Society of Korea Industrial and Systems Engineering Vol. 36, No.3 : 63-70, September 2013 ISSN : 2005-0461(print)
http://dx.doi.org/10.11627/jkise.2013.36.3.63 ISSN : 2287-7975(online)

Particle Swarm Optimizations to Solve Multi-Valued Discrete Problems

Dong-Soon Yim†

Department of Industrial and Management Engineering, Hannam University

다수의 값을 갖는 이산적 문제에 적용되는 Particle Swarm Optimization

임 동 순†

한남대학교 산업경영공학과

Many real world optimization problems are discrete and multi-valued. Meta heuristics including Genetic Algorithm and Particle Swarm Optimization have been effectively used to solve these multi-valued optimization problems. However, extensive comparative study on the performance of these algorithms is still required. In this study, performance of these algorithms is evaluated with multi-modal and multi-dimensional test functions. From the experimental results, it is shown that Discrete Particle Swarm Optimization (DPSO) provides better and more reliable solutions among the considered algorithms. Also, additional experiments shows that solution quality of DPSO is not lowered significantly when bit size representing a solution increases. It means that bit representation of multi-valued discrete numbers provides reliable solutions instead of becoming barrier to performance of DPSO.

Keywords : Multi-valued Discrete Problem, Genetic Algorithm, Particle Swarm Optimization

1. 서 론

실제적으로 발생하는 많은 최적화 문제들은 결정변수가 다수의 이산적 값을 갖는 문제(multi-valued discrete problem)에 속한다. 이들은 해공간이 이산적일 뿐 아니라 변수간, 또는 변수 수준 간에 정성적 구분이 있는 경우를 포함한다. 예를 들어 어떤 변수는 수치적인 값보다는 A/B가 있는 경우, A만 있는 경우, B만 있는 경우, 그리고, A/B 모두 없는 경우의 4가지를 나타낼 수 있다. 또는 어떤 변수는 연속적인 수치값을 갖지만, 문제의 단순함을 위해 값의 범위를 50개 수준으로 나누어 각각을 0부터 49까지의 인덱스로 표현할 수 있다. 특히, 최적화 대상이 되는 목적함수의 평가값을 몬테카를로 또는 이산사건 시뮬레이션에 의해 구하는 경우 적지 않은 실행 시간으로 인하여 해공간의 크기를 제한하는 것이 필요하다[11]. 이를 위해 연속적인 결정변수값을 다수의 수준으로 나누어 이산적인 변수로 간주한다. 이러한 문제들은 다수의 이산적 값을 갖는 문제로 이를 해결하기 위한 방법에 대한 연구들이 활발해지고 있다[6, 9].

다수의 이산적 값을 갖는 문제의 해결을 위한 알고리즘에 대한 연구들은 대부분 유전 알고리즘이나 Particle Swam Optimization(PSO) 등의 집단 개체에 기초한 메타 휴리스틱을 대상으로 하고 있다. 이는 이들 알고리즘들이 이산적 값의 해를 갖는 문제에 용이하게 적용될 수 있고, 목적함수가 정형적으로 정의될 수 없는 실제적인 많은

Received 8 August 2013; Accepted 6 September 2013
† Corresponding Author : dsyim@hnu.kr

문제들에 효과적으로 적용될 수 있는 장점을 가지고 있기 때문이다.

해를 이진 비트로 표현하는 유전 알고리즘에서는 이산적인 값의 해를 비교적 용이하게 표현할 수 있다. 그러나, 실수값으로 해를 표현하는 PSO에서는 이산적인 값의 해를 표현하기 위한 별도의 방법이 요구된다. Discrete PSO (DPSO)는 해를 이진 비트로 표현하는 방법으로 개발되었다[3]. 이진 비트가 아닌 일반적인 M 진 비트와 유사한 방법으로 해를 표현하는 방법도 제안되었다. Multi-Valued PSO(MVPSO)는 다수의 정수형 값을 갖는 해를 l 개의 확률벡터로 표현하였다[6]. M 진 비트로 표현하는 다른 방법들은 실수에 의한 계산 결과를 다 수준 수량화 (multi-level quantization) 절차를 통해 M 진수로 변환하거나[7], 확률적 요소를 부가한 후 M 진수로 변환하는 방법[9]을 이용하였다.

이들 다수의 이산적 값을 갖는 문제에 적용될 수 있는 메타 휴리스틱들은 실험을 통해 알고리즘의 효과성을 입증하는 절차를 거치었지만 심도있는 실험과 분석이 이루어지고 있지 않은 실정이다. 해가 실수값을 갖는 최적화 문제에서의 성능 분석[8]은 대체로 잘 이루어졌으나, 다수의 이산적 값을 해로 갖는 문제에서의 대부분 실험은 결정변수의 수가 2 또는 3인 작은 문제에 국한되고, 목적함수로 이용된 평가함수 역시 일부 단순한 수식에 그치고 있는 실정이다. 이에 따라 다양한 메타 휴리스틱 방법 간의 심도있는 비교 분석이 필요하다.

본 논문은 다수의 이산적 값을 갖는 문제의 해결을 위해 개발된 유전 알고리즘, PSO, DPSO, MV PSO 등 메타 휴리스틱 방법들의 성능을 비교 분석하는 목적을 가진다. 특히, 다모드(multi modal), 다차원(multi-dimensional)을 갖는 고도의 복잡한 문제에 대해 각 알고리즘들의 성능을 분석하는 실험 결과를 논한다.

논문의 구성은 다음과 같다. 제 2장에서는 다수의 이산적 값을 갖는 문제에 이용되는 유전 알고리즘과 PSO들을 소개한다. 제 3장에서는 소개된 메타 휴리스틱들의 성능을 분석하기 위해 선정된 표준 평가함수를 소개한다. 제 4장에서는 실험 분석 결과를 설명하고, 마지막으로 향후 연구 방향을 제 5장에서 서술한다.

2. 이산적 값을 갖는 문제에 적용되는 유전 알고리즘과 PSO

본 논문에서 다루는 최적화 문제에서는 결정변수 x_i의 값이 0부터 M_i-1까지의 정수값을 갖는다. 이러한 이산적 값을 갖는 결정변수로 구성된 최적화 문제의 해결을 위한 유전 알고리즘과 PSO를 설명한다.

2.1 유전 알고리즘

유전 알고리즘(Genetic Algorithm : GA)은 여러 개의 개체가 동시에 병렬적으로 주어진 환경에 따라 적자생존의 방법으로 진화하여, 궁극적으로 최적의 상태에 도달하는 생태계의 진화이론에서 도입되었다[1]. 이 알고리즘은 여러 개의 개체로 구성된 군집이 진화할 때 구 세대가 얻은 환경에 대한 정보는 염색체에 저장되어 다음 세대로 전달된다. 이 때 조상의 염색체가 그대로 복제되어 자손에게 전달되는 것이 아니라 조상의 염색체에 교배(crossover), 돌연변이(mutation), 전위(inversion) 등의 연산을 가하여 얻은 염색체로 전달된다. 구세대 중에서 한 개체가 선택되어 자손에게 유전정보를 남길 확률은 일반적으로 그 개체가 주위환경, 그리고 나머지 개체와 어떻게 상호 작용하는가에 의존하는 적응값(fitness)에 따라 변한다. 일반적으로 개체값이 좋을수록 자손을 남길 확률이 높아지는 적자생존(survival of fitness)의 법칙이 적용된다. 이 알고리즘은 여러 가지 종류의 최적화 문제에 응용되어 좋은 결과를 낳고 있다. 참고 문헌[5] 은 GA에 대한 기본적인 소개와 응용분야를 자세히 설명하고 있다.

GA를 정수형 문제에 적용하기 위하여는 이 문제의 해결에 적합한 염색체의 표현방법(chromosome representation)과 그 표현에 알맞는 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. 예를 들어, 변수 x_i의 값이 0부터 100까지의 101가지 정수를 가진다면 이 수를 표현하기 위해 7 bit가 필요하다.

$$64 = 2^6 < 10 \leq 12^7 = 128$$

이진 벡터를 정수로 변환하는 방법은 이진수를 정수로 변환하는 방법과 동일하다. 그러나, 고려되는 변수 x_i의 크기가 제한되어 0부터 M_i-1까지의 정수만을 허용하므로 약간의 변환이 필요하다. 위 예에서 이진 벡터 $(b_6\, b_5 \cdots b_0)$는 다음과 같은 절차에 따라 정수로 변환된다.

1) 이진 벡터를 10진수로 변환한다.

$$((b_6 b_5 \cdots b_0))_2 = (\sum_{i=0}^{6} b_i 2^i)_{10} = x'$$

2) 10진수를 0부터 100까지의 정수로 변환한다.

$$x = x' \cdot \frac{100}{2^7 - 1}$$

본 연구에서의 GA는 한 세대에서의 초기 염색체들에 대하여 평가값에 기초한 선택 확률을 구하고 룰렛 휠(roulette wheel) 방식에 의해 염색체들을 선택한다. 선택된 염색체들은 교배와 돌연변이 연산을 통하여 다음 세대의 염색체들을 생성한다.

이진 표현 방법에 적용될 수 있는 대표적인 교배 연산자는 cut point이다. 교배확률 P_c에 의해 선택된 두 염색체에 대한 이 연산자는 염색체에서 임의의 다수 위치를 선택하여 이 위치를 기준으로 두 부모 염색체 수들을 서로 교환시킨다. 예를 들어, 두 염색체가 다음과 같이 표현되었다고 하자. 하나의 컷 포인트를 이용하여, 임의의 컷 포인트를 |로 표현하였다.

P1 = (00000000|0011111111000000001)
P2 = (11100101|0101110000101010111)

두 부모의 컷 포인트를 중심으로 좌우 수들은 서로 교환되어 두 자손에 상속된다.

O1 = (00000000|0101110000101010111)
O2 = (11100101|0011111111000000001)

이진 표현에서의 대표적인 돌연변이 연산자는 각 이진값에 대해 돌연변이 확률 P_m에 따라 0은 1로, 1은 0으로 바꾸는 것이다. 예를 들어, 위 염색체 O1의 유전자 중 4번째 유전자가 바뀐다면 다음과 같은 염색체가 된다.

O1 = (00001000010111000010101011 1)

2.2 PSO

PSO는 먹을 것을 찾아 다니는 한 무리 새떼의 행동을 본따 최적화 문제를 해결할 수 있도록 고안되었다[2]. Particle이라고 불리는 각 개체는 지금까지 자신이 경험한 가장 좋은 해 그리고, 이웃한 개체들이 경험한 것 중 가장 좋은 해의 두 가지 정보를 이용하여 현재 위치에서 자신의 이동방향을 결정하고, 이에 따라 다음 단계에서의 위치가 결정된다. 이 같은 위치는 최적화 문제에 대한 해로 1 : 1 매핑되어 개체의 이동은 해 공간에서의 탐색과정을 의미한다.

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

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

위 식에서 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. 더욱이 실수값의 범위를 가지는 연속적인 n-차원 탐색 공간에서는 위치벡터가 유효한 해를 직접적으로 표현하기 때문에 PSO의 적용이 용이하다. 그러나, 이산적인 해공간을 갖는 문제에서는 PSO의 적용이 그다지 용이하지 않다. 이산적 값을 갖는 최적화 문제를 해결하기 위해 지금까지 제안된 방법들은 크게 3가지로 나뉠 수 있다.

1) Truncation 방법(PSO)
2) DPSO
3) MVPSO

2.2.1 Truncation

정수형 문제에 PSO를 적용시키는 쉬운 방법은 우선 실수에 의한 속도와 위치벡터 계산을 원래의 PSO 방법에 의해 수행한 후 위치 벡터인 실수해가 0부터 M_i-1까지의 범위를 가지도록 하는 것이다. 즉, 범위를 벗어난 해는 가장 가까운 범위내의 수로 변환한다. 실수해로부터 정수해를 구하는 방법은 단순하다. 실수를 정수로 truncation 시키거나, 가장 가까운 정수로 반올림시킨다[4].

2.2.2 DPSO

DPSO는 정수형 문제를 풀기 위해 이전에 설명한 유전알고리즘에서와 같이 해를 실수가 아닌 이진 벡터로 표현한 것이다[3]. 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 Sigmoid 함수를 이용해 0부터 1사이의 수로 변환한다.

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

V_{id}는 i번째 d개체의 번째 속도 값을 의미한다. 그 다음에는 0부터 1사이의 난수 U를 생성하고 이를 sigmoid 함수값과 비교하여 0 또는 1의 값을 결정한다.

If ($U < S_{id}$) then $X_{id} = 1$
Else $X_{id} = 0$

즉, 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 율(mutation rate)를 크게 하는 효과가 있다. 위치벡터 값을 원래 문제의 정수값으로 변환하는 방법은 이진 벡터 표현의 GA에서와 동일하다. 이진 벡터를 10진수로 변환하고, 다시 10진수 값을 0부터 $M_i - 1$까지의 정수로 변환한다.

2.2.3 MVPSO

MVPSO에서의 위치 벡터는 0부터 l까지의 정수 범위를 가지는 정수 벡터에 부가적으로 각 정수값에 대한 확률 벡터를 포함한다[6]. 만약 DPSO와 같이 l이 2인 이진 벡터를 사용한다면 각 성분의 0, 1에 해당하는 2개 성분의 확률이 부가적으로 포함된다. 따라서, 각 개체의 해는 3차원의 행렬 x_{ijk}로 i번째 j개체의 성분의 k번째 값에 대한 확률로 간주된다. 해가 확률인 실수이므로 표준 PSO에서의 속도벡터 수정식을 그대로 사용할 수 있다.

수정된 속도에 의한 계산된 위치 벡터는 직접적으로 확률 그 자체를 의미하지는 않는다. 위치 벡터 값들은 별도의 계산을 거쳐 확률로 변환된다.

$$x_{ij}^{'} = \sum_{k=0}^{l} sig(x_{ijk})$$

$$P(s_j = k) = \frac{sig(x_{ijk})}{x_{ij}^{'}}$$

위치 벡터는 정확히 확률을 의미하지는 않지만 가급적 확률의 의미를 가지도록 하는 것이 바람직하다. 이를 위해 수정이 끝난 위치벡터의 값들을 다음과 같이 수정한다.

$$x_{ijk}^{'} = x_{ijk} - c_{ij}$$

c_{ij}의 결정은 다음 식을 만족하는 값으로 한다.

$$\sum_{k=0}^{l} sig(x_{ijk}^{'}) = 1$$

그러나, 위 식을 이용하여 c_{ij}를 수식적으로 결정할 수는 없다. 대신에 수치적 근사 방법을 이용하여 구할 수 있다.

위치벡터 값을 원래 문제의 정수값으로 변환하는 방법은 DPSO에서의 방법과 유사하다. l진법의 위치 벡터 값을 10진수로 바꾸고, 다시 이 값을 0부터 $M_i - 1$까지의 정수로 변환한다.

3. 성능분석을 위한 평가함수

본 연구에서 다루는 메타 휴리스틱 방법들의 성능 분석을 위해 기 개발된 표준 벤치마크(standard literature benchmarks) 문제[10]들을 고려하였다. 특히, 단순한 문제가 아닌 고도의 복잡한 문제들에 대해 유전 알고리즘, PSO, DPSO, MVPSO의 성능을 비교하는 목적으로 변수의 수가 많고, 극점(extreme point)이 다수인 다모드, 다차원 문제들을 대상으로 하였다. 이러한 목적에 의해 문헌에서 제공되는 표준 평가함수 중 7개 평가함수를 선정하였다. 선정된 평가함수들은 근본적으로 결정변수가 연속적인 값을 갖는 최소화 문제들이다. 이 문제들을 이산적 값의 해를 갖는 문제로 변형하기 위해 결정변수 값의 소수점 이하 자리수를 고정시키고, 최소와 최대값을 주어 제한된 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 수들은 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. 선정된 7개 평가함수들은 다음과 같다.

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 : Extended Easom's function

$$f(x) = -(-1)^n\left(\prod_{i=1}^{n}\cos^2(x_i)\right)\exp\left[-\sum_{i=1}^{n}(x_i - \pi)^2\right]$$

평가함수 3 : 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 4 : 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 5 : Rastrigin's function

$$f(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$$

평가함수 6 : Rosenbrock's function

$$f(x)=\sum_{i=1}^{n-1}\left[(x_i-1)^2+100(x_{i+1}-x_i^2)^2\right]$$

평가함수 7 : Schwefel's function

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

4. 실험 및 분석

<Table 1>은 7개 평가함수에 대해 각 결정변수의 최소값(min), 최대값(max)과 이산적인 수로 변환할 때 해의 크기(no of solutions), 문제를 구성하는 결정변수의 수(no of Variables)를 나타낸다. <Table 2>는 각 평가함수에 대해 한 변수의 값을 이진벡터로 표현할 때 벡터의 크기(bits/variable), 모든 변수를 포함하는 해를 구성하는 총 이진벡터의 크기(total bits), 그리고, 모든 해의 크기(solution space size)를 나타낸다. 각 함수에 대한 변수의 수는 20 또는 30으로 하여 대체로 많은 차원의 문제를 고려하였다. 함수 1의 경우 한 변수의 값이 -32.768부터 32.767까지의 범위를 갖는다. 이들 범위를 소수점 3자리로 국한하여 총 65536개의 실수로 구성된다. 이 값들은 0부터 65535인 정수로 코딩되어 16비트로 표현된다. 총 20개의 변수가 있으므로 요구되는 총 비트수는 320이다. 즉, 이진표현으로 하나의 해를 표현하기 위해 320비트를 사용한다.

<Table 1> Test Functions

Function	Min	Max	No. of Solutions	No Variables
1	-32.768	32.767	65536	20
2	-6.283	6.282	12566	20
3	-600.0	600.0	12001	20
4	0	3.14159	314160	20
5	-5.12	5.11	1024	30
6	-5.00	5.00	1001	30
7	-500.00	500.00	100001	20

<Table 2> Required Bits in Bit Representation

Function	Bits/variable	Total Bits
1	16	320
2	14	280
3	14	280
4	19	380
5	10	300
6	10	300
7	17	340

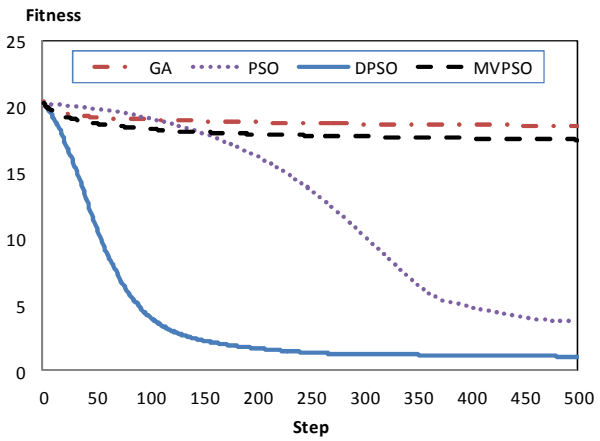
해 공간을 이루는 모든 가능한 해의 수는 함수 1의 경우 $65536^{20}=2.1\times10^{96}$으로 매우 많은 해를 포함한다. 기존의 성능 분석 연구에서는 2 또는 3개의 변수를 대상으로 실험한 것에 비교하여 본 연구에서는 보다 많은 변수를 고려한 복잡한 문제의 경우를 대상으로 하였다.

GA, PSO, DPSO, MVPSO의 4가지 최적화 방법들에 대한 실험은 개체수를 100으로 하고, 단계(세대) 수는 모든 방법들이 동일한 실행시간을 갖도록 조정하였다. 이를 위해 DPSO의 반복수를 500을 했을 때의 실행시간을 기준으로 하였다. MVPSO의 경우 이진 표현과 같이 0, 1의 두가지 값을 갖는 방법에 국한하였다. 기타 파라미터는 <Table 3>과 같다.

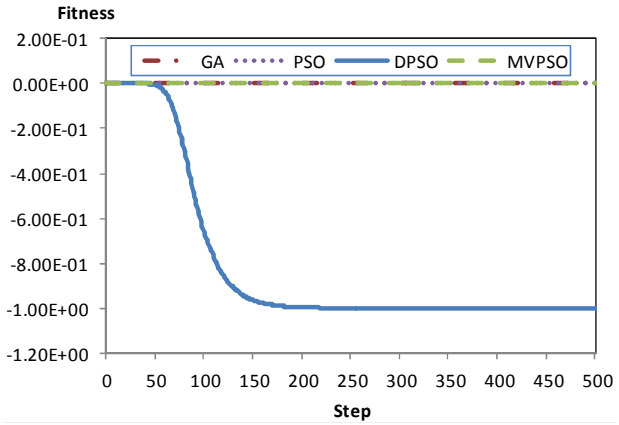
<Table 3> Algorithm Parameters

Algorithm	Parameter
GA	P_c=0.25 P_m=0.01
PSO	c_1=2.0, c_2=2.0, V_{max}=6.0 w_{min}=0.4, w_{max}=0.9, $w_{\in c}$=0.975
DPSO	c_1=2.0, c_2=2.0, V_{max}=6.0
MVPSO	c_1=2.0, c_2=2.0, V_{max}=6.0 w_{min}=0.4, w_{max}=0.9, $w_{\in c}$=0.975

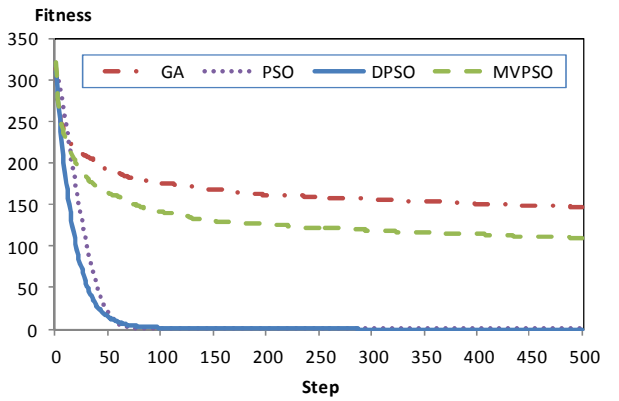
<Table 4>는 각 평가함수에서의 GA, PSO, DPSO, MVPSO에 대해 100번 반복 실행하여 구한 가장 좋은 해의 평균값과 표준편차를 나타낸다. <Figure 1>에서 <Figure 7>까지의 그림들은 단계별 가장 좋은 해의 평균값 추이를 나타낸다. 가로축은 모든 방법이 동일한 시간을 가지게 하여 DPSO를 기준으로 GA는 약 4배, PSO는 약 15배의, MVPSO는 0.3배의 단계로 나타내었다. 즉, DPSO는 500단계, GA는 약 2000 세대, PSO는 약 75,000 단계, MVPSO는 약 170 단계를 실행하였다.



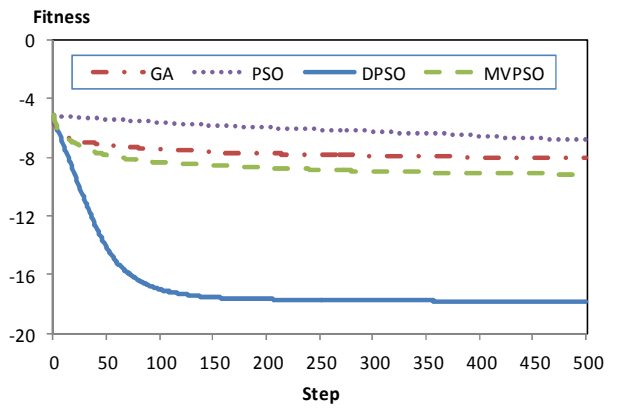
<Figure 1> Trend of Best Solutions for Test Function 1



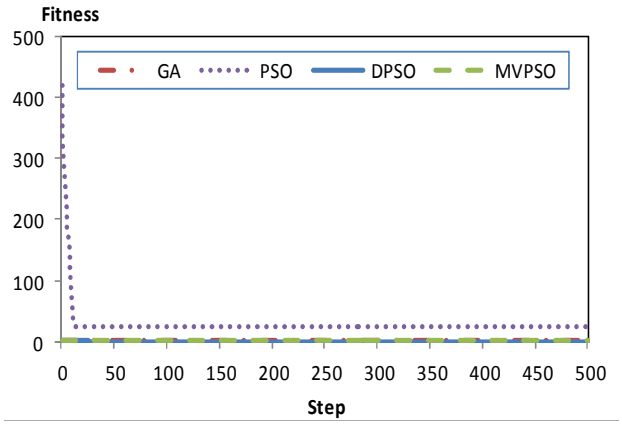
<Figure 2> Trend of Best Solutions for Test Function 2



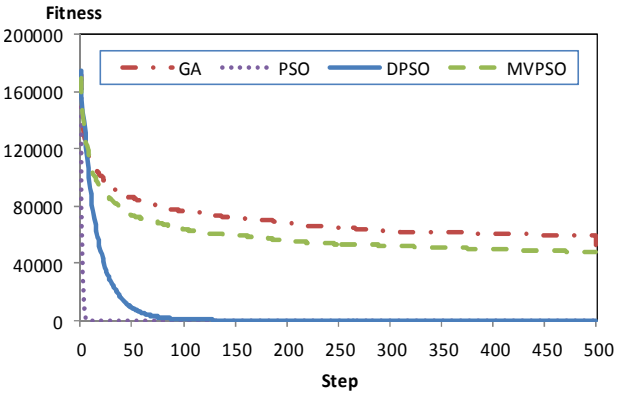
<Figure 3> Trend of Best Solutions for Test Function 3



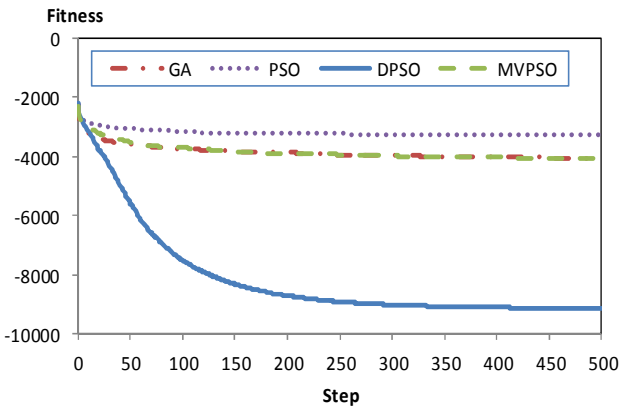
<Figure 4> Trend of Best Solutions for Test Function 4



<Figure 5> Trend of Best Solutions for Test Function 5



<Figure 6> Trend of Best Solutions for Test Function 6



<Figure 7> Trend of Best Solutions for Test Function 7

<Table 4>와 <Figure 1>에서 <Figure 7>까지의 그림에 의하면 DPSO가 알고리즘 초기에 빠르게 근사 최적해로 수렴하여 GA, PSO, MVPSO에 비해 우수한 해를 생성하였음을 알 수 있다. 특히, <Figure 2>의 평가함수 2에서는 DPSO만이 최적해인 평가값 -1에 수렴한 반면 나머지 방법들은 평가값 0에 수렴하였다. PSO의 경우 일부 문제(함수 5, 6)에서 갑자기 해의 평가값이 하락하는 현상을 보였다. <Figure 5>의 평가함수 5에서는 초기 단계에서 하락하여 평가값 6.63에 수렴하였고, 나머지 방법들의 평가값은 0에 수렴하여 PSO의 성능이 가장 떨어졌다. <Figure 6>의 평가함수 6에서는 PSO가 급작스런 평가값 하락을 보여 DPSO보다 우수한 해를 생성하였지만 다른

<Table 4> Experiment Results

Test Function		GA	PSO	DPSO	MVPSO
1	Ave	18.57	3.79	1.07	17.53
	Std	0.41	0.73	1.04	1.39
2	Ave	0.00	0.00	-1.00	0.00
	Std	0.00	0.00	0.00	0.00
3	Ave	146.99	0.26	0.03	114.93
	Std	18.65	0.22	0.06	42.79
4	Ave	-8.08	-6.83	-17.84	-9.14
	Std	0.48	0.48	0.48	1.35
5	Ave	1.11	23.05	0.00	0.85
	Std	0.13	6.76	0.00	0.25
6	Ave	51227	10.56	131	4583
	Std	8476	2.55	114	18333
7	Ave	-4060	-3236	-9150	-4082
	Std	407	205	270	407

평가함수에서는 DPSO보다 좋은 해를 생성하지 못하였다. GA와 MVPSO는 모든 함수에서 DPSO에 비해 해의 질이 떨어졌다.

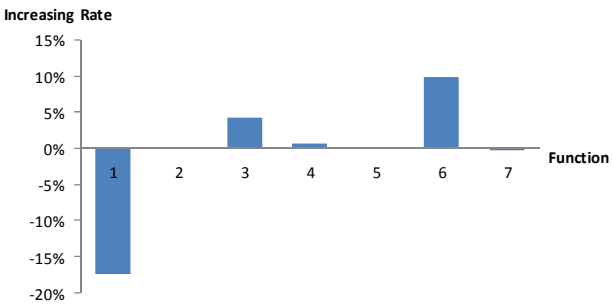
각 방법들에 대한 가장 좋은 해의 표준편차는 MVPSO가 가장 큰 값을 나타내어 일관성 있는 해를 생성하지 못함을 의미한다. 함수 1, 7을 제외하면 DPSO가 GA에 비해 적은 표준편차를 보인다. PSO도 비슷한 결과를 가져왔지만, 함수 5에서는 GA에 비해 51배의 큰 표준편차를 보여 문제에 따라 불안정한 해를 생성함을 의미한다.

위의 분석에 따라 DPSO가 우수한 해를 비교적 안정적으로 생성하여 비교 평가된 방법 중 가장 신뢰할 만한 방법임을 알 수 있다.

GA와 DPSO에서는 이진 비트로 정수형 해를 표현한다. 이진 표현 방법은 고려되는 범위내의 정수형 해를 1 : 1로 매핑하지 못하는 경우가 발생할 수 있다. 별도의 변환 과정을 거쳐 이진 표현의 정수값을 정해진 범위내의 정수형 해로 바꾼다. 정수형 해의 수 보다 이진 표현 정수값의 수가 같거나 많도록 이진 표현을 사용하므로 서로 다른 이진 표현 해가 동일한 정수형 해로 변환될 수 있다. 이러한 변환은 GA, DPSO에서 한 해와 이웃해 간의 향상 정도를 구별할 수 없어 알고리즘의 성능을 저해하는 요소로 작용한다고 알려져 있다[6, 7]. 이전의 실험에서 가장 좋은 성능을 보인 DPSO에 대해 이진 표현의 변환 효과를 분석하기 위한 부가적인 실험을 수행하였다. 7개 평가함수에 대해 한 변수의 표현 비트 수를 1 증가 시켜 그 영향을 분석하였다. <Table 5>는 각 평가함수에 대해 변수의 표현 비트를 1 증가시킨 결과 이진 표현의 정수 가지수에 대한 실제 해의 수 비율을 나타낸다. 예를 들어, 1번 평가함수의 경우 0부터 65,535까지의 65,536개 값을 표현하기 위해 16비트를 사용하여 실제해와 이진 표현 해가 100% 1 : 1 매칭되었으나, 17비트로 해를 표현할 경우 50%의 매칭율을 나타낸다. 즉, 실제해의 수가 이진 표현해의 수보다 50% 적어 두개의 서로 다른 이진 표현해가 동일한 해 값을 나타내고 있다. <Figure 8>은 <Table 5>의 비트 크기 증가에 따른 가장 좋은 해의 평가값 증가율을 나타낸다. 평가함수 1의 경우 평가값은 17비트로 해를 표현했을 때 16비트로 해를 표현한 경우에 비해 17%의 감소율을 나타내어 더 좋은 결과를 가져왔음을 의미한다. 그러나, 평가함수 3, 4와 6에서는 각각 4%, 1%, 10%의 평가값 증가율을 보여 비트 크기의 증가가 해의 질을 약간 악화시켰음을 나타낸다. 그 외 평가함수 2, 5, 7에서는 비트 크기의 증가가 해의 질에 영향을 미치지 못하였다. 결론적으로 모든 문제에서 비트 크기의 증가가 해의 질에 나쁜 영향을 크게 미친다고 보기 어렵다. 문제 1에서는 해의 질이 크게 향상되기도 하였고, 다른 문제에서도 해의 평가값이 동일하거나, 약간 악화되는 경우가 있었기 때문이다.

<Table 5> Bit Sizes in Bit Representation

Function	Minimum Bit Size		Increased Bit Size	
	Bit size	Matching Rate (%)	Bit size	Matching Rate (%)
1	16	100.0	17	50.0
2	14	76.7	15	38.3
3	14	73.2	15	36.6
4	19	60.0	20	30.0
5	10	100.0	11	50.0
6	10	97.8	11	48.9
7	17	76.3	18	38.1



<Figure 8> Increasing Rate in Best Fitness Values

4. 결 론

다수의 정수형 값을 갖는 최적화 문제를 해결하기 위해

유전 알고리즘, Particle Swarm Optimization 등의 개체 기반 메타 휴리스틱이 효과적으로 이용된다. 본 연구에서는 실험을 통해 GA, PSO, DPSO, MVPSO의 4가지 방법들에 대한 성능 분석을 수행하였다. 이를 위해 7개 표준 평가함수를 선정하였고, 휴리스틱 방법에서의 이산적 정수형 해를 실수값으로 변환하는 절차를 포함하였다. 실험 결과 DPSO가 다른 방법에 비해 우수한 해를 빠르고 안정적으로 생성하여 가장 신뢰성 있는 방법임을 입증하였다. DPSO는 이진 비트로 정수형 해를 표현한다. 이진 표현 방법은 고려되는 범위내의 정수형 해를 1 : 1로 매핑하지 못하여 알고리즘의 성능을 저해하는 요소로 작용한다고 알려져 있다. DPSO에 대해 이진 표현의 변환 효과를 분석하기 위한 실험을 수행하였다. 이를 위해 한 변수를 표현하는 비트 수를 증가시켜 동일한 해의 중복이 많도록 하였다. 실험 결과 비트의 증가가 해의 질에 나쁜 영향을 크게 미친다고 볼 수 없었다. 어떤 문제에서는 해의 질이 크게 향상되기도 하였고, 다른 문제에서도 해의 평가값이 동일하거나, 약간 악화되는 경우가 있어 일관성 있는 결과가 도출되지 못하였다. 이에 따라 이진 비트 표현에 의한 해를 실제 해로 변환하는 과정에서 발생하는 알고리즘의 성능 저하 효과가 그다지 크지 않은 것으로 분석되었다.

본 논문에서는 해의 평가값이 확정적인 경우를 대상으로 하였다. 그러나, 몬테카를로 또는 이산 사건 시뮬레이션을 이용한 최적화에서는 평가값이 랜덤 효과를 포함한다. 이 때 하나의 해에 대한 평가값을 추정하기 위해 다수의 시뮬레이션 실행이 요구된다. 이러한 문제에 DPSO를 효과적으로 이용하기 위해서는 적정의 개체수와 샘플수(하나의 평가값을 구하기 위한 실행 수)를 결정하여야 한다. 향후 이에 대한 연구가 지속되어야 할 것으로 예상된다.

Acknowledgement

This paper has been supported by 2012 Hannam University Research Fund.

References

[1] Goldberg, D.E. and Lingle, R., Alleles, Loci, and the TSP, Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985, p 154-159.

[2] Kennedy, J., Eberhart, R.C., Particle Swarm Optimization, Proceeding of the 1995 IEEE International Conference on Neural Networks, 1995, p 1942-1948.

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

[4] Laskari, E.C., Parsopoulos, K.E., and Vrahatis, M.N., Particle Swarm Optimization for Integer Programming, Proceedings of the IEEE 2002 Congress on Evolutionary Computations, 2002, p 1582-1587.

[5] Michaelwicz, Z., Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1992.

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

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

[8] Vesterstrom, J. and Thomsen, R., A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems, Proceedings of the IEEE 2004 congress on Evolutionary Computation, 2004, p 1980-1987.

[9] 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, p 141-149.

[10] Yang, Xie-She, Test Problems in Optimization in Engineering Optimization : An Introduction with Metaheuristic Applications (Eds Xin-She Yang), John Wiley and Sons, 2010.

[11] Yim, D.S, Park, C.H., Cho, N.C, and Oh, H.S., A Case Study on the Scheduling for a Tube Manufacturing System. *Journal of the Society of Korea Industrial and Systems Engineering*, Vol. 32, No. 3, 2009, p 110-117.