병렬 처리 구조를 이용한 차분 진화 알고리즘

임동현 ○ 이종현 안창욱*
성균관대학교 정보통신공학부 진화알고리즘연구실(SEAL)
{logicboom,ljh08375}@skku.edu. *cwan@evolution.re.kr (교신)

Differential Evolution Algorithm using Parallel Processing Structure

Dong-Hyun Lim○, Jong-Hyun Lee, Chang Wook Ahn*
Sungkyunkwan Evolutionary Algorithms Lab. (SEAL)
School of Information & Communication Engineering, SungKyunKwan University

요 약

본 논문은 차분 진화 알고리즘의 최적해 탐색 능력을 향상시키기 위해 병렬 처리 기법을 적용한 기법을 제안한다. 이를 위해서 기존의 개체군들을 5개의 그룹으로 나누어서 독립적으로 최적화 과정을 하도록 하여 일정한 확률에 의해서 각 그룹이 다른 그룹의 Best individual들을 변이 과정에서 참조하도록 하였다. 이러한 방식을 통해서 기존 차분 진화 알고리즘이 가지고 있는 지역해 수렴 문제를 해결하는 할 수 있도록 하였다. 실험을 통해서 제안된 차분 진화 알고리즘(P-DE)의 탐색 능력을 비교 및 분석 하였다. 실험 결과 제안된 차분 진화 알고리즘(P-DE)이 지역해 수렴 문제를 충분히 해결함으로써 기존의 알고리즘에 비해서 우수한 성능을 보이는 것을 확인 하였다.

1. 서 론

차분 진화 알고리즘은 Price와 Storm 에 의해 각 개체간의 벡터 차이를 이용하는 알고리즘으로써 현재 널리 사용되는 휴리스틱 탐색법인 유전 알고리즘과 유사한 알고리즘으로써 초기 개체군을 생성하고 이들을 교배, 변이 선택 과정을 거쳐 적합도가 개선되는 새로운 개체들을 추출해 낸다는 공통점이 있다. 하지만 기존의 유전 알고리즘은 표현형을 유전형으로 바꾸는 코딩과정이 필요한 반면 차분진화 알고리즘은 개체를 벡터로 표현하기 때문에 이러한 과정이 필요 없으며 새로운 개체를 생성할 때 산술적 연산을 통해서 생성한다는 차이점이 있다[1],[2],[3].

따라서 차분진화 알고리즘은 유전 알고리즘 보다 적은 수의 제어인자가 필요하며 비교적 적은 계산 시간을 소요한다는 장점을 가지고 있다. 하지만 차분 진화 알고리즘은 탐색이 수행 되는 동안 지역해에 수렴하게 되는 발생하는 단점이 있다. 지역해로 수렴하게 될 경우 벗어나지 못하고 머무는 확률이 높다. 이러한 문제점은 차분 진화 알고리즘의 성능을 감소하는 원인이 된다.

본 논문에서는 이러한 문제점을 병렬 처리 기법을 통해서 해결 하고자 한다. 병렬 처리는 서로 독립적으로 프로세서를 진행 하면서 결과를 얻는 방식이다 병렬 처리는 많은 분야들의 알고리즘에 적용하여 기존 알고리즘들의 성능을 향상시키는 결과를 가져 왔다[3],[4],[6],[8]. 본 논문은 이러한 병렬 처리를 통해서 기존의 알고리즘 보다 다수의 best individual들을 생성함으로써 지역해에 수렴하게 되는 상황을 감소 시키는 기법을 제안하고자 한다.

본 논문의 2장에서 차분 진화 알고리즘에 대해 소개와 제안된 알고리즘을 설명하고, 3장에서 본 논문에서 사용된 최적화 함수와 실험에 대해 고찰 한 후, 4장에서 결론을 맺는다.

2. 본 론

2-1. 차분 진화 알고리즘

차분 진화 알고리즘은 진화 알고리즘과 기본적인 연산은 같다. 초기 개체군을 무작위로 생성하여 변이, 교배, 선택을 통하여 적합도가 개선되는 개체들을 추출한다는 공통점을 가지고 있다. 하지만 차분 진화 알고리즘은 개체들을 벡터로 표현하기 때문에 진화 알고리즘이 가지 있는 코딩과정이 불필요하며 이들의 산술적인 연산을 통해 새로운 개체를 생성한다는 차이점이 있다[1],[5], [12],[13].

차분 진화 알고리즘에서의 초기 개체군의 각 개체들은 다음과 같이 구성된다.

$$[x_{i,0} = (x_{1,i,0}, \ldots\ldots, x_{D,i,0})], i = 1, 2, \ldots, \mathrm{NP} \quad (1)$$

이때 NP 는, 최적화 과정에서 사용되는 개체군의 수를 나타내며, 이 값은 탐색과정에서 언제나 일정하게 고정되어 있어져 있다. D 는 벡터의 차원으로 목적함수에서 사용되는 변수의 개수를 의미한다. 이러한 개체들이 모여서 다음 과 같이 개체군이 생성된다[1],[4],[5].

$$[x(low) \le x_{j,i,0} \le x(high)] \; for \; j = 1, 2, \ldots\ldots, D \quad (2)$$

초기 개체군을 생성한 이후 다음 3가지의 기본적인 연산들을 반복적으로 수행함으로써 최적화를 수행한다.

1) 변이 연산자 (Mutation operator)

차분 진화 알고리즘은 두 개체간 벡터의 차이를 임의의 개체에 더함으로써 새로운 개체의 벡터를 생성한다. 이 벡터를 새로운 시행 벡터라고 한다. 이러한 시행 벡터를 생성하는데 있어서 다양한 방식이 있지만 본 논문에서는 다음과 같은 시행 벡터를 사용한다.

$$v_{i,G+1} = x_{best,G} + f(x_{r1,G} - x_{r2,G}) \qquad (3)$$

여기에서 $x_{r1,G}$, $x_{r2,G}$는 G번째 세대에서 임의의 두 개체의 벡터이고, $x_{best,G}$ 는 G 번째 세대에서 목적함수에서 가장 적합한 개체의 벡터이다. f 는 가중치로서 벡터의 차이$(x_{r1,G} - x_{r2,G})$ 의 비중을 결정하는 제어상수로 0과 1 사의 값으로 목적함수에 따라서 다르게 정의한다. 차분 진화 알고리즘은 시행 벡터를 생성하는 규칙에 따라서 서로 다른 성능을 보여주게 된다[5],[10],[11],[14].

2) 교배 연산자 (Crossover operator)

변이 과정에서 생성된 시행 벡터를 현재의 벡터와 교배되어서 새로운 시행 벡터가 생성된다. 이 때 CR은 교배확률을 결정하는 상수로 0과 1 사이에 위치하며 1에 가까울수록 한쪽의 벡터만을 가지고 있을 확률이 높아지게 된다. 때문에 본 논문에서는 균등하게 교배하기 위해서 0.5의 확률로 설정하였다[5], [10],[11],[14].

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1} & for\ rand\ () < CR \\ v_{i,j,g} & for\ others \end{cases} \qquad (4)$$

3) 선택 연산자 (Selection operator)

변이와 교배를 통해서 생성된 시행 벡터를 현재의 벡터와 비교하여 목적 함수에 더욱 적합한 개체의 벡터를 사용한다.

$$x_{i,j,g+1} = \begin{cases} u_{i,g+1} & if\ \ f(u_{i,g+1}) < f(x_{i,g}) \\ x_{i,g} & otherwise \end{cases} \qquad (5)$$

만약 생성된 개체가 현재의 개체보다 적합도가 떨어지게 되면 기존 개체를 유지하게 된다. 반대로 생성된 개체의 적합도가 더 우수하게 되면 현재의 개체를 생성된 개체로 대체한다[5],[10],[11],[14].

앞서 언급한 3가지의 연산들을 반복적으로 수행함으로써 보다 우수한 개체들을 생성하는 방식으로 차분 진화 알고리즘은 수행된다. 이때 변이 연산자는 차분 진화 알고리즘의 성능을 결정하는 매우 중요한 연산자로서의 시행 벡터의 생성 방식에 따라서 차분 진화 알고리즘의 성능의 달라지게 된다. 시행 벡터에 사용되는 임의 개체가 모두 임의 개체들로만 사용될 경우 탐색 공간을 좀 더 넓게 탐색이 가능하여 최적해 발견 가능성이 높아지게 되지만, 높은 탐색 수행 시간을 요구하게 된다. 반면, 가장 적합한 개체를 사용하여 시행 벡터를 만들게 되면 탐색 수행 시간은 줄어들 수 있지만 탐색 공간을 Best individual에 주로 의존해서 탐색하기 때문에 최적해 발견 확률이 감소하게 된다. 본 논문에서는 이러한 차분 진화 알고리즘의 탐색 수행 시간 단축과 최적화 성능 향상을 동시에 만족 할 수 있는 알고리즘을 제안한다.

2-2. 제안 알고리즘(P-DE)

차분 진화 알고리즘은 개체간의 벡터 차이를 이용하는 알고리즘으로써 알고리즘이 수행 되면서 개체들이 지역해에 수렴하게 되는 위험성이 존재 하게 된다. 이러한 문제를 해결하기 위한 방법으로 병렬 처리 방식을 통해서 독립적인 탐색 공간을 생성하여 각 공간마다 개체들이 일정 확률로 개체들이 속해 있는 공간을 제외한 다른 탐색 공간에서의 best value들을 참조하여 현재의 탐색의 방향에서 다른 방향으로 변화를 주면서 지역해로 수렴하게 되는 상황을 방지 해주는 역할을 하게 된다. 이러한 방식을 사용함으로써 과도하게 한 지점으로 개체들이 모이는 현상을 완화 시킴과 동시에 공간 탐색을 넓게 할 수 있다는 장점이 생긴다. 제안된 차분 진화 알고리즘은 다음과 같은 순서로 최적화를 수행하게 된다.

1. 초기 개체군 설정
2. N개의 그룹으로 구분
3. 일정 확률에 의하여 변이 연산자에서 사용하는 참조 개체를 다른 그룹에서의 개체를 이용할지를 결정
4. 각 그룹마다 독립적인 연산을 수행. (변이 교배 평가)
5. 반복적으로 앞의 3번과 4번 연산들을 정해진 진화 세대 까지 반복하여 수행

3. 실험 및 결과

3-1. 실 험

본 논문에서는 기존의 기본적인 차분 진화 알고리즘과 제안 알고리즘(P-DE)을 비교하여 그 정확성을 비교하였다. 두 알고리즘은 계산 시간(즉, 전체 적합도 함수 평가 회수) 측면에서 동일한 조건을 설정하고, 각 알고리즘의 최적화 정도를 비교 하였다. 실험에 사용된 변수들은 다음과 같이 설정하였다.

개체 수: 20 ~ 200
벡터의 차원 수: 30
최대 목적함수 계산 횟수: 100000
제안 알고리즘의 그룹 수: 5
교배 확률 : 0.5
참조 확률 : 0.2
가중치 f : 0.95

개체군 수는 최소 20에서 200까지 10씩 차이를 두어서 실험 하였으면 이때 각 개체수마다 최대 목적함수 계산 횟수를 100000으로 설정하여 계산 회수를 초과시 프로세스를 중단하고 그 때까지 발견한 최종해를 비교하였다. 제안된 알고리즘(P-DE)의 그룹 수는 5개로 정하였으며 이는 함수에 따라 최적 값이 다르다. 각 개체수 마다 목적 함수에 사용되는 벡터의 차원 수는 30으로 설정하였으며, 차원이 높을수록 최적해를 찾기 어려

워 진다. 본 실험에서는 다음과 같은 5가지 종류의 최적화 문제를 사용하였다: Sphere function, Rosenbrock's saddle function, Shifted Rotated Ackley's Function, Rastigrin's function, Griewangk's Function.

1) Sphere function

$$f(x) = \sum_{i=1}^{D} x_i^2 \qquad [-100,100] \quad (6)$$

Sphere function 은 단일 모드로 이뤄진 매우 단순한 최소화 문제이다. 최적해는 f(0,...0)=0 이다.

2) Rosenbrock's saddle function

$$f(x) = \sum_{i=1}^{D} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \qquad [-30,30] \quad (7)$$

Rosenbrock's saddle function은 비선형 함수로서 최적화 알고리즘을 테스트하기 위해서 많이 사용되는 대표적인 함수이다. 이 함수는 함수 값의 변화가 급격하고 최적해 주위라 하더라도 최적 해를 찾기가 쉽지 않다. 최적해로는 $f(1,...1)=0$ 이다.

3) Rastigrin's function

$$f(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10] \qquad [-5.12,5.12] \quad (8)$$

Rastigrin's function 역시 많은 지역해가 전역 최적해를 감싸는 형태로 이루어져 있다. 최적해는 $f(0,...0)=0$ 이다.

4) Shifted Rotated Ackley's Function

$$f(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i) + 20 + e \; [-32,32] \quad (9)$$

Shifted Rotated Ackley's Function은 수많은 지역 최적해를 가지고 있으며 전역 최적해가 봉우리 형태로 솟아 올라있는 형태이다. 최적해는 $f(0,...0)=0$ 이다.

5) Griewangk's Function

$$f(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1 \qquad [-600,600] \quad (10)$$

Griewangk's function이 적합도 함수로 사용되었다. 이 함수 역시 최적화 문제를 테스트 하는 목적으로 널리 쓰이고 있는 함수로, 지역 최적해들이 전역 최적 해를 둘러싸고 있다. 최적해는 $f(0,...0)=0$ 이다.

3-2. 실험 결과

기존의 차분 진화 알고리즘과 제안된 차분 진화 알고리즘(P-DE)의 성능은 앞의 5가지 최적화 문제들을 통하여 비교, 분석되었다. 실험을 통하여 얻어진 기존의 차분 진화 알고리즘과 제안된 차분 진화 알고리즘(P-DE)의 결과는 그림 1 - 그림 5에 나타나 있다.

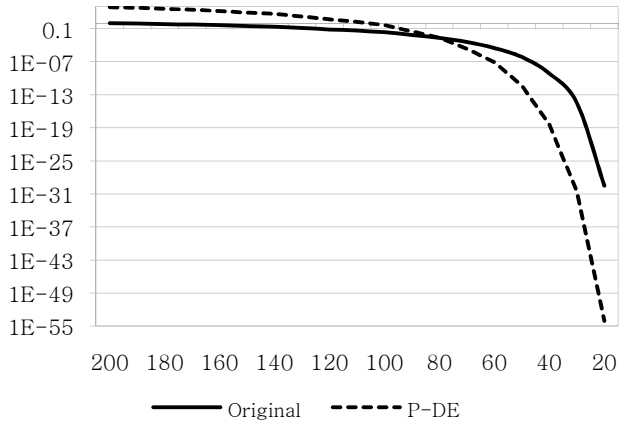


그림 1 Sphere function에서의 성능 비교

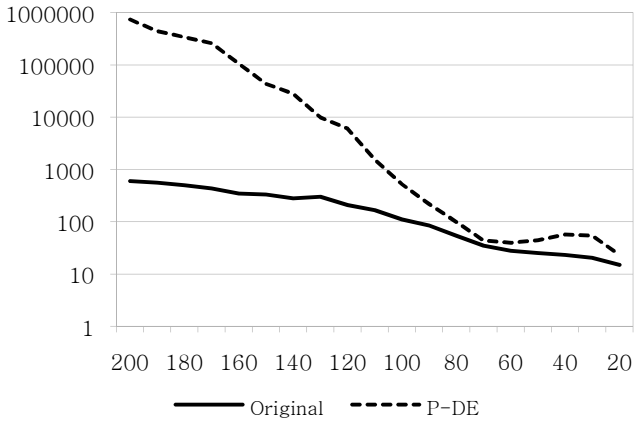


그림 2 *Rosenbrock's* function에서의 성능 비교

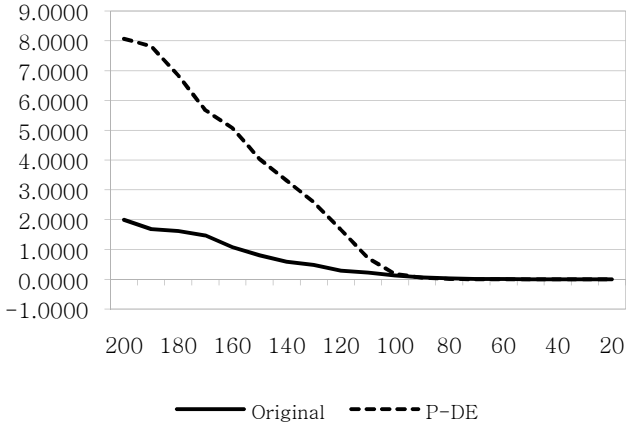


그림 3 Rastigrin's function에서의 성능 비교

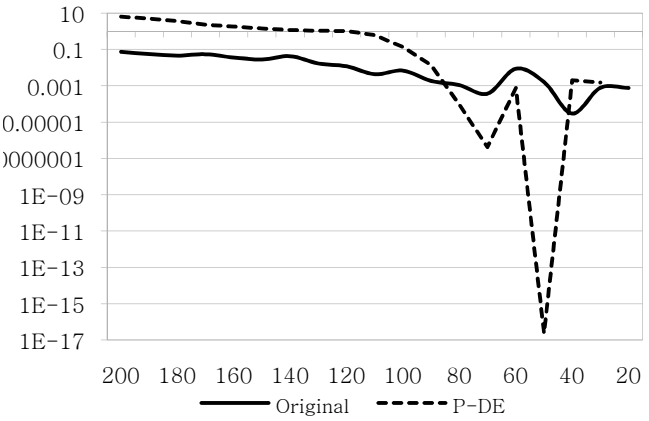
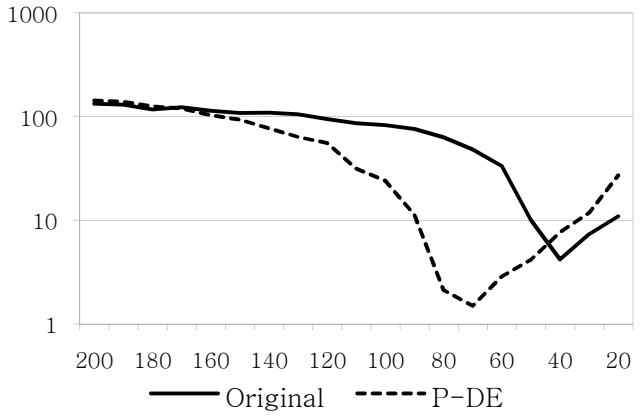


그림 4 Shifted Rotated Ackley's Function에서의 성능 비교



□ □ **5 Griewangk's Function 에서의 성능 비교**

실험 결과, Rosenbrock's saddle function을 제외한 모든 함수에서 우수한 성능을 보였다. Rosenbrock's saddle function 역시 거의 유사한 수준의 성능을 발휘하는 것을 확인할 수 있었다. 전체적으로 제안 알고리즘(P-DE)에서는 객채수가 많아질수록 그 성능이 기존의 알고리즘에 비해서 떨어지게 되는 경향을 보였다. 즉, 개체수가 적어질수록 제안 알고리즘(P-DE)에서 보다 우수한 성능을 나타내었다.

그림 1은 Sphere function의 결과로서 100개 이상의 개체수를 사용하게 될 경우 제안된 알고리즘(P-DE)이 기존의 알고리즘 보다 성능이 떨어지게 되지만 100개 이하의 개체수를 사용하게 되는 경우 전체적으로 성능이 제안된 알고리즘(P-DE)이 우수하게 나오는 것을 확인할 수 있었다.

그림 2는 Rosenbrock's saddle function의 결과를 나타내며 전체적으로 기존 알고리즘보다 제안된 알고리즘(P-DE)이 성능이 떨어지는 결과를 가져왔다. 하지만 일부 구간에서는 비슷한 성능을 발휘하게 되는 것을 알 수가 있었다.

그림 3은 Rastigrin's function의 결과를 나타내며, 제안 알고리즘(P-DE)이 비교적 우수한 성능을 보였다. 하지만, 개체수가 적은 상태에서는 우수한 성능을 보인 반면 50 근처에서 성능이 떨어지는 현상을 보이는 경우가 발생하였다. 이러한 문제점은 Rastigrin's function 에서 사용되는 개체수 50개 부근에서 제대로 성능을 발휘 하지 못하는 경우라고 할 수가 있다. 이러한 문제는 그룹수의 조정을 통해서 어느 정도 해결이 가능 할 것이라고 보인다.

그림 4는 Shifted Rotated Ackley's Function의 결과를 보여주는 그래프로서 전체적인 양상은 비슷하지만 그 변동폭이 기존의 알고리즘에 비해서 비교적 크다고 볼 수 있다. 특히 개체수가 40인 부분에서 기존 알고리즘에 비해서 우수한 성능을 발휘 하는 것을 알 수가 있다

그럼 5는 Griewank's Function의 결과를 나타낸. 그래프로서 개채수가 160이하인 지점에서 제안된 알고리즘(P-DE)의 성능이 우수하게 나오는 것을 볼 수가 있었다.

전체적으로 제안된 알고리즘(P-DE)의 성능이 기존 알고리즘의 성능에 비해서 우수하게 결과가 나왔다. 또한 성능이 떨어지는 경우 기존 알고리즘과 비슷한 성능을 발휘 하는 것을 알 수 가 있었다. 특히 지역 최적해들이 다수 분포하여 전역 최적해를 감싸는 형태의 목적함수(3)(4)(5)에서 더 나은 성능을 발휘 하는 것을 볼 때 지역해 수렴 문제에서 강한 성능을 발휘 하는 것을 알 수가 있다. 표 1은 각 목적 함수 마다 최적해들의 값과 이때 사용된 개채수의 값이다.

표 1 실험 결과

	DE		P-DE	
function	*NP*	Value	*NP*	Value
f_1	20	3.08E-30	20	7.6347E-55
f_2	20	14.87377	20	15.7958207
f_3	40	0.0002818	60	1.502474
f_4	50	2.931E-05	40	1.52E-018
f_5	40	4.19207	70	1.0098532

4. 결 론

본 논문은 기존의 차분 진화 알고리즘이 가지고 있는 지역해 수렴 문제를 병렬 처리를 통해서 해결하여 좀 더 빠른 수렴 속도와 최적화를 이룰 수 있는 기법을 제안 하였다.

기존의 차분진화 알고리즘에서는 최적화 과정에서 지역해 수렴 문제를 해결하지 못하고 지역 최적해 주변에서 값들이 머물게 되는 경우가 많았다. 우리는 이러한 문제를 병렬 처리를 통해서 다수의 best value의 생성을 통해서 더 넓은 공간 탐색을 할 수 있도록 하였다. 5가지의 최적화 문제를 통해서 기존 알고리즘 보다 우수한 성능을 발휘 한다는 것을 알 수가 있었다. 특히 지역 최적해들이 전역 최적해를 감싸는 형태의 문제들에서 그 성능이 더욱 좋아지는 것을 확인할 수 있었다.

이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (과제번호 2010-0015520).

5. 참고 문헌

[1] R. Storn and K. Price, "Differential Evolution a simple and efficient heuristic for global optimization over continuous spaces," J. Global Optimization, vol. 11, no. 4, pp. 341-359, 1997.

[2] K. V. Price, R. M. Storn, and J. A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, 1st ed. New York: Springer-Verlag, Dec. 2005.

[3] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis,

"Parallel differential Evolution," in IEEE Congress on Evolutionary

[4] V.P. Plagianakos and M.N. Vrahatis, "Parallel evolutionary training algorithms for `hardware.friendly" neural networks, Natural Computing 1 (2002), 307.322

[5] J. Zhang and A. C. Sanderson, "JADE: Adaptive Differential Evolutionwith Optional External Archive," IEEE Transactions on Evolutionary Computation, in press, 2008.

[6] Geist, A.Beguelin, J.Dongarra, W.Jiang, R.Manchek, and V.Sunderam, Pvm: Parallel virtual machine. a user's guide and tutorial for networked parallel computing, MIT Press, Cambridge, 1994.

[7] R. Storn and K. V. Price "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," J. Global Optimization, vol. 11, pp. 341, 1997

[8] S. Das , A. Abraham and A. Konar "Adaptive clustering using improved differential evolution algorithm," IEEE Trans. Syst., Man, Cybern. A, vol. 38, pp. 218, Jan. 2008.

[9] J. Kennedy and R. Eberhart "Particle swarm optimization," Proc. IEEE Int. Conf. Neural Netw., 1995, p. 1942.

[10] J. Liu and J. Lampinen "A fuzzy adaptive differential evolution algorithm," Soft Computing-Fusion Found., Methodologies Applicat., vol. 9, pp. 448, 2005.

[11] H. Abbass "The self-adaptive pareto differential evolution algorithm," Proc. 2002 Congr. Evol. Comput. Honolulu, HI, vol. 1, May 2002, p. 831.

[12] M. Mitchell, An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press, 1996.

[13] D.E. Goldberg - Genetic Algorithms in Search, Optimization, and Machine Learning Reading, MA: Addsion Wesley, 1989.

[14] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," Soft Comput.: Fusion Found., Methodologies Applicat., vol. 10,no. 8, pp. 673-686, 2006.

[15] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in Proc. IEEE Congr. Evol.Comput., vol. 2. Sep. 2005, pp. 1785-1791.

[16] Wenjun Zhang, Xiaofeng Xie, "DEPSO: Hybrid particle swarm with differential evoluion operator," IEEE Int. Conf. on System, Man & Cybernetics (SMCC), Washington, USA, 3816-3821, 2003.