가값을 결정할 때 n개의 샘플을 구하기 보다는 이미 결정된 주변해의 평가값들을 활용하는 방안도 제안되었다. 즉, 주변해들에 대한 평균을 구하여 주어진 해의 평가값으로 대신하였다[3].

함축적 평균은 집단 개체에 기초한 최적화에서 개체수를 증가시키는 방법이다. 개체수가 클수록 각 개체의 평가값 추정에 대한 노이즈 영향을 적게 한다. 이러한 영향을 함축적 평균 효과라고 부른다[7]. GA에서 개체수가 많을수록 적자 선택 시 노이즈의 영향을 감소시킨다는 결과가 보고되었다[9].

GA에서 샘플수와 개체수 간에 존재하는 상충효과를 분석하는 연구도 수행되었다. Miller와 Goldberg[9]는 샘플수와 개체수를 최적화하기 위한 이론적 모델을 개발하였다. Fitzpatrick과 Grefensette[5]의 연구에서는 개체수를 증가시키는 것이 샘플수를 증가시키는 것 보다 더 좋은 결과를 가져온다고 보고하였다. 이와 같이 GA에서는 개체수와 샘플수 간의 관계를 분석한 연구들이 다수 존재하지만, PSO에서는 충분치 않은 것으로 파악된다.

본 논문에서는 다수의 이산적 값을 갖는 시뮬레이션 최적화 문제에 이용될 수 있는 DPSO(Discrete Particle Swarm Optimization)의 효과적 적용을 목적으로 한다. 특히, 시뮬레이션 최적화 관점에서 실수값의 해를 갖는 PSO의 적용에 대해서는 비교적 충분한 연구가 이루어진 반면, 이산적 값의 해를 갖는 DPSO의 적용에 대해서는 아직 많은 연구가 필요한 것으로 판단된다. 노이즈가 있는 평가값에 대해 개체수와 샘플수 간의 관계를 분석하여 DPSO에 의해 최적의 해를 효과적으로 생성할 수 있는 방법을 모색한다. 시뮬레이션 최적화 문제에서 DPSO 성능을 향상시킬 수 있는 방법 중 하나는 OCBA(Optimal Computing Budget Allocation)[8]이다. OCBA는 주어진 총 실행시간 하에서 샘플수를 조정하기 위한 목적에서 개발되었다. DPSO에 적용되는 OCBA의 효과를 분석하는 것도 본 논문의 부가적인 목적이다.

논문의 구성은 다음과 같다. 제 2장에서는 이산적 값의 결정변수로 구성된 최적화 문제의 해결을 위한 DPSO와 성능 비교를 위해 사용된 GA를 소개하고, 제 3장에서는 DPSO에 적용될 수 있는 OCBA 방법론을 기술한다. 제 4장에서는 다모드, 다차원을 갖는 표준 평가함수에 대해 실험한 결과를 논한다.

2. 이산 최적화 문제 해결을 위한 GA와 DPSO

2.1 유전 알고리즘

GA를 정수형 문제에 적용하기 위하여는 문제 해결에 적합한 염색체의 표현방법과 그 표현에 합당한 교배, 돌연변이 연산자를 정의하여야 한다. 정수형 변수를 표현하는 가장 일반적인 방법은 이진 표현이다. 이진 표현방법에서 염색체는 n개의 이진 벡터로 정수를 표현한다. 이진 벡터의 크기는 표현하려는 정수의 크기에 따라 결정된다.

본 연구에서의 GA는 한 세대에서의 초기 염색체들에 대하여 평가값에 기초한 선택 확률을 구하고 룰렛 휠 방

식에 의해 염색체들을 선택한다. 선택된 염색체들은 교배와 돌연변이 연산을 통하여 다음 세대의 염색체들을 생성한다. 이진 표현 방법에 적용될 수 있는 대표적인 교배 연산자는 절삭위치(cut point)에 의한 것이다. 교배확률 P_c에 의해 선택된 두 염색체에 대한 이 연산자는 염색체에서 임의의 다수 위치를 선택하여 이 위치를 기준으로 두 부모 염색체 수들을 서로 교환시킨다. 이진 표현에서의 대표적인 돌연변이 연산자는 각 이진값에 대해 돌연변이 확률 P_m에 따라 0은 1로, 1은 0으로 바꾸는 것이다.

2.2 이산 입자 군집 최적화

n-차원 공간에서 정의되는 변수 벡터 x의 목적함수 $f(x)$에 대한 최적화 문제를 위한 기본적인 PSO 알고리즘은 m개의 개체로 구성된 집단을 사용하고, k번째 단계에서의 i번째 개체는 연속적인 n-차원 탐색 공간에서의 해인 위치 벡터 x_i^k로 표현되어 평가값 $f(x_i^k)$를 가진다. i번째 개체가 k번의 단계를 통해 경험한 가장 좋은 해를 p_i라고 하고, 이웃한 모든 개체가 경험한 가장 좋은 해를 g라고 하자. $k+1$번째 단계에서 i번째 개체의 속도 벡터와 위치 벡터는 다음과 같이 표현된다.

$$v_i^{k+1} = w \cdot v_i^k + c_1 r_1 (p_i - x_i^k) + c_2 r_2 (g - x_i^k)$$
$$x_i^{k+1} = x_i^k + v_i^k$$

w, c_1, c_2는 각 항의 가중치이고, r_1, r_2는 0부터 1 사이의 난수이다. 위 식은 비교적 적은 수의 파라미터를 포함하여 최적화 문제에 대한 적용이 용이하다는 장점을 가진다.

DPSO는 정수형 문제를 풀기 위해 해를 실수가 아닌 이진 벡터로 표현한 것이다[8]. 즉, 위의 식에서 위치 벡터인 x_i^k는 정수값을 가지며 이를 이진 벡터로 표현된다. 속도는 이진 벡터와 동일한 크기를 가지는 실수 벡터로 구성한다. 속도의 수정 계산은 실수의 PSO에서와 같은 동일한 식을 사용하고, $w=1$로 한다. 위치 벡터의 수정을 위해서는 우선 실수의 속도를 시그모이드(Sigmoid) 함수를 이용해 0부터 1사이의 수로 변환한다.

$$S_{id} = sid(V_{id}) = \frac{1}{1+e^{-V_d}}$$

V_{id}는 i번째 개체의 d번째 속도 값을 의미한다.

다음에는 0부터 1사이의 난수 U를 생성하고 이를 시그모이드 함수값과 비교하여 0 또는 1의 값을 결정한다.

$$\text{if } (U < S_{id}) \text{ then } X_{id} = 1$$
$$\text{else } X_{id} = 0$$

따라서, X_{id}가 1일 확률이 S_{id}이고, 0일 확률은 1-S_{id}가 된다.

속도 v는 최대값을 $v_{\max}$로 제한한다. $v_{\max}$가 작을수록 돌연변이 율을 크게 하는 효과가 있다.

3. DPSO에서의 OCBA 적용

의사 결정을 목적으로 하는 시뮬레이션 프로젝트에서는 여러 대안 중 가장 좋은 결과를 가져오는 최선의 대안을 선택하는 것이 일반적이다. 최선의 대안을 올바르게 선택할 수 있는 확률을 높이기 위해서는 중요한 대한들에 많은 샘플수를 할당하는 것이 좋을 것이다. 즉, 우수하다고 판단되는 중요한 대안들의 추정치를 보다 정확하게 구하는 것이다. 반면 중요치 않는 대안들에는 추정치의 분산이 클지라도 최선의 대안을 선정하는 목적에서는 적은 샘플수를 할당해도 무방하다. 주어진 총 실행시간 하에서 시뮬레이션 실행 횟수를 조정하기 위한 이러한 목적에서 OCBA가 개발되었다. OCBA는 샘플수를 할당하는 문제를 비선형 최적화 문제로 만들고, 이 문제의 해를 구하는 방법으로 구성된다[4, 20].

k개의 대안이 있고 이중 가장 작은 값의 추정치를 가지는 대안을 선택한다고 하자. 총 실행 횟수 T가 주어졌을 때 각 대안의 샘플수 N_i를 구하는 문제는 다음 식과 같다. 목적함수는 올바른 선택(Correct Selection)을 할 확률인 $P\{CS\}$를 최대화하는 것이다.

$$\begin{aligned} \max \quad & P\{CS\} \\ s.t. \quad & N_1 + N_2 + \cdots + N_k = T \\ & N_i \in N, i = 1, \cdots, k \end{aligned}$$

이 문제를 풀기 위해서는 $P\{CS\}$를 구할 수 있어야 한다. Chen[4]은 시뮬레이션에 의한 성능척도 추정치들이 정규분포에 따른다는 가정과 Bonferroni 부등식에 의해 APCS(Approximate Probability of Correct Selection)를 구하였다. APCS를 최대화하는 비선형 문제는 라그란지안 완화에 의한 식으로 변형되고, 이를 풀기 위해 Karush-Khun-Tucker(KKT) 조건을 적용하였다. 이러한 과정을 거쳐 개발된 순차적 OCBA의 절차는 다음과 같다.

알고리즘 : 순차적 OCBA

단계 0. 각 대안들에 샘플수 n_0를 할당하여 성능척도에 대한 평균치(J_i)와 표준편차(σ_i)를 구한다.

$$l = 0; \; N_1^l = N_2^l = \cdots = N_k^l = n_0$$

단계 1. 만약 $\sum_{l=1}^{k} N_i^l \geq T$이면 정지한다.

단계 2. 총 샘플수를 Δ만큼 추가한 후 다음 식에 의해 각 대안의 샘플수 $N_1^{l+1}, N_2^{l+1}, \cdots, N_k^{l+1}$를 구한다.

$$\frac{N_i^{l+1}}{N_j^{l+1}} = \left(\frac{\sigma_i/\sigma_{b,i}}{\sigma_j/\sigma_{b,j}}\right)^2,\ i, j \in \{1, 2, \cdots, k\},$$

and $i \neq j \neq b$

$$N_b^{l+1} = \sigma_b \sqrt{\sum_{i=1,\, i\neq b}^{k} \frac{(N_i^{l+1})^2}{\sigma_i^2}}$$

$$\sigma_{b,i} = \bar{J}_b - \bar{J}_i,\ \ \bar{J}_b = \min_i \bar{J}_i$$

단계 3. 각 대안들에 $\max\{0, N_i^{l+1} - N_i^l\}$만큼 추가 샘플수를 할당하여 성능척도에 대한 평균과 표준편차를 구한다.

$$l = l + 1$$

단계 1로 간다.

이 알고리즘에서 초기에 각 대안에 할당되는 샘플수 n_0와 샘플수의 증가치 Δ를 결정하여야 한다. Chen[4]은 n_0는 5에서 20사이의 값을, Δ는 $k/10$보다 크고, $k/5$보다 작은 값을 사용하기를 권고하였다.

PSO에서는 각 단계에서 지금까지 가장 좋은 값을 갖는 개체의 평가값을 필요로 한다. 이를 위해 m개의 개체 중 가장 좋은 값을 갖는 개체를 선택하고, 이 개체를 이전 단계까지의 가장 좋은 개체와 비교한다. m개의 개체 중 가장 좋은 값의 개체를 선택하는 문제는 OCBA에서 고려하는 최선의 대안을 구하는 문제와 동일하다. 따라서, PSO에서의 가장 좋은 개체를 선택하는 과정에서 OCBA를 적용하는 것은 시뮬레이션 실행 횟수의 효과적 배분이라는 관점에서 적절한 방법이라고 할 수 있다.

OCBA를 PSO에 적용한 연구들은 다수 있다. 10차원의 2차 함수를 평가함수로 하고 노이즈를 추가한 실험[2]과 본 연구에서 사용한 표준 평가함수와 유사한 형태의 함수를 대상으로 한 실험[10]에서 PSO에 OCBA를 적용한 결과 성능 향상을 가져왔다고 보고하였다. Zhang et al.[19]은 잡샵 스케줄링 문제를 대상으로 랜덤 키 표현(random key representation)과 최소 위치값(smallest position value) 규칙에 의한 PSO에 OCBA를 적용하였다. Horng 등[6]은 웨이퍼 프로브 검사 시뮬레이션 모델의 최적화를 위한 정수 PSO에 OCBA를 적용하였다. 이들 연구에서는 PSO의 각 단계에서 생성되는 현재해에 OCBA를 적용하였다. 그러나, Rada-Viella[12]의 연구에서는 현재해 보다는 각 현재해가 지금까지 경험한 가장 좋은 해에 OCBA를 적용하였다.

DPSO에 OCBA를 적용하는 방법은 PSO에서의 방법과 동일하다. 그러나, 위의 연구에 부가적으로 총 샘플수와 단계수가 주어졌을 때 각 단계에 샘플수를 할당하는 방법이 필요하다. 가장 단순한 방법은 동일한 값의 샘플수를 모든 단계에 할당하는 것이지만, GA에서의 Zhai [18] 연구와 같이 단계의 증가에 따라 선형적으로 증가시킬 수 있고, 반대로 단계의 증가에 따라 선형적으로 감소시킬 수 있다. 각 단계에서의 샘플수가 결정되었다면 OCBA에서 필요로 하는 초기 샘플수와 증가 샘플수를 결정하여야 한다.

4. 실험 및 분석

4.1 표준평가함수 및 실험 파라미터

실험을 위해 기 개발된 표준 벤치마크 문제[15, 16]들을 고려하였다. 특히, 단순한 문제가 아닌 고도의 복잡한 문제들을 위해 변수의 수가 많고, 극점이 다수인 다모드, 다차원 문제들을 대상으로 하였다. 이러한 목적에 의하여 문헌에서 제공되는 표준 평가함수 중 다음과 같은 4개 평가함수를 선정하였다.

평가함수 1 : Ackley's function

$$f(x) = -20\exp\left[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right] - \exp\left[-\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right] + 20 + e$$

평가함수 2 : Griewank's function

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

평가함수 3 : Michaelwicz's function

$$f(x) = -\sum_{i=1}^{n}\sin(x_i)\cdot\left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{-20}$$

평가함수 4 : Schwefel's function

$$f(x) = -\sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right)$$

선정된 평가함수들은 근본적으로 결정변수가 연속적인 값을 갖는 최소화 문제들이다. 이 문제들을 이산적 값의 해를 갖는 문제로 변형하기 위해 결정변수 값의 소수점 이하 자릿수를 고정시키고, 최소와 최대값을 주어 제

한된 수의 동일한 간격을 갖는 값을 갖도록 하였다. 예를 들어 -1 부터 1사이의 실수에 대해 소수점 1자리수로 제한하면 -1.0, -0.9, -0.8, …, 0.8, 0.9, 1.0으로 구성된 21개의 수가 된다. 이 21개 수 들은 본 연구에서의 이산적 해인 0부터 20까지의 정수로 매핑시킨다. 즉, 이산적 해를 생성하는 최적화 알고리즘을 통해 0부터 20까지의 21개 정수인 해가 생성되면 이를 -1.0 부터 1.0까지의 실수로 변환하고, 이 실수를 입력으로 평가함수 값을 구한다.

<Table 1>은 4개 평가함수에 대해 각 결정변수의 최소값(min), 최대값(max)과 이산적인 수로 변환할 때 해의 크기(no of sols), 변수의 값을 이진벡터로 표현할 때 벡터의 크기(bits/variable), 문제를 구성하는 결정변수의 수(no of vars), 그리고, 한 해를 구성하는 총 이진벡터의 크기(total bits)를 나타낸다. 본 논문에서의 GA, DPSO는 이진벡터로 정수형 해를 표현한다. 변수의 수는 20 또는 30으로 하여 대체로 많은 차원의 문제를 고려하였다. Ackeley 함수의 경우 변수의 값이 -32.768부터 32.767까지의 범위를 갖는다. 이들 범위를 소수점 3자리로 국한하면 총 65,536개의 실수로 구성된다. 이 값들은 0부터 65,535인 정수로 코딩되어 16비트로 표현된다. 총 20개의 변수가 있으므로 요구되는 총 비트수는 320이다. 즉, 이진 표현으로 하나의 해를 표현하기 위해 320비트를 사용한다. 해 공간을 이루는 모든 가능한 해의 수는 Ackeley 함수의 경우 $65{,}536^{20} = 2.1\times10^{96}$으로 매우 많은 해를 포함한다.

<Table 1> Test Functions

Function	Ackeley	Girewank	Michaelwicz	Schwefel
min	-32.768	-600.0	0	-500.00
max	32.767	600.0	3.14159	500.00
no of sols	65536	12001	314160	100001
bits/variable	16	14	19	17
no of vars	20	20	20	20
total bits	320	280	380	340

<Table 1>의 4가지 함수 값을 올바른 평가값이라고 가정하고, 올바른 평가값에 노이즈를 더한 값을 실험에서 구해지는 평가값으로 간주하였다. <Table 2>에서와 같이 노이즈는 평균 0을 갖는 정규분포로 가정하고, 표준편차는 각 평가함수 값의 범위에 0.01을 곱한 값으로 하였다.

<Table 2> Standard Deviations in Test Functions

Functions	Ackeley	Girewank	Michaelwicz	Schwefel
Range of fitness values (R)	20	1800	20	10000
Std of noise (0.01×R)	0.2	18.0	0.2	100.0

DPSO와 GA의 실행에 사용된 파라미터는 <Table 3>과 같이 예비 실험을 통해 적절하다고 판단되는 값을 선정하였다.

<Table 3> Algorithm Parameters

Algorithm	Parameter
GA	$P_c = 0.25,\ P_m = 0.01$
DPSO	$c_1 = 2.0,\ c_2 = 2.0,\ V_{\max} = 6.0$

4.2 DPSO와 GA의 성능 비교

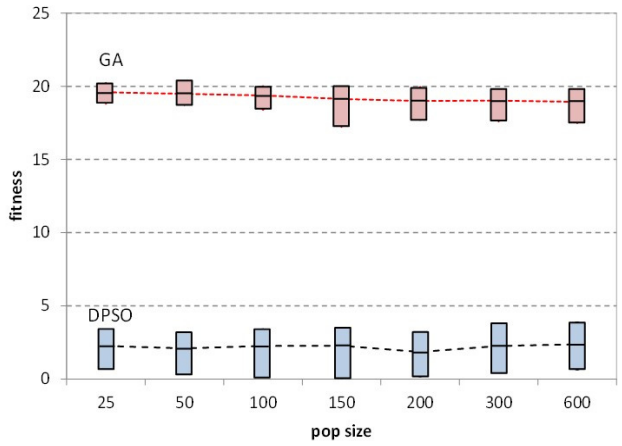
DPSO와 GA의 성능 비교를 위한 실험은 허용되는 총 샘플수를 150,000으로 가정하고, 단계수를 250으로 고정하여 한 단계에서의 허용되는 샘플수를 600으로 한정하였다. 이에 따라 생성된 <Table 4>의 7가지의 개체수와 한 해의 평가에 필요한 횟수인 샘플수의 조합으로 실험하였다. 두 알고리즘에서의 초기 집단은 랜덤으로 생성된 해로 구성하였다.

<Table 4> Population and Sample Size

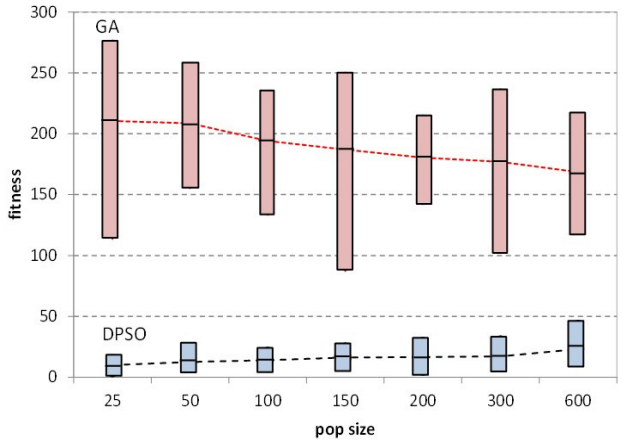
No	Population Size	Sample Size/Solution
1	25	24
2	50	12
3	100	6
4	150	4
5	200	3
6	300	2
7	600	1

<Figure 1>부터 <Figure 4>는 노이즈를 갖는 4개 평가함수의 최소화 문제에 대한 실험결과로 DPSO와 GA를 각각 25회 반복 실행하여 생성된 해의 올바른 평가값인 평가값 기대치에 대한 범위를 나타내었다. 알고리즘에서 해의 평가값은 노이즈를 포함한 값으로 계산되지만, 그림에서는 구해진 해를 표준함수의 올바른 평가값인 기대치로 환산하였다.

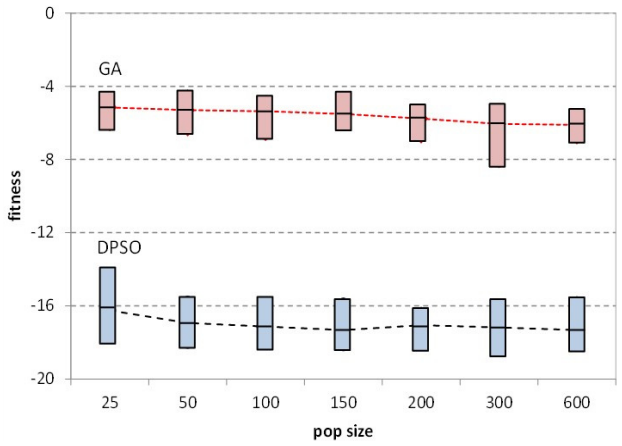
실험결과는 DPSO가 모든 평가 함수에서 GA에 비해 월등히 우수한 해를 생성하였음을 나타낸다. Ackeley 함수의 실험결과인 <Figure 1>에서 GA는 개체수 600에서 가장 성능이 우수하여 18.93의 평균 기대 평가값을 갖는 해를 생성했지만 DPSO는 개체수 200에서 가장 성능이 우수하여 1.85의 평균 기대 평가값을 갖는 해를 생성하였다. 그 외 평가함수에서도 유사한 결과를 가져와 DPSO의 성능이 GA에 비해 월등히 우수함을 나타낸다. 또한, 안정성 면에서도 DPSO가 GA에 비해 유사하거나, 또는



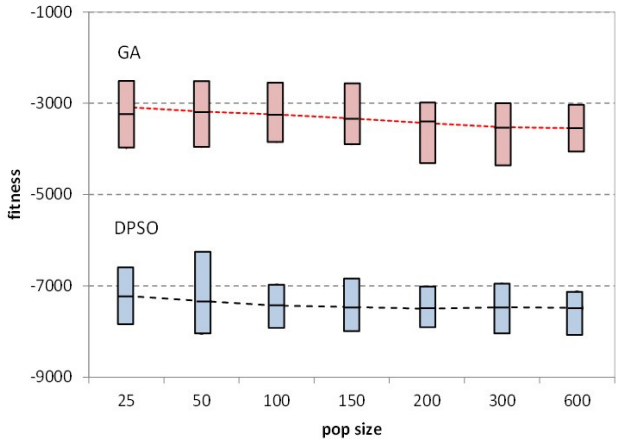
<Figure 1> Fitness Values with(Pop Size, Sample Size) Combinations in Ackeley Function



<Figure 2> Fitness Values with(Pop Size, Sample Size) Combinations in Girewank Function



<Figure 3> Fitness Values with(Pop Size, Sample Size) Combinations in Michaelwicz Function



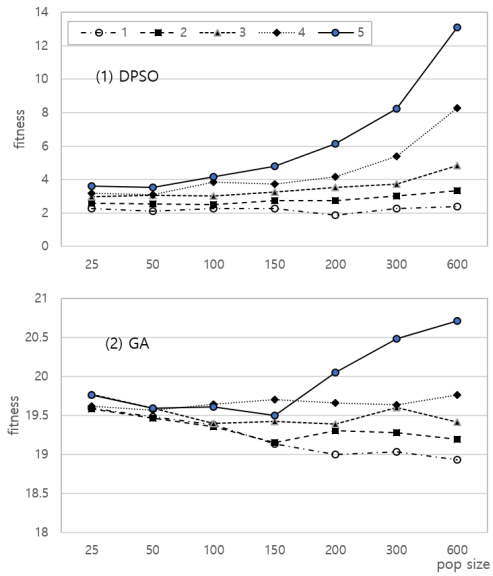
<Figure 4> Fitness Values with(Pop Size, Sample Size) Combinations in Schewefel Function

더 우수하다고 할 수 있다. <Figure 1>과 <Figure 3>에서 GA 해에 대한 범위와 DPSO에 대한 해의 범위에 차이가 적어 두 방법의 안정성을 비교하기 어렵지만, <Figure 2>와 <Figure 4>에서는 DPSO의 해가 더 적은 범위를 가진다. 특히, Girewank 함수에서는 GA와 DPSO에 대한 해 평가값 범위의 차이가 커 DPSO가 보다 안정적으로 해를 생성한다고 할 수 있다.

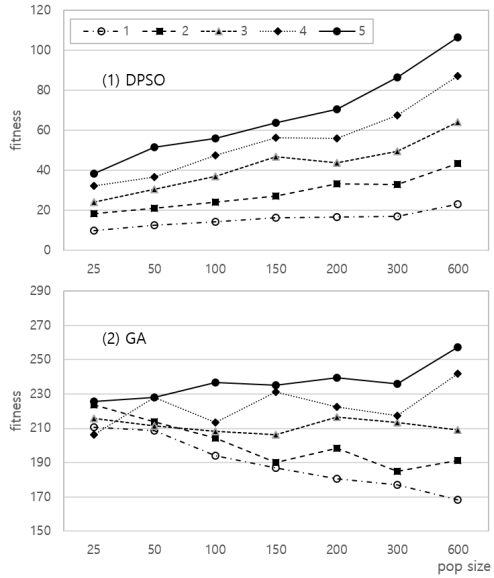
GA는 모든 평가함수에서 해의 평가값에 큰 차이는 아닐지라도 개체수가 증가할수록 더 우수한 해를 생성하는 양상을 보인다. 이 같은 결과는 샘플수의 증가보다는 개체수의 증가가 GA에서 더 좋은 결과를 가져온다는 기존 연구결과[5]와 일치한다.

DPSO에서는 평가함수 3, 4에서는 개체수가 증가할수록 더 우수한 해를 생성하는 양상을 보이지만 평가함수 2에서는 반대의 결과를 가져왔다. 그러나, 개체수의 변화에 따라 해 값에는 그다지 큰 차이를 보이지 않아 개체수와 샘플수 간의 관계를 알기 어렵다.

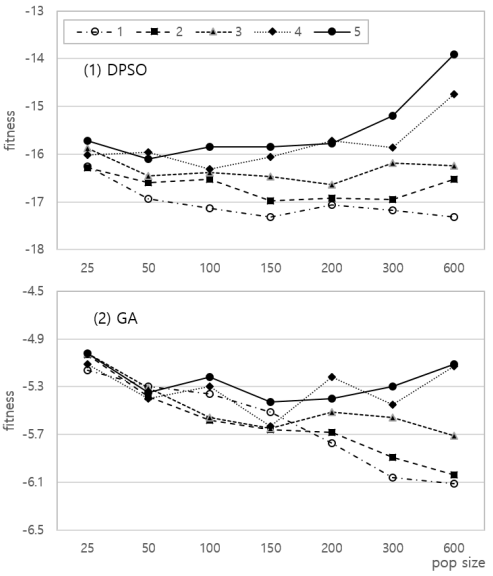
GA에서는 다음 세대를 구성할 개체를 선택하기 위해 적은 평가값을 갖는 우수한 개체가 선택된다. 이러한 선택은 평가값이 다소 정확하지 않더라도 다수의 우수한 개체를 선택하는데 큰 장애가 되지 않는다. 그러나, DPSO에서는 각 개체가 지금까지 경험한 가장 좋은 해와 전체 개체들이 경험한 가장 좋은 해의 두 가지 해에 의해 개체들의 다음 위치가 결정된다. 이때 가장 좋은 해들의 올바른 선택을 위해서 적정 이상의 샘플수로 평가값을 보다 정확하게 추정하는 것이 요구된다. 이 같은 현상은 <Figure 2>의 Girewank 함수에서 두드러져 개체수의 증가(즉, 샘플수의 감소)에 따라 성능이 나빠짐을 나타낸다. 또한 <Figure 3>, <Figure 4>에서와 같이 개체수가 25로 너무 적은 경우 해의 성능이 나빠지는 결과를 가져온다. 즉, 적정 개체수를 보장하여야 우수한 해를 생성함을 의미한다.



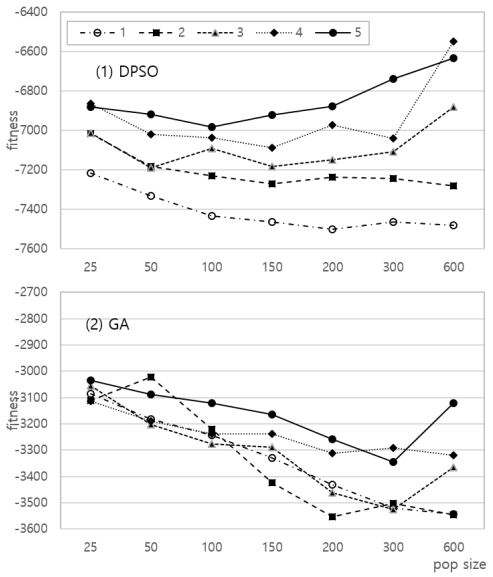
<Figure 5> Average Fitness Values with Increasing Pop Size in Ackeley Function



<Figure 6> Average Fitness Values with Increasing Pop Size in Girewank Function



<Figure 7> Average Fitness Values with Increasing Pop Size in Michaelwicz Function



<Figure 8> Average fitness values with increasing pop size in Schwefel function

개체수와 샘플수 간의 관계를 좀 더 파악하기 위한 추가 실험을 수행하였다. <Figure 5>부터 <Figure 8>까지는 노이즈의 크기인 표준편차를 <Table 2>의 주어진 값에 1, 2, 3, 4, 5배로 증가하여 각 평가함수에 대해 실험한 결과로 25회 실행한 해의 평균 기대 평가값을 나타낸다. 각 그림의 상단은 DPSO, 하단은 GA를 나타내어 두 방법 모두 노이즈의 크기가 클수록 해의 질이 떨어지는 자연스러운 결과를 나타난다. 특히, Girewank 함수에 대한 결과인 <Figure 6>이 이 같은 현상을 가장 잘 나타낸다. GA는 노이즈가 작을 경우 이전 분석과 같이 대체로 개체수가 클수록 모든 평가함수에서 좋은 해를 생성한다. 그러나, 노이즈가 클 때 <Figure 5>와 <Figure 6>에서 보듯이 개체수 600, 샘플수 1의 너무 적은 샘플에서는 해의 질이 떨어지는 양상을 보인다. DPSO는 Ackeley, Girewank 함수와 노이즈가 큰 Michaelwicz, Schwefel 함수에서 개체수가 클수록 해의 질이 떨어지는 현상이 발생한다. 특히, Ackeley, Girewank 함수에서는 이러한 결과가 확연히 나타나 DPSO에서 대체로 개체수보다는 샘플수의 확보가 더 중요하다는 것을 의미한다. Michaelwicz, Schwefel 함수에서도 노이즈가 클 때 개체수 보다는 샘플수가 중요하다는 현상을 나타낸다.

4.3 DPSO에서 단계별 평가횟수 할당

이전의 실험에서는 DPSO의 각 단계에 고정된 샘플수 600을 할당하였지만 단계의 증가에 따라 샘플수를 선형적으로 증가 또는 감소시킬 수 있다.

이러한 단계별 샘플수의 변화 방법이 어떠한 효과를 가져오는지를 분석하기 위한 실험을 수행하였다. 이전의 실험에서와 같이 총 샘플수를 150,000, 단계수를 250, 개체수를 100으로 하였다. 선형적 증가에서는 단계 1에서 샘플수 100(개체 당 1)을 할당하고, 단계 250에서는 1,200(개체 당 12)을 할당하여 단계의 증가에 따라 개체 당 샘플수를 선형적으로 비 감소되도록 하여 총 샘플수가 15,000에 이르도록 하였다. 선형적 감소에서는 개체 당 샘플수를 단계 1에서는 12, 단계 250에서는 1로 하여 단계의 증가에 따라 선형적으로 비 증가되도록 하였다.

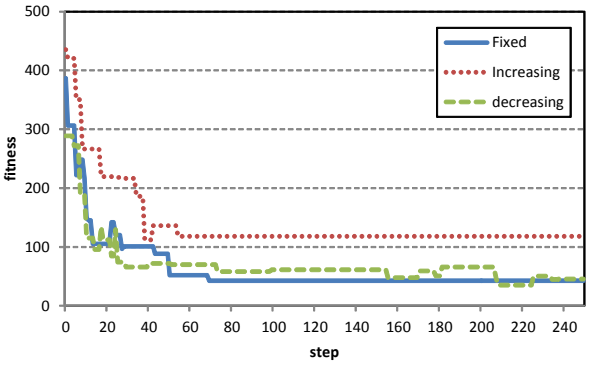
<Table 5>는 노이즈의 크기인 표준편차를 <Table 2>에 주어진 값의 5배로 하여 각 평가함수에 대해 실험한 결과로 25회 실행한 해의 기대 평가값에 대한 평균과 표준편차를 나타낸다. 모든 평가함수에서 선형적 증가 방법은 고정된 방법과 선형적 감소 방법에 비해 신뢰수준 90%에서 통계적으로 차이를 보여 성능이 떨어진다. 그러나, 고정된 방법과 선형적 감소는 통계적으로 차이를 보이지 않는다.

선형적 증가 방법은 초기 단계에 불충분한 샘플수가 할당된다. 이에 따라 올바르지 않은 평가값으로 인해 해의 질은 떨어지고, 잘못된 해로 수렴할 수 있다. 단계가 증가할수록 많은 샘플수로 정확한 평가값 추정치를 생성할지라도 초기의 잘못된 해가 지금까지의 가장 좋은 해로 인식되어 올바르지 않은 해를 생성할 수 있다. Girewank 평가함수에서 DPSO를 1회 실행시켜 구한 단계별 가장 좋은 해의 기대 평가값을 나타내는 <Figure 9>는 이 같은 사실을 반영한다. 반면, 선형적 감소 방법은 초기에 비교적 정확한 해를 생성하여 올바른 방향으로 해들이 이동하고, 빠르게 수렴한다. 빠른 수렴으로 인하여 후기 단계에서의 잘못된 평가값 추정은 해의 질에 그다지 큰 영향을 미치지 않는다. 고정된 수의 평가횟수 할당 방법은 선형적 감소 방법과 큰 차이를 보이지 않는다.

<Table 5> Comparison of Sample Size Allocation Methods

Test function	Fixed	Increasing	Decreasing
Ackeley	4.11/0.62	20.17/3.06	4.34/0.84
Girewank	50.10/12.80	158.12/68.69	44.09/17.68
Michaelwicz	-15.95/0.81	-14.62/1.32	-15.63/0.96
Schwefel	-7032.77/247.83	-6481.07/701.74	-7049.02/310.66

*average/standard deviation of expected fitness value.



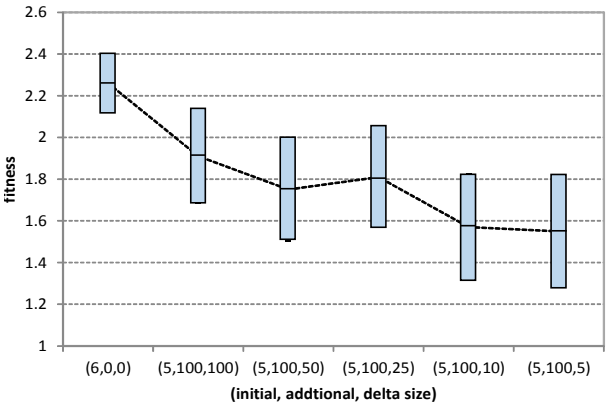
<Figure 9> Trends of Expected Fitness Values of Solutions in Girewank Function

4.4 DPSO에의 OCBA 적용

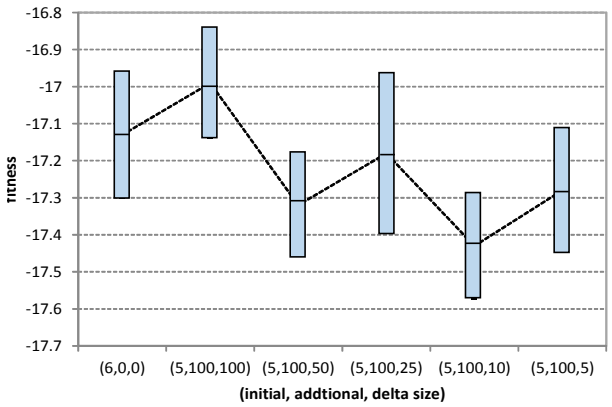
이전의 실험결과 DPSO에서는 대체로 개체수 보다 샘플수를 크게 하는 것이 선호되고, 각 단계에 평가횟수를 선형적으로 증가시키기보다 감소시키거나 고정시키는 방법이 좋다는 분석이 가능하였다. 이는 가장 좋은 해의 올바른 선택이 중요하고, 이를 위해 각 개체에 적정 이상의 샘플수를 할당하여 정확한 평가값을 추정하여야 한다는 것을 의미한다. 실행시간이 제한되어 있을 때 올바른 평가값 추정을 위한 방법으로 OCBA의 적용을 고려할 수 있다.

본 연구에서 OCBA는 각 단계에서의 현재해를 대상으로 한다. 즉, 현재해 중 가장 좋은 해를 선택하는 문제에 OCBA를 적용한다. OCBA를 적용하기 위해서는 초기 샘플수, 각 단계에서의 추가 샘플수, 증가 샘플수를 정의하여야 한다. OCBA의 효과를 분석하기 위한 실험에서는 이전에 수행된 개체수 100, 각 개체의 고정된 샘플수 6으로 수행한 실험을 기준으로 하였다. 이에 따라 DPSO의 각 단계에 주어지는 샘플수를 600으로 고정시켜 각 개체에 주어지는 초기 샘플수를 5로 하고, 추가 샘플수를 100으로 하였다. OCBA에서는 증가 샘플수의 차이가 어떤 효과가 있는지를 분석하기 위하여 5가지의 증가치 5, 10, 25, 50, 100으로 실험하였다. 예를 들어, 증가치 5로는 순차적 OCBA 알고리즘에서 단계 2와 단계 3을 20회 수행한다.

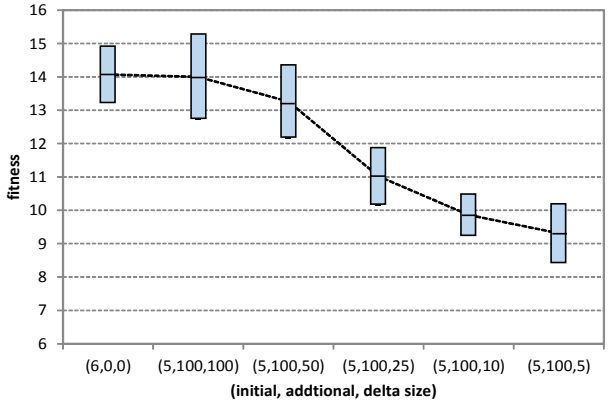
<Figure 10>부터 <Figure 13>은 초기 샘플수, 추가 샘플수, 증가 샘플수의 각 조합에서 25회 실행하여 구한 해의 기대 평가값 평균에 대한 90% 신뢰구간(Confidence Interval : CI)을 나타낸다. Ackeley, Girewank 함수에서는 각 개체에 고정된 샘플수를 할당한 (6, 0, 0)에 비해 OCBA를 적용한 효과를 확연히 나타낸다. 특히, 증가 샘플수를 작게 할수록 즉, 한번의 샘플수 할당 결정에서 주어진 총 샘플수가 작을수록 더 좋은 성능을 가져온다. Michaelwicz,



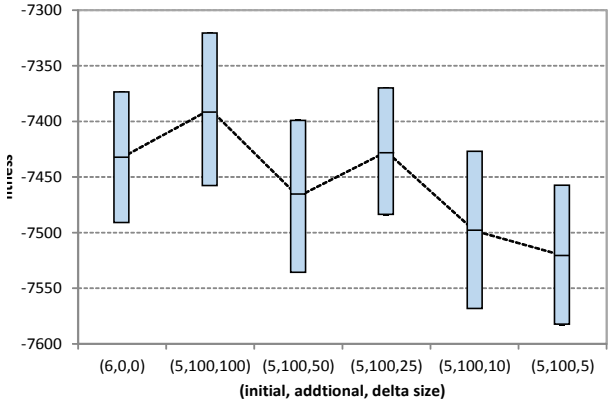
<Figure 10> 90% CI of Expected Fitness values of DPSO with OCBA in Ackeley Function



<Figure 12> 90% CI of Expected Fitness Values of DPSO with OCBA in Michaelwicz Function



<Figure 11> 90% CI of Expected Fitness Values of DPSO with OCBA in Girewank Function



<Figure 13> 90% CI of Expected Fitness Values of DPSO with OCBA in Schwefel Function

Schwefel 함수에서는 증가 샘플수의 감소에 따른 효과를 확연히 알기 어렵지만, 적어도 OCBA가 고정된 샘플수보다 더 효과적일 수 있다는 것을 알 수 있다. 예를 들어, Michaelwicz 함수에서는 한 개체의 평가값 추정에 6회의 고정된 샘플수를 할당하는 것 보다 (5, 100, 10)의 초기 샘플수, 추가 샘플수, 증가 샘플수에 의한 OCBA가 더 좋은 결과를 생성한다. 또한, Schwefel 함수에서는 한 개체의 고정된 샘플수를 할당하는 것 보다 (5, 100, 5)의 초기 샘플수, 추가 샘플수, 증가 샘플수에 의한 OCBA가 더 좋은 결과를 생성한다고 볼 수 있다.

4. 결 론

본 논문에서는 해가 이산적인 값을 갖고, 평가값에는 노이즈가 포함되는 최적화 문제를 다루었다. 최적해를 구하기 위한 DPSO의 성능을 분석하기 위해 유전 알고리즘과 비교하는 실험을 수행하였다. 실험은 4개의 표준 평가함수를 대상으로 이들 함수값을 기대 평가값으로 간주하고, 가상적인 노이즈를 발생시키도록 하였다. 총 평가횟수는 제한되어 고정된 값을 갖는다고 가정하고, 개체수와 각 개체에 할당되는 샘플수를 변경시켜 실험하였다. 실험결과 해의 질이나, 안정성 면에서 DPSO가 GA에 비해 월등히 우수한 성능을 보였다. 또한, GA에서는 샘플수 보다는 개체수를 증가시키는 것이 선호되는 반면, DPSO에서는 개체수 보다 샘플수의 확보가 더 중요하다는 분석을 가능케 하였다. 더욱이 DPSO에서 단계의 증가에 따라 샘플수를 증가시키는 방법은 좋지 않은 해로 수렴하여 해의 질이 떨어진다. 이보다는 각 단계에 고정된 샘플수를 할당하거나, 선형적으로 감소하는 샘플수를 할당하는 방법이 더 좋은 해를 생성한다. DPSO에서는 OCBA를 적용하여 우수한 해에 더 많은 샘플수를 할당하는 것이 좋은 효과를 가져온다. 이 같은 결과는 OCBA를 적용한 DPSO 방법이 이산적 값을 갖는 시뮬레이션 최적화 문제에서 좋은 결과를 가져올 수 있음을 시사한다. 본 논문에서는 표준 평가함수에 가상적인 노이즈를 추가한 문제를 대상으

로 하였으나, 향후 몬테카를로 또는 이산사건 시뮬레이션의 현실적인 문제를 대상으로 DPSO와 OCBA의 효과를 분석하는 연구가 필요할 것으로 예상된다.

References

[1] Aizawa, A.N. and Wah, B.W., Dynamic Control of Genetic Algorithms in a Noisy Environment, *Proceeding of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 103-115.

[2] Bartz-Beielstein, T., Blum, D., and Branke, J., Particle Swarm Optimization and Sequential Sampling in Noisy Environments, *Proceedings of the 6th Metaheuristics International Conference (MIC2005)*, 2005, pp. 89-94.

[3] Branke, J., Schmidt, C., and Schmeck, H., Efficient Fitness Estimation in Noisy Environment, in Genetic and Evolutionary Computation, Spector, L. et al., Eds, Morgan Kaufman, 2001, pp. 243-250.

[4] Chen, C.-H., Lin, J., Yucesan, E., and Chick, S.E., Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization, *Discrete Event Dynamic Systems : Theory and Applications*, Vol. 10, No. 3, 2000, pp. 251-270.

[5] Fitzpatrick, J.M. and Grefenstette, J.I., Genetic Algorithm in Noisy Environments, *Machine Learning*, Vol. 3, No. 2, 1988, pp. 101-120.

[6] Horng, S.-C., Yang, F.-Y., and Lin, S.-S., Applying PSO and OCBA to minimize the Overkills and Re-Probes in Wafer Probe Testing, *IEEE Transactions on Semiconductor Manufacturing*, Vol. 25, No. 3, 2012, pp. 531-540.

[7] Jin, Y. and Branke, J., Evolutionary Optimization in Uncertain Environments-A Survey, *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 3, 2005, pp. 303-317.

[8] Kennedy, J. and Eberhart, R.C., A Discrete Binary Version of the Particle Swarm Algorithm, *IEEE International Conference on Systems, Man, and Cybernetics*, 1997, pp. 4104-4108.

[9] Miller, B.L. and Goldberg, D.E., Genetic Algorithms, Selection Schemes and the Varying Effects of Noise, *Evolutionary Computation*, Vol. 4, No. 2, 1996, pp. 113-131.

[10] Pan, H., Wang, L., and Liu, B., Particle Swarm Optimization for Function Optimization in Noisy Environment, *Applied mathematics and Computation*, Vol. 181, No. 2, 2006, pp. 908-919.

[11] Pugh, J. and Martinoli, A., Discrete Multi-Valued Particle Swarm Optimization, *Proceedings of IEEE Swarm Intelligence Symposium*, 2006, pp. 103-110.

[12] Rada-Juan, R.-V., Zhang, M., and Joknston, M., Optimal Computing Budget Allocation in Particle Swarm Optimization, *GECCO '13*, 2013, pp. 81-88.

[13] Song, H., Diolata, R., and Joo, Y., Photovoltaic System Allocation Using Discrete Particle Swarm Optimization with Multi-level Quantization, *Journal of Electrical Engineering and Technology*, Vol. 4, No. 2, 2009, pp. 185-193.

[14] Veeramachaneni, K., Osadciw, L., and Kamath, G., Probabilistically Driven Particle Swarms for optimization of Multi Valued Discrete Problems : Design and Analysis, *Proceedings of IEEE Swarm Intelligence Symposium*, 2007, pp. 141-149.

[15] Vesterstrom, J. and Thomsen, R., A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems, *Evolutionary Computation CEC 2004 Congress*, Vol. 2, 2004, pp. 1980-1987.

[16] Yang, X.-S., Test Problems in Optimization, in Engineering Optimization : An Introduction with Metaheuristic Applications (Eds Xin-She Yang), John Wiley & Sons, 2010.

[17] Yim, D.S., Particle Swarm Optimization to Solve Multi-Valued Discrete Problems, *Journal of the Society of Korea Industrial and Systems Engineering*, 2013, Vol. 36, No. 3, pp. 63-70.

[18] Zhai, W., Kelly, P., and Gong, W.-B., Genetic Algorithms with Noisy Fitness, *Mathematical and Computer Modeling*, Vol. 23, No. 11/12, 1996, pp. 131-142.

[19] Zhang, R., Song, S., and Wu, C., A Two stage hybrid particle swarm optimization algorithm for the stochastic job shop scheduling problem, *Knowledge-Based Systems*, Vol. 27, 2012, pp. 393-406.

[20] Zhang, S., Chen, P., Lee, H.L., Peng, C.E., and Chen, C.-H., Simulation Optimization Using the Particle Swarm Optimization with Optimal Computing Budget Allocation, *Proceedings of the 2011 Winter Simulation Conference*, edited by Jain, S., Creasey, R.R., Himmelspach, J., White, K.P., Fu, M., 2011, pp. 4303-4313.

ORCID

Dong-Soon Yim | http://orcid.org/0000-0003-4968-9091