J. of university of anbar for pure science : Vol.5:NO.2 : 2011

ISSN: 1991-8941

Differential evolution for neural networks learning enhancement.

Abdul Sttar Ismail wdaa
College of Education for pure sciences- University of Anbar
Received: 21/7/2010 Accepted:8/5/2011

Abstract**:**In this paper ,we use new treatment ,Differential Evolution,, Differential Evolution (DE) has been used to determine optimal value for ANN parameters such as learning rate and momentum rate and also for weight optimization. In ANN, there are many elements need to be considered, and these include the number of input nodes, hidden nodes, output nodes, learning rate, momentum rate, bias parameter, minimum error and activation/transfer functions. Three programs have developed; Differential Evolution Neural Network (DENN), Genetic Algorithm Neural Network (GANN) and Particle Swarm Optimization with Neural Network (PSONN) to probe the impact of these methods on ANN learning using various datasets. The results have revealed that DENN has given quite promising results in terms of convergence rate and smaller errors compared to PSONN and GANN.

Key words : Differential evolution , neural networks ,learning enhancement.

Introduction:

Evolutionary computing (EC) is an exciting development in Computer Science. It amounts to building, applying and studying algorithms based on the Darwinian principles of natural selection, the main concepts behind evolutionary computing. Figure 1.1 illustrates
the cycle of Evolutionary Computing.

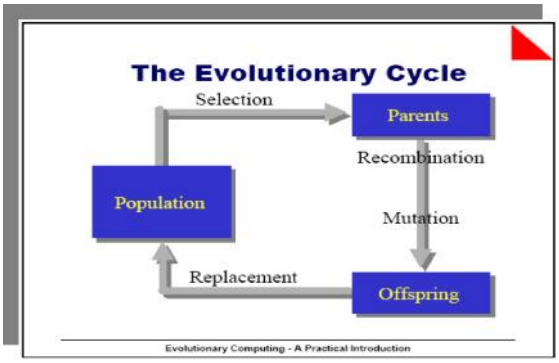


Figure 1.1**: The Evolutionary Cycle [1]**

Evolutionary computing contain a grub of the algorithms mention some of them. Genetic Algorithm (GA) is one of the famous evolutionary techniques in ANN learning. A basic genetic algorithm (GA) comprises of three genetic operators: selection, crossover, and mutation. Starting from an initial population of strings (representing possible solutions), GA uses these operators to calculate successive generations. First, pairs of individuals of the current population are selected to mate with each other to form the offspring, which then form the next generation. Another algorithm similar to GA called (GP).

Differential Evolution (DE) algorithm is an evolutionary algorithm, which was proposed by[2] .It is a small and simple mathematical model of a big and naturally complex process of evolution. So, it is easy and efficient. According to [3], this algorithm is simple and one of the most powerful tools for global optimization. A genetic algorithms, evolution strategies, or evolutionary programming is the three basic trends of evolutionary optimization, also well known under the common term of evolutionary algorithms. Recently with the advent of new ideas and new methods in optimization, including DE too, they have got a second fashionable name of artificial evolution, and DE belongs to this suite. The intelligence usage of differences between: individuals

realized in a simple and fast linear operator, so-called differentiation, makes differential
Evolution unique; hence, Differential Evolution (DE) is proclaimed.

Differential Evolution (DE)

Differential Evolution is a small and simple mathematical model of a big and naturally complex process of evolution. So, it is easy and efficient! First and foremost Differential Evolution (DE) is an optimization algorithm. And without regard to its simplicity DE was and is one of the most powerful tools for global optimization. Perhaps you have heard about genetic algorithms, evolution strategies, or evolutionary programming. These are three basic trends of evolutionary optimization, also well known under the common term of evolutionary algorithms. Lately, with the advent of new ideas and new methods in optimization, including DE too, they have got a second fashionable name of artificial evolution. DE belongs to this suite.

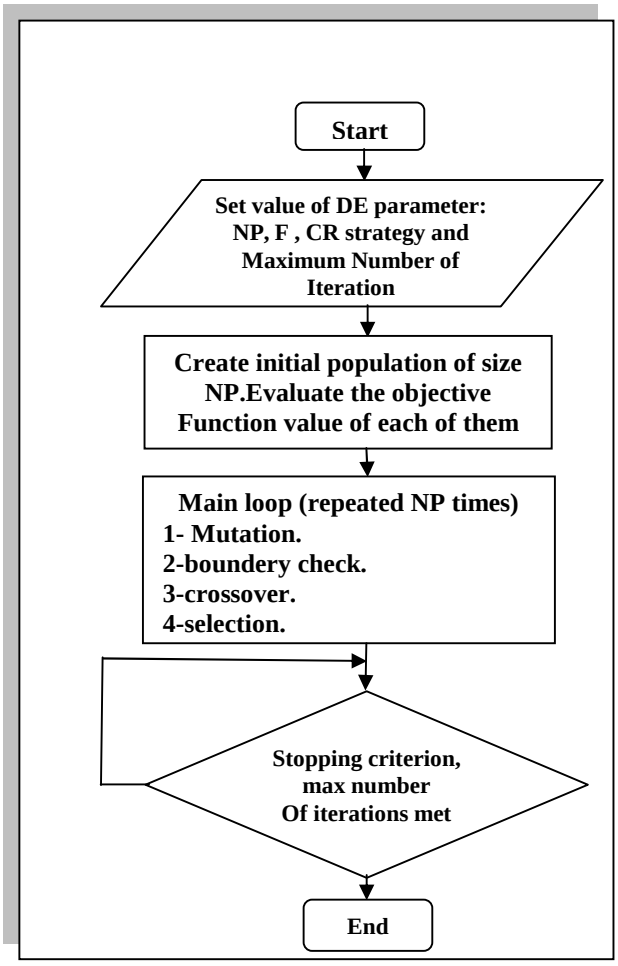


Figure 2.1: Flowchart of DE algorithm.

2.1 Procedure DE

Require**: D problem dimension (optional)**
NP,F,Cr - control parameters
GEN – stopping condition
L,H boundary constraints
Initialize population $Pop_{ij} \leftarrow rond_{ij}$ [L,H] and Evaluate fitness $Fit_j \leftarrow f(Pop_j)$
For **g= 1 to GEN** do
For **j= 1 to NP** do
Choose randomly r1,2,3 ∈ [1…,NP],r1≠r2≠r3≠j
Create trial individual X ← S(r, F,Cr,Pop)
Verify boundary constants if (x_i, ∈ (L,H) xi ← rondi [L,H]
Select better solution (X or Pop_j), and update iBest required

Figure 2.2 **shows the basic pseudo-code for the DE algorithm**

The Basics of Differential Evolution

DE has three operations: Mutation, crossover and selection.

Mutation.

For each target vector x_{ij}^{t} , a mutant vector v is generated according to.

$$V_i^{t+1} = x_{r1}^{t} + F.(x_{r2}^{t} - x_{r3}^{t}).$$

The r1, r2 and r3 are randomly chosen indexes and r1, r2, r3 {1, ∈ ..,NP}.

F is a real number to control the amplification of the difference vector.

$(x_{r2}^{t} - x_{r3}^{t})$

Range of F is in [0, 2] 0<F<=2

2. Crossover

The target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector u.

$$u_{ij}^{t+1} = \begin{cases} v_{ij}^{t+1}, & rand(j) \le CR \quad or \quad j = randn(i) \\ x_{ij}^{t}, & rand(j) > CR \quad and \quad j \ne randn(i) \end{cases}$$

Where j=1,2,…..,D, rand (j) ∈ is the jth

∈ ∈

evolution of a uniform random generator number . r. CR [0, 1] is the crossover probability constant, which has to be determined previously by the user rand (i)

v_i^{t+1} (1, 2,….D) is a randomly chosen index which ensures that u_i^{t+1} gets at least one element from u_i^{t+1} .Otherwise, no new parent vector would be produced and the population would not alter

3. Selection

DE adapts greedy selection strategy. If and only if, the trial vector u_i^{t+1} yields a better fitness function value than x_i^t then u_i^{t+1} is set to .Otherwise, the old value is x_i^t retained.

Differential Evolution Training Algorithm

Differential evolution can be classified as a floating-point encoded evolutionary algorithm for global optimization over continuous spaces. As such, it can be applied to global searches within the weight space of a typical feed-forward neural network. Output of a feed-forward neural network is a function of synaptic weights W and input values x, i.e. , y = f (x , W). In standard training processes, both the input vector x and the output vector y are known. The synaptic weights in W are adapted to obtain appropriate functional mappings from the input x to the output y. Generally, the

adaptation can be carried out by minimizing the network error function E which is of the form.

E (y , f (x, W)) : (y D1 ,x D 2 ,x D 3 , f) → R . (1)

The optimization goal is to minimize the objective function E (y , f (x, W)) by optimizing the values of the network weights (now D = D3)

W=(W1,…….WD) (2)

Similar to other evolutionary algorithms, DE operates on a population, PG,of candidate solutions ,not just a single solution. These candidate solutions are the individuals of the population. DE maintains a population of constant size that consists of NP,real-value vectors, Wi ,G, wher is i index to the population and G is the generation to which the population belongs.

PG=(W1,G,….,WNP,G), G=0,…,Gmax (3)

Additionally, in network training each vector contains D network weights (chromosomes of individuals):

Wi ,G = (w1,i,G ,….w D,i,G), i =1,NP , G = 0,..Gmax (4)

DE's self-referential population reproduction scheme is different from other evolutionary algorithms. After initialization of the first population, vector s in the current population, PG+١,are randomly sampled and combined to create candidate vectors for the subsequent generation, PG+١.The population of candidates, trial vectors PG+١= Ui,G+١= uj,i,G+١ is generated as follows:

uj,i,G+١ = w j,r3,G + F . (wj,r1,G - wj,r2,G)

uj,i,G+1 = uj,i,G+1 , if rand j [0, 1] ≤ CR (5)

w j,i,G , otherwise

Where

i=1,….NP , j = 1,……D

r1, r2, r3 ⊂ {1……..NP }, randomly selected except r1 ≠ r2 ≠ r3 ≠ i

CR ⊂ [0, 1], F (≥ 1)

CR is a real-valued crossover factor that controls the probability that a trial vector parameters will come from the randomly chosen, mutated vector uj,i,G+١ instead of the current vector wj,i,G. In general, F and CR affect the convergence speed and robustness of the search process. Their optimal values depend both on objective function characteristics and the population size NP,and thus, the selection of optimal parameter values is an application dependent task[4,5].

DE's selection scheme also differs from other evolutionary algorithms. The population for the next generation, P G+١,is selected from the current population PG or the child population according to the following rule

W i,G+1 = U i,G+1 , if E (y , f(x ,W i, G+1) ≤

E (y , f(x ,W i, G))

Otherwise W i, G

Thus, each individual of the temporary population is compared to its counterpart in the current population. Assuming that the objective function is to be minimized, the vector with the lower objective function value wins a place in the next generation's population. As a result ,all the individuals of the next generation are as good as or better than their counterparts in the current generation. The interesting point concerning DE's replacement scheme is that a trial vector is only compared to one individual,

not to all individuals in the current population. It is also noteworthy that the replacement scheme ensures that the population does not diverge from or lose the best solution found so far.

EXPERIMENTAL RESULT

Experimental results of the study. Three programs have been developed: Differential Evolution Forward Feed Neural Network (DENN), Particle Swarm Optimization Feed Forward Neural Network (PSONN) and Genetic Algorithm Feed Forward Neural Network (GANN) using four dataset: XOR, Cancer , heart and Iris. The results for each dataset

are compared and analyzed based on the convergence rate and classification performance.

Results on XOR Dataset

Table 4.1: Result of DENN , PSONN and GANN on XOR dataset

	DENN	PONN	GANN
Learning Iteration	41	51	61
Error Convergence	0.0048865	0.00473763	0.04125
Convergence Time	7 Sec	12 Sec	37 Sec
Classification (%)	98.97	95.17	85.66

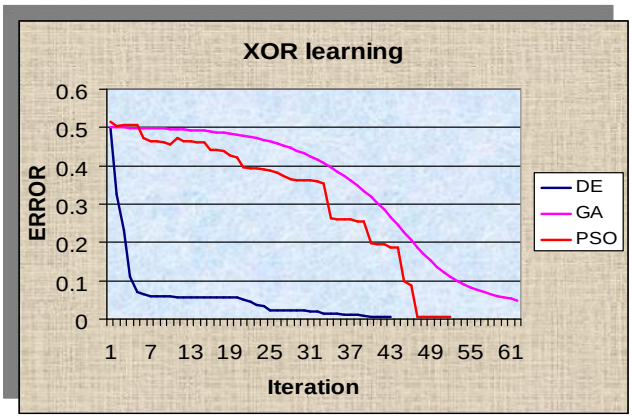


Figure 4.1: **Convergence of XOR dataset**

Results on Cancer Dataset

Table 4.2: Result of DENN , PSONN and GANN on Cancer dataset

	DENN	PSONN	GANN
Learning Iteration	443	219	10000
Error Convergence	0.00499	0.004870	0.50049
Convergence Time	195 Sec	110 Sec	273 Sec
Classification (%)	98.40	98.65	97.73

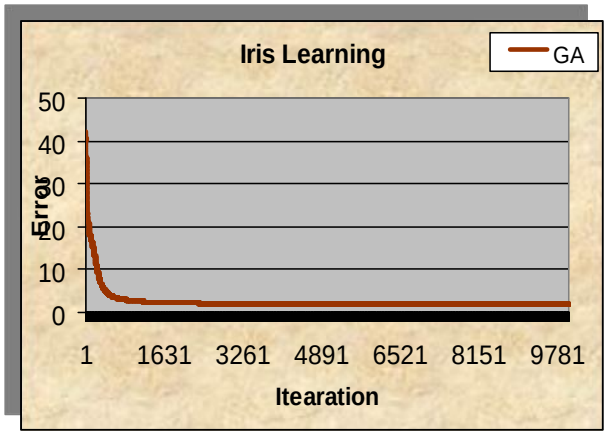


Figure 4.2: **GA Convergence of iris dataset**

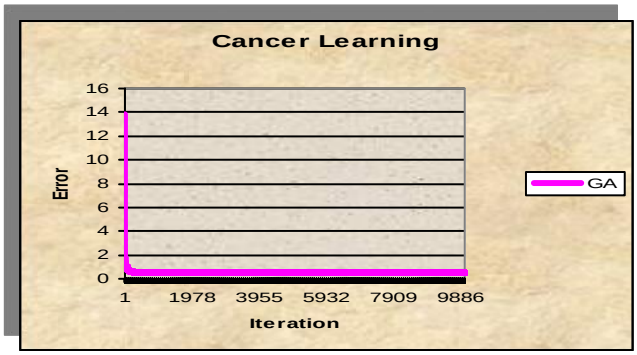


Figure 4.3: **GA Convergence of Cancer dataset**

Results on Iris Dataset

Table 4. 3: Result of DENN, PSONN and GANN on Iris dataset

	DENN	PSONN	GANN
Learning Iteration	61	818	10000
Error Convergence	0.049803	0.049994	1.88831
Convergence Time	16 Sec	170 Sec	256 Sec
Classification (%)	95.014972	93.86	97.72

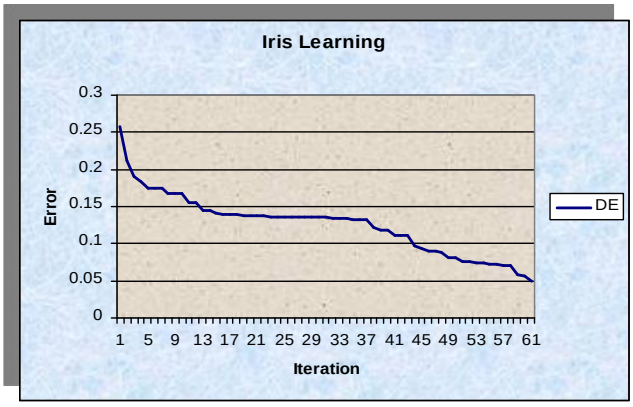
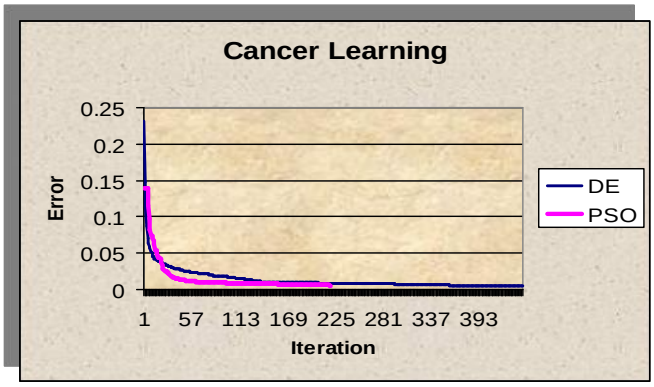


Figure 4.4: **DE Convergence of Iris dataset**



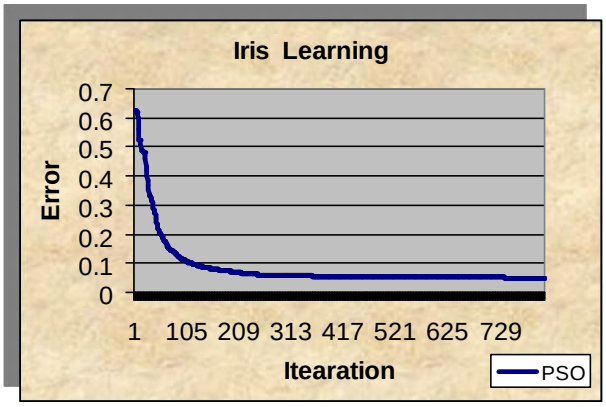


Figure 4.6: **PSO Convergence of Iris dataset**

Figure 4.8: **PSO,GA Convergence of Heart dataset**

	DENN	PSONN	GANN
Learning Iteration	**58**	**10000**	**9000**
Error Convergence	**0.048925**	**1.46392**	**3.00**
Convergence Time	**16 Sec**	**170 Sec**	**110 Sec**
Classification (%)	**85.50**	**89.56**	**92.83**

4. 4 Results on Heart Dataset

Table 4.4: **Result of DENN , PSONN and GANN on Heart dataset**

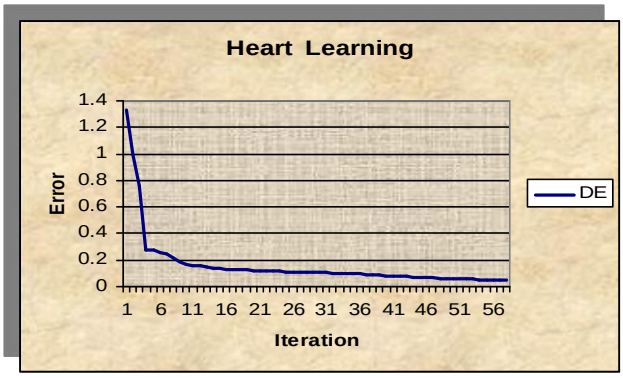
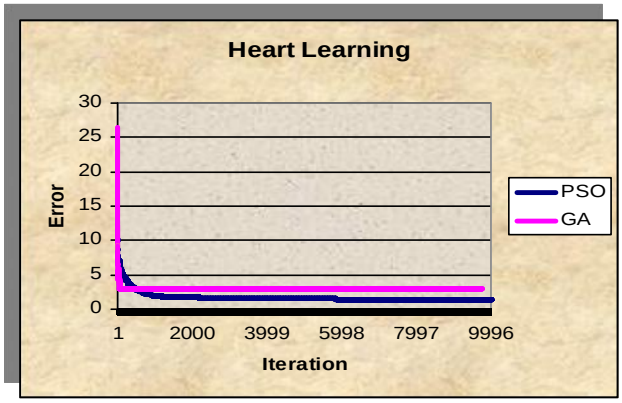


Figure 4.7: **DE Convergence of Heart dataset**



Comparison DENN ,PSONN and GANN

This analysis is carried out to compare the results between PSONN and GANN. To do this, the learning patterns for all algorithms are compared using all Four datasets. The comparative correct classification percentage for all datasets is shown in figure (4.9)

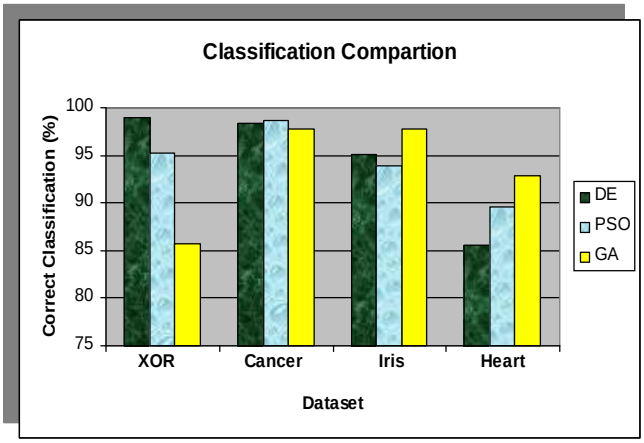


Figure 4.9: **Comparative of correct classification Percentage DENN, PSONN and GANN**

For XOR dataset, the results show that DENN has better results on convergence time and correct classification percentage. DENN converges in a short time with high correct classification percentage. For Cancer dataset, PSONN classification results are better than DENN and GANN although both algorithms do not converge to the solution within specified minimum error, but it shows that at this time, GANN classification results are better than PSONN. But in terms of convergence time, it shows that PSONN is better than DENN, and GANN significantly reduces the error with minimum iterations. For overall performance, the experiments show that PSONN produces feasible results in terms of convergence time and classification percentage.

Discussions

In this study, a differential evolution (DE) training algorithm was used to train feed forward term for network weights and biases (MSEREG) was included, the performance of the differential evolution training algorithm appeared to be comparable to that of the gradient based methods. Due to its scalability, it is evident that the DE method can be applied to the training of enormous neural networks. Based on the results, it is clear that DENN is better than PSONN and GANN in term of convergence time.

Choosing PSONN parameters also depend on the

problem and dataset to be optimized. This parameter can be adjusted to achieve better optimization. But to have better comparison in this study, the same parameters for all four datasets have been used. For GANN, learning rate and momentum rate which critical for standard learning is provided by GA algorithm with a set of weight. This is to ensure the convergence time is faster with better results. Although Standard BP learning becomes faster based on parameters provided by GA, the overall process including parameters selection in GA takes much time compared to overall process in PSONN.

Conclusions

The DE is successfully applied in neural network and has been tested using XOR, Cancer, Heart and Iris datasets. The analysis is done by comparing the results for each dataset produced by DENN, PSONN and GANN. Based on the analyses, it shows that DE is successfully in the some of dataset like xor and cancer , applied in neural network and produced better result compared to PSONN, GANN.

References

1. Yao, X. (1999). 'Evolving artificial neural networks', Proceedings of the IEEE. vol. 87, no. 9: 1423 -1447.
2. Storn,R. and Price,K.: Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces,Technical Report -95-012,International Computer Science Institute,Berkeley,CA,USA
3. Hussein A. Abbass, Ruhul Sarker, and harles Newton. A pareto differential evolution approach to vector optimization problems. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC2001, Seoul, Korea,IEEE Press, 2001.
4. Storn,R. and Price,K.: Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization Over ontinuous Spaces,Technical Report TR-95-012,International Computer Science Institute,Berkeley,CA,USA http://www.icsi. berkeley.edu/ techreports/1995.abstracts/tr-95-012.htm), 1995.
5. Storn,R. and Price,K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 11(4) 1997,341–359.

استخدام التطور التفاضلي في الشبكة العصبية لتحسين التعلم للشبكة

عبد الستار اسماعيل وداعه

E-mail:sttarwdaa@yahoo.com

الخلاصة:خوارزمية (Differential Evolution) طبقت على (feed forward) للشبكة العصبية الاصطناعية لتحسين عملية التعلم للشبكة أثبتت هذه الخوارزمية نسبة تقارب جيده ودقة تصنيف عالية, حيث طبقت ثلاث برامج لأجراء هذه المقارنة: التطور التفاضلي للشبكة العصبية (Differential Evolution For Neural Network) – الخوارزمية الوراثية للشبكة العصبية (Genetic Algorithm For Neural Network) أمثلية حشد الجزيئات للشبكة العصبية (Partial Swarm Optimization For Neural Network). لمعرفة تأثير كل من هذه الخوارزميات طبقت على أربعة انواع من البيانات (XOR) – (Iris) – (Cancer) – (Heart) . كشفت النتائج إن خوارزمية التطور التفاضلي (DE) وفي بعض من البيانات أعطت نتائج واعده جدا من ناحية نسبة التقارب وتقليل نسبة الخطأ مقارنة مع (GA) و (PSO) .