Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 60 (2013) 16333-16340

ISSN: 2229-712X

An Improved DEPSO with Adaptive Parameters for Global Optimization Problems

Rajashree Dash and Srotasmita Mahapatra

School of Computer Science & Engineering, ITER, Siksha O Anusandhan University, Bhubaneswar, Orissa, India.

ARTICLE INFO

Article history:

Received: 29 April 2013;

Received in revised form: 24 June 2013;

Accepted: 9 July 2013;

Keywords

Global Optimization,

PSO,

DE,

DEPSO,

Improved DEPSO.

ABSTRACT

Population based stochastic search techniques inspired by nature has attracted much attention recently as an effective approach for solving numerical optimization problems. Two of the most commonly used population-based global optimization algorithms are PSO and DE. Although both have been successfully applied to a wide range of test and real life problems but they have certain shortcomings associated with them. Since DE utilizes the differential information to get the new candidate solution, sometimes it results in instability of performance. Again DE has no mechanism to memory the previous process and use the global information about the search space, so it results in a waste of computing power and may get trapped in local optima. Although PSO converges quickly but, easily gets stuck in local optima because of loss of diversity of swarm. In this paper a simple hybrid version of DEPSO has proposed. DEPSO combines the differential information obtained by DE with six mutation policies and the memory information extracted by PSO to create the promising solutions. Again a time decreasing parameter tuning has been proposed which brings in a significant improvement of the performance of DEPSO. Performances are presented through the tests of four complex benchmark functions with three different dimensionalities. The experimental results show that the convergence rates and qualities of solutions are greatly improved.

Introduction

Optimization can be defined as an art of searching the best alternative among a given set of options. Many problems can eventually be reduced to optimization in various disciplines such as engineering designs, agricultural sciences, manufacturing systems, economics, physical sciences, control engineering, pattern recognition and statistics etc. Optimization can thus be viewed as one of the major quantitative tools in network of decision making, in which decisions have to be taken to optimize one or more objectives in some prescribed set of circumstances. The increasing practical utility of optimization problems in different fields put challenges before researchers to develop more efficient and robust computational algorithms which can numerically solve on computers the mathematical models of medium as well as large size optimization problem.

During the past two decades, Population-based global optimization algorithms is becoming more attractive than others computing strategy. A common feature of all population-based algorithms is that the population consisting of possible solutions to the problem is modified by applying some operators on the solutions depending on the information of their fitness. Hence, the population is moved towards better solution areas of the search space. Two important classes of population-based optimization algorithms are evolutionary algorithms and swarm intelligence-based algorithms. Although Genetic Algorithm (GA), Genetic Programming (GP), Evolution Strategy (ES) and Evolutionary Programming (EP) are popular evolutionary algorithms, GA is the most widely used one in the literature. GA is based on genetic science and natural selection and it attempts to simulate the phenomenon of natural evolution at genotype level while ES and EP simulate the phenomenon of natural evolution at phenotype level. One of the evolutionary algorithms which have been introduced recently is Differential Evolution (DE) algorithm. In the basic GA, a selection operation is applied to the solutions evaluated by the evaluation unit. At this operation the chance of a solution being selected as a parent depends on the fitness value of that solution. One of the main differences between the GA and the DE algorithm is that, at the selection operation of the DE algorithm, all solutions have an equal chance of being selected as parents, i.e. the chance does not depend on their fitness values. In DE, each new solution produced competes with its parent and the better one wins the competition. In recent years, swarm intelligence has also attracted the interest of many research scientists of related fields. A swarm can be considered as any collection of interacting agents or individuals. An ant colony can be thought of as a swarm whose individual agents are ants; a flock of birds is a swarm of birds. An immune system can be considered as a swarm of cells and molecules as well as a crowd is a swarm of people. A popular swarm-intelligence-based algorithm is the Particle Swarm Optimization (PSO) algorithm which was introduced by Eberhart and Kennedy in 1995. PSO is also a population-based stochastic optimization technique and is well adapted to the optimization of nonlinear functions in multidimensional space. Empirical and theoretical studies have shown that the convergence behavior of PSO is strongly dependent on the values of the inertia weight and the acceleration coefficients. Wrong choices of values for these parameters may result in divergent or cyclic particle trajectories. By modulating the PSO parameters, convergence can be

Tele:

E-mail addresses: rajashree_dash@yahoo.co.in

speeded up and the ability to find the global optimal can be enhanced. But the main disadvantage of PSO is that it depends upon its local optima and the method is slow down at its global optimum. Therefore, accelerating convergence speed and avoiding the local optima have become the two most important and appealing goals in PSO research. The performance of DE, on the other hand, is influenced mainly by the scale parameter and the probability of recombination. Although recommendations for values of these parameters have been made in the literature, these values are not universally applicable. The best values for DE control parameters remain problem dependent, and need to be fine tuned for each problem.

DE algorithm has some advantages, such as its ability to maintain the diversity of population, and to explore local search, but it has no mechanism to memory the previous process and use the global information about the search space, so it results in a waste of computing power and may get trapped in local optima. The differential information can be helpful for the search ability, but it also leads to instability of some solutions. Although PSO converges quickly, easily gets stuck in local optima because of loss of diversity of swarm [1]. A number of variations of both PSO and DE have been developed in the past decade to improve the performance of these algorithms. One class of variations includes hybridization of PSO and DE, where the advantages of the two approaches are combined [2, 3, 6, 7, 8].

In this paper a simple hybrid version of DEPSO has proposed. Initial population set is the parent vector for both DE and PSO. For each individual a new position is obtained using Particle Swarm's velocity and position update equations and a mutant vector is created using mutation operator of DE. Then a trial vector is created applying crossover on PSO updated position and mutant vector. Comparing the fitness values of trial vector with the original vector, a candidate solution set is obtained either from parent or trial vector and accordingly the *pbest* and *gbest* position of the particles are updated. The method is repeated iteratively till the optimum value is reached. The inclusion of PSO phase creates a perturbation in the population, which in turn helps in maintaining diversity of the population and producing an optimal solution.

Although the algorithm takes the advantage of both DE and PSO still its performance is dependent on parameter setting. Choosing suitable parameter values is a problem dependant task and generally requires time-consuming trial-and error parameter tuning process. This approach is not appropriate if the global optimization is required in an automated environment or if the user has no experience in the fine art of the control parameter tuning. Thus, to obtain optimal performance, time decreasing parameter tuning is necessary [4]. The inertia weight is designed as a tradeoff between the global and local search. Larger inertia weight facilitates global exploration while lower values encourage a local search. Similarly a bigger value of the differential factor may accelerate the convergence rate and increase more chances for the algorithm to converge a local optimum, while a smaller differential factor could have a slow convergence rate and more opportunities to obtain the global solution. The crossover probability has much influence on the population diversity and the convergence rate. As the mutation operator is a crucial idea in DE for generating mutant vectors while the crossover was claimed to be unimportant hence in this paper, a dynamic adjustment approach for differential factor and inertia weight has been proposed to accelerate the convergence rate. Performances are presented through the tests of four complex benchmark functions with three different dimensionalities. The experimental results show that the convergence rates and qualities of solutions are greatly improved.

Particle Swarm Optimization Algorithm (PSO)

PSO is a robust stochastic optimization technique based on the movement and intelligence of swarms. PSO applies the concept of social interaction to problem solving. It was developed in 1995 by James Kennedy (social-psychologist) and Russell Eberhart (electrical engineer). It uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution. Each particle is treated as a point in an N-dimensional space which adjusts its "flying" according to its own flying experience as well as the flying experience of other particles. PSO uses a population of individuals, to search feasible region of the function space. In this context, the population is called ***swarm*** and the individuals are called ***particles.*** Every single solution (called a particle) "flies" over the solution space with a velocity, which is adjusted at each time step in search for the optimal solution. The particles are evaluated using a fitness function to see how close they are to the optimal solution. The particle flies towards a position, which depends on its own past best position and the position of the best of its neighbours. The quality of a particle position depends on a problem specific objective function. Particles are initialized randomly and updated afterwards according to:

$$X_{ij}^{k+1} = X_{ij}^{k} + V_{ij}^{k+1} \quad \textbf{(1)}$$

$$V_{ij}^{k+1} = \omega V_{ij}^{k} + c_1 r_1 (pBest_{ij}^{k} - X_{ij}^{k}) + c_2 r_2 (gbest_{j}^{k} - X_{ij}^{k}) \quad \textbf{(2)}$$

Where *w*, *c₁*, *c₂* are inertia, cognitive and social acceleration constants respectively, r_1 and r_2 are random numbers within [0, 1]. W is set within (0.1) and c1 and c2 within (0.2]. The best solution of the particle achieved so far is represented by *lbest* , which indicates the tendency of the individual particles to replicate their corresponding past behaviors that have been successful. The global best solution so far is represented by *gbest*, which indicates the tendency of the particles to follow the success of others. A large inertia weight (*w*) facilitates a global search while a small inertia weight facilitates a local search. By linearly decreasing the inertia weight from a relatively large value to a small value through the course of the PSO run gives the best PSO performance compared with fixed inertia weight settings [9].

Steps of PSO based learning is:

Step 1: Initializing D-dimensional N particles, together called a swarm, where each particle position represents a trial solution to the optimization problem. Initialize local best position and global best position of all particles.

Step 2: Evaluating the desired optimization fitness function in d variables, for each particle.

Step 3: Comparing particle's fitness evaluation with particle's lbest. If current value is better than lbest, then set lbest value equal to the current value and the lbest location equal to the current location in d-dimensional space.

Step 4: Comparing fitness evaluation with the population's overall previous best. If the current value is better than gbest, then reset gbest to the current particle's value.

Step 5: Changing the velocity and position of the particle according to the equations (1) and (2), respectively:

Step 6: Loop to step 2 until the criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

Differential Evolution Algorithm (DE)

Differential Evolution (DE) is a population-based stochastic function optimizer, which uses a rather greedy and less stochastic approach for problem solving in comparison to classical evolutionary algorithms, such as genetic algorithms, evolutionary programming, and PSO. DE combines simple arithmetical operators with the classical operators of recombination, mutation, and selection to evolve from a randomly generated starting population to a final solution. DE also incorporates an efficient way of self-adapting mutation using small populations. It is able to reproduce the same results consistently over many trials unlike PSO which is more dependent on the randomized initialization of individuals. DE algorithm is like genetic algorithm using similar operators; crossover, mutation and selection. The main difference in constructing better solutions is that genetic algorithms rely on crossover while DE relies on mutation operation [5, 10]. The optimization process is conducted by means of three main operations: mutation, crossover and selection. In each generation, individuals of the current population become target vectors. For each target vector, the mutation operation produces a mutant vector, by adding the weighted difference between two randomly chosen vectors to a third vector. The crossover operation generates a new vector, called trial vector, by mixing the parameters of the mutant vector with those of the target vector. If the trial vector obtains a better fitness value than the target vector, then the trial vector replaces the target vector in the next generation. The mutant vector can be generated using any one of the following strategies:

DE/Rand1

$$v_{ij} = x_{r_1 j} + F \times (x_{r_2 j} - x_{r_3 j}) \quad \textbf{(3)}$$

$$where\ r_1 \neq r_2 \neq r_3 \neq i$$

DE/Rand2

$$v_{ij} = x_{r_1 j} + F \times (x_{r_2 j} - x_{r_3 j}) + F \times (x_{r_4 j} - x_{r_5 j}) \quad \textbf{(4)}$$

$$where\ r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$$

DE/Best1

$$v_{ij} = x_{best,j} + F \times (x_{r_1 j} - x_{r_2 j}) \quad \textbf{(5)}$$

$$where\ r_1 \neq r_2 \neq i$$

DE/Best2

$$v_{ij} = x_{best,j} + F \times (x_{r_1 j} - x_{r_2 j}) + F \times (x_{r_3 j} - x_{r_4 j}) \quad \textbf{(6)}$$

$$where\ r_1 \neq r_2 \neq r_3 \neq r_4 \neq i$$

DE/Current to Best

$$v_{ij} = x_{ij} + F \times (x_{best,j} - x_{ij}) + F \times (x_{r_1 j} - x_{r_2 j}) \quad \textbf{(7)}$$

$$where\ r_1 \neq r_2 \neq i$$

DE/(Centroid mutation)

$$v_{ij} = (x_{best,j} + x_{r_1 j} - x_{r_2 j})/3 + F \times (x_{r1j} - x_{r_2 j}) \quad \textbf{(8)}$$

$$where\ r_1 \neq r_2 \neq i$$

The random numbers used in mutation are mutually exclusive integers generated in the range [1, np) and F is the scaling factor.

Steps of DE based learning are:

Step 1: Initialize the position of each individual according to the population size.

Step 2: Find the fitness function value of each individual.

Step 3: Create a mutated individual v_i for each individual target vector x_i using a suitable mutation equation.

Step 4: Create a new vector called trial by mixing the parameters of the mutant vector with those of the target vector by using the following crossover operation:

$$u_{ij} = v_{ij}\ if\ rand[0,1] \leq cr\ or\ j = j_{rand} \quad \textbf{(9)}$$

$$else\quad x_{ij}$$

Step 5: Compare the fitness value of the trial and target vector to select a vector having less fitness value in the next generation.

If fv (u_i) <fv (x_i)
then $x_i(t+1)=u_i(t+1)$
Else x_i (t+1) =x_i (t)

Step 6: Repeat steps 3, 4, 5 until some termination condition is reached, such as predefined number of iterations is reached.

Proposed Improved DEPSO

In PSO every single solution (i.e. particle) flies over the solution space with a velocity, which is adjusted at each time step in search for the optimal solution. The particle flies towards a position, which depends on its own past best position and the position of the best of its neighbours. Although PSO converges quickly, easily gets stuck in local optima because of loss of diversity of swarm. DE algorithm has some advantages, such as its ability to maintain the diversity of population, and to explore local search, but it has no mechanism to memory the previous process and use the global information about the search space, so it results in a waste of computing power and may get trapped in local optima. The differential information can be helpful for the search ability, but it also leads to instability of some solutions. So to take the advantage of the evolutionary operators of DE and memory information about the past best position and global best position used in PSO a simple hybrid version of DEPSO has proposed. Initial population set is the parent vector for both DE and PSO. For each individual a new position is obtained using Particle Swarm's velocity and position update equations and a mutant vector is created using mutation of DE. Then a trial vector is created applying crossover on PSO updated position and mutant vector. Comparing the fitness values of trial vector with the original vector, a candidate solution set is obtained either from parent or trial vector and accordingly the *pbest* and *gbest* position of the particles are updated. The method is repeated iteratively till the optimum value is reached.

Although the algorithm takes the advantage of both DE and PSO still its performance is dependent on parameter setting. Inertia weight and differential factor are the two important parameters of DEPSO that affects the convergence rate by providing a tradeoff between global and local search in the solution space. Larger inertia weight facilitates global exploration while lower values encourage a local search. Similarly at the beginning of the evolutionary process, a large differential factor *F* is required to make sure the algorithm has a strong global search capability while at the later stage, a small *F* is needed for a better local search. Hence it has been proposed to adapt different values for the differential factor and inertia weight in different iterations without keeping it fixed.

Steps of Improved DEPSO based learning is:

Step 1: Initialize the position of each particle according to the population size.

Step 2: Find the fitness function value of each particle.

Step 3: Compare particle's fitness evaluation with particle's lbest. If current value is better than lbest, then set lbest value equal to the current value and the lbest location equal to the current location in d-dimensional space.

Step 4: Compare fitness evaluation with the population's overall previous best. If the current value is better than gbest, then reset gbest to the current particle's value.

Step 5: Find a new velocity and position pp_i of the particle x_i according to the equations (1) and (2), respectively:

Step 6: Create a mutated individual v_i for each individual target vector x_i using a suitable mutation equation.

Step 7: Create a new vector called trial by mixing the parameters of the mutant vector with those of the target vector by using the following crossover operation:

$$\begin{aligned} u_{ij} &= v_{ij} \ \text{if} \ rand[0,1] \le cr \ or \ j = j_{rand} \\ & else \\ &= pp_{ij} \end{aligned} \tag{10}$$

Step 8: Compare the fitness value of the trial and target vector to select a vector having less fitness value in the next generation.

```
    If fv ($u_i$) <fv ($x_i$)
    then $x_i(t+1)=u_i(t+1)$
    Else $x_i$ (t+1) =$x_i$ (t)
    Step 9:   update the differential factor as follows
    f= f - (it/itmax) * (uf-lf)
    Step 10:   update the inertia weight as follows
    w= w - (it/itmax) * (uf-lf)
```

Step 11: Repeat steps 3 to 10 until some termination condition is reached, such as predefined number of iterations (itmax) is reached.

Experimental Result Analysis

In order to verify the validity of improved algorithms, a test suit consisting of four unconstrained benchmark functions is applied to conduct the experiments. Table 1 lists the names, types, dimensionalities, ranges of each variable of the optimization problems. All these functions are minimization problems with minimum value zero.

The proposed algorithm has been compared with traditional DEPSO, DE, PSO method. Initially the performance of DE with different mutation strategies has been compared and the best one has used in improved DEPSO algorithm. Each algorithm runs for 100 generations with population size 20 for three different dimensions of the functions. The crossover probability, cognitive and social acceleration constants are fixed with 0.9, 1.9 and 1.9 respectively for the experiment. At the beginning of the evolutionary process, a large mutation factor *F* i.e. 0.6 has chosen for accelerating the global search capability while at the later stage, a small *F* i.e. 0.3 has chosen for a better local search. Consequently, a dynamic adjustment method for differential factor *F* has been considered. Similarly the inertia weight *w* has been linearly decreased from 0.9 to 0.4. The performance comparison of DE with different mutation strategy in terms of the mean and best fitness value has shown in table 2 and the corresponding outputs has shown in figure 1 to 12. In the figures DE/Rand1, DE/Rand2, DE/Best1, DE/Best2, DE/Current to Best DE/(Centroid mutation) are labeled as De, De4, DE2, DE3, DE1 and DE5 respectively. In most of the cases DE/Best2 provides best result compare to other mutation strategies. Hence it has used in the adaptive DEPSO algorithm. The performance comparison of Adaptive DEPSO (labeled as DEPSO1), traditional DEPSO, DE and PSO in terms of the mean and best fitness value has shown in table 3 and the corresponding outputs has shown in figure 13 to 24.

Conclusion

In this paper an improved DEPSO algorithm integrating the advantages of both DE and PSO with adaptive parameter tuning has been presented for solving various global optimization problems. Different optimization problems require different mutation strategies with different parameter values depending on the nature of problem and available computation resources. So initially the performance of DEPSO has compared with six different mutation strategies to adapt the best one. Again instead of using fixed parameter values, a dynamic adjustment method for the parameters like differential factor and inertia weight to accelerate the convergence rate of DEPSO has been proposed with the best mutation strategy. Evaluating the performance of improved DEPSO on a set of benchmark functions, more accurate results are obtained compared to DE, PSO and traditional DEPSO with faster convergence rate.

References

[1] Vesterstorm Jakob, Thomsen Rene (2004), " A comparative Study of Differential Evolution, Particle Swarm Optimization and Evolutionary Algorithms on Numerical Benchmark Problems", Congress on Evolutionary Computation, 2, PP: 1980-1987.

[2] Thiemo Krink, Sandra Paterlini (2006), "Differential Evolution and Particle Swarm Optimization in Partitional Clustering", Computational Statistics and Data Analysis vol. 50 (5), PP: 1220-1247.

[3] Zhi-Feng Hao, Guang-Han Guo (2007), "A Particle Swarm Optimization Algorithm with Differential Evolution", Proceedings of Sixth International Conference on Machine Learning and Cybernetics, PP: 1031-1035.

[4] Jixiang Cheng, Gexiang Zhang (2009), "Improved Differential Evolutions Using a Dynamic Differential Factor and Population Diversity", International conference on Artificial Intelligence and Computational Intelligence, PP: 402-406.

[5] Jen-ing g. Hwang, Chia-jung Huang (2010), "Evolutionary Dynamic Swarm Optimization for Data Clustering", Ninth International Conference on Machine Learning and Cybernetics, PP: 3240-3245.

[6] Wen-Jun Zhang, Xiao-Feng Xie (2003), "DEPSO: Hybrid Particle Swarm with Differential Evolution Operator", IEEE International conference on Systems, Man and Cybernetics, PP: 3816-3821.

[7] Rui Xua, Ganesh K. Venayagamoorthyb, Donald C. Wunsch (2007), "Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization", Neural Networks, vol.20 PP: 917–927.

[8] K. Gnanambal, C.K. Babulal (2012), "Maximum loadability limit of power system using hybrid differential evolution with particle swarm optimization", Electrical Power and Energy Systems vol. 43, PP: 150–155.

Table 1 Details of Benchmark functions used for experiment

Function Name Function Type	Range	Dim
Sphere $f(x) = \sum (x_i)^2$ **Unimodal**	**[-5.12, 5.12]**	**[5,10, 20]**
Rosenbrock $f_2(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ **Multimodal**	**[-2.048, 2.048]**	**[5,10, 20]**
Rastriging $f_3(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$ **Multimodal**	**[-5.12, 5.12]**	**[5,10, 20]**
Griewank $f_4(x) = 1/4000 \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$ **Multimodal**	**[-300, 300]**	**[5,10, 20]**

Table 2 Performance comparison of DE with different mutation strategy

Function	Dimension	DE	DE1	DE2	DE3	DE4	DE5
		Mean Min	Mean Min	Mean Min	Mean Min	Mean Min	Mean Min
Sphere	5	1.0882e-008 1.2143e-009	4.6754e -016 3.5101e-016	3.8918e-007 3.8918e-007	8.2552e-012 1.1849e-012	7.9503e-005 2.3076e-005	0.0529 0.0515
	10	0.0032 0.0022	0.1426 0.1426	3.3114 3.3114	7.3671e-004 5.6700e-004	0.3039 0.1464	2.3526 2.1226
	20	0.7931 0.5613	7.3695 7.3694	5.3274 5.3274	0.0582 0.0488	9.9241 7.0252	5.4648 5.4324
Rosenbrock	5	0.0601 0.0543	2.7273 2.7273	2.0905 2.0905	3.8279e-005 1.2166e-005	0.4836 0.4115	4.4151 3.4112
	10	7.1669 7.0618	8.8757 8.8757	21.0789 21.0789	6.5251 6.4874	37.47992 56.5950	22.7399 21.4393
	20	61.6185 57.5648	111.5156 111.5136	362.7766 362.7766	20.1524 20.0460	679.2695 374.1176	223.2574 221.2564
Rastringin	5	14.6634 5.8606	7.5695 1.6390	5.9709 5.9709	14.0046 7.3956	20.348 6.9166	1.0651 1.0441
	10	59.2737 37.7343	58.7590 40.5695	15.7275 15.7275	63.9586 44.0976	83.1306 85.5810	26.9391 24.9323
	20	160.0167 115.7832	129.5826 92.9804	72.2531 72.2531	154.3311 115.7274	240.2773 194.2356	80.6025 78.6215
Griewank	5	0.4359 0.2012	0.3515 0.0848	0.4229 0.4229	0.6174 0.4329	0.7054 0.3475	0.9087 0.9026
	10	0.9043 0.4951	0.3127 0.2252	4.4585 4.4585	0.8479 0.6158	1.2763 1.1212	2.4709 2.4312
	20	1.5005 1.3387	4.4722 4.4720	8.2468 8.2468	1.1217 1.0848	20.7550 13.4865	11.5962 9.5662

Table 3 Performance comparison of DE with different mutation strategy

Function	Dimension	ADAPTIVE DEPSO	DEPSO	DE3	PSO
		Mean Min	Mean Min	Mean Min	Mean Min
SPHERE	5	1.0033e-029 5.2160e-032	7.0217e-026 4.2129e-029	2.8572e-013 5.1864e-014	.0053 .0033
	10	1.4867e-015 6.3248e-020	3.9866e-016 1.9912e-019	6.6435e-007 4.6741e-007	0.5846 0.3816
	20	8.0499e-006 1.4192e-009	9.1221e-005 2.2588e-008	1.2421 1.1883	7.3313 6.2313
ROSENBROCK	5	1.5985e-028 0	1.7557e-023 1.0615e-025	0.0325 0.0186	2.1837 2.0037
	10	1.8000e-015 1.1614e-019	6.1322e-012 5.7857e-015	3.3686 3.3398	9.6963 8.6963
	20	0.0039 4.6278e-006	0.1086 5.4861e-005	67.2985 66.4350	173.4763 170.4563
RASRINGIN	5	0 0	0 0	11.6974 4.0160	3.3916 3.1416
	10	2.6645e-015 0	1.9540e-14 0	60.1054 41.0296	30.6940 29.6344
	20	7.1932e-005 1.0407e-007	2.5815e-004 7.0160e-007	161.3604 137.4773	55.4377 52.4107
GRIEWANK	5	0 0	0 0	0.6254 0.4644	0.0492 0.0472
	10	1.5377e-015 0	1.9531e-011 5.4956e-014	0.9121 0.6731	0.7738 0.7438
	20	5.1159e-006 6.1544e-010	6.4799e-004 4.2116e-010	1.1259 1.1009	2.1015 2.0995

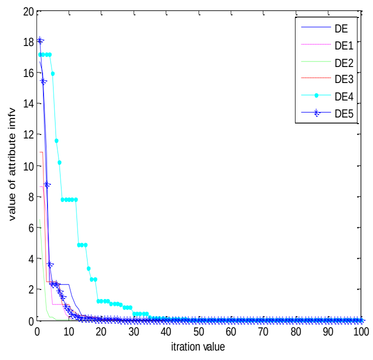

Fig1 Sphere with dimension 5

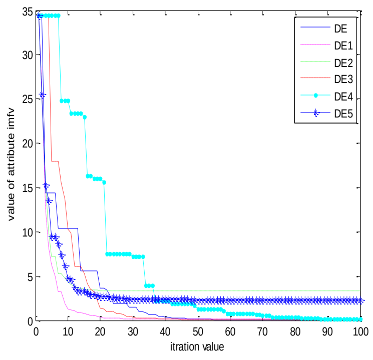

Fig 2 Sphere with dimension 10

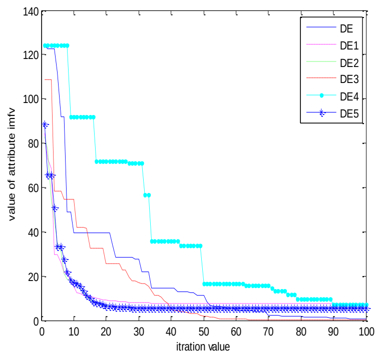

Fig 3 Sphere with dimension 20

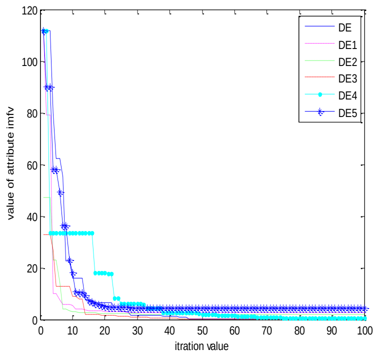

Fig4 Rosenbrock with dimension 5

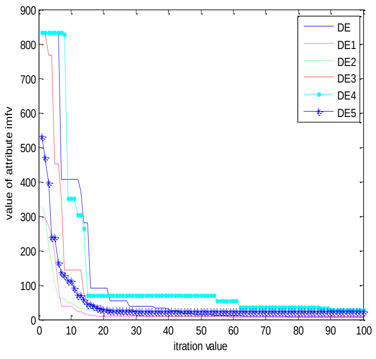

Fig 5 Rosenbrock with dimension 10

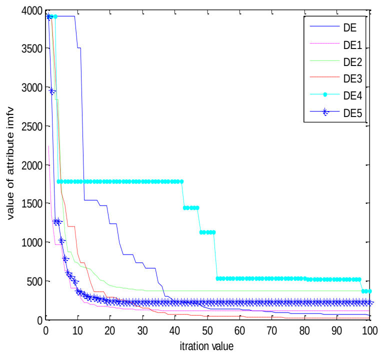

Fig 6 Rosenbrock with dimension 20

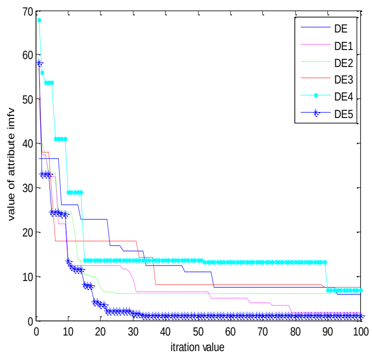

Fig 7 Rastrigin with dimension 5

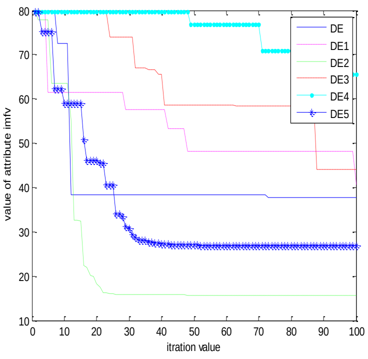

Fig 8 Rastrigin with dimension 10

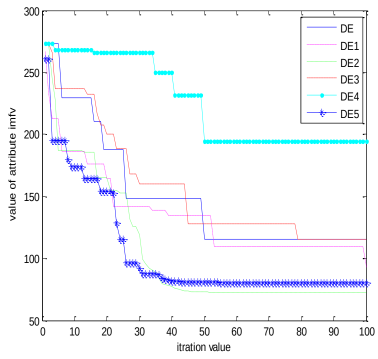

Fig 9 Rastrigin with dimension 20

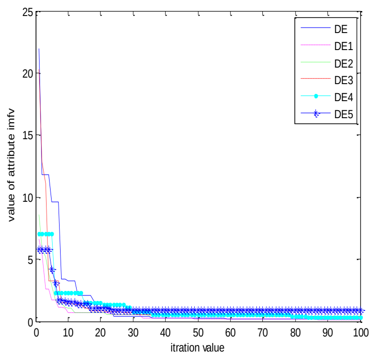

Fig 10 Griewank with dimension 5

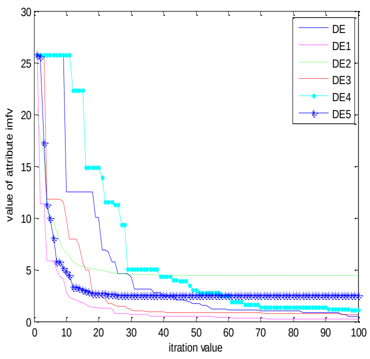

Fig 11 Griewank with dimension 10

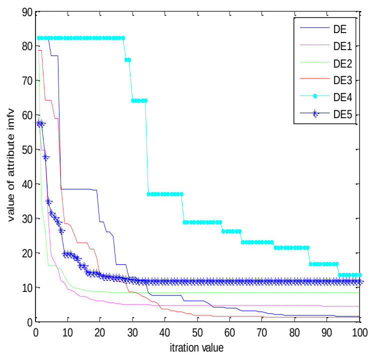

Fig 12 Griewank with dimension 20

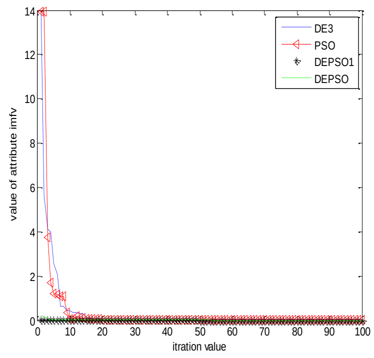

Fig 13 Sphere with dimension 5

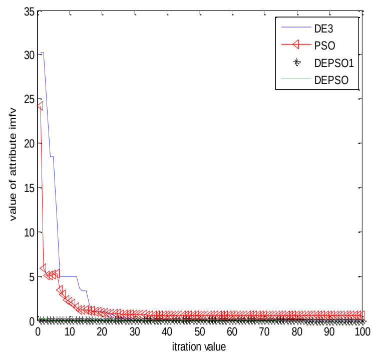

Fig 14 Sphere with dimension 10

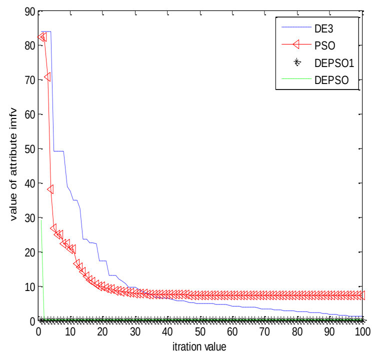

Fig 15 Sphere with dimension 20

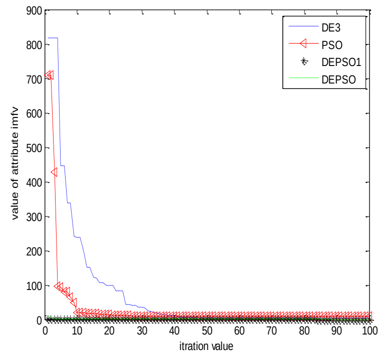

Fig 16 Rosenbrock with dimension 5

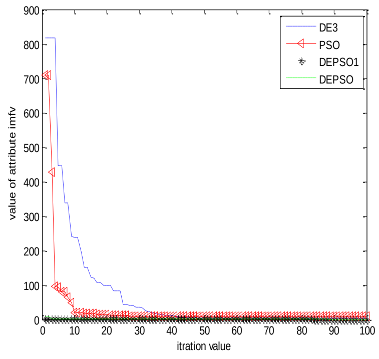

Fig 17 Rosenbrock with dimension 10

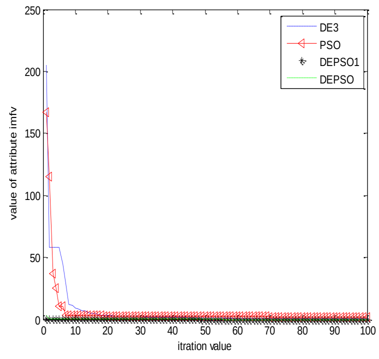

Fig 18 Rosenbrock with dimension 20

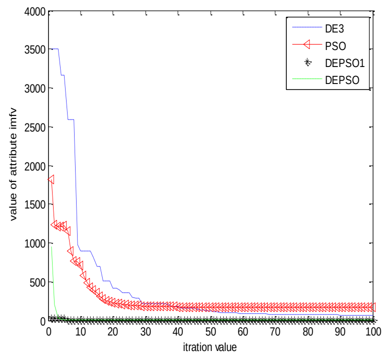

Fig 19Rastrigin with dimension 5

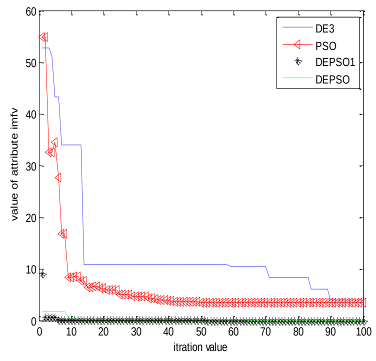

Fig 20 Rastrigin with dimension 10

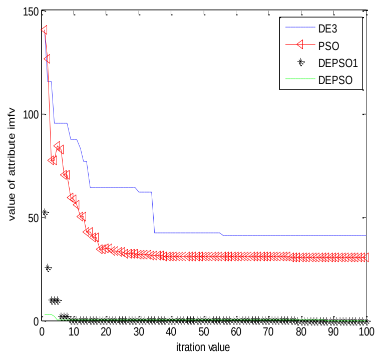

Fig 21 Rastrigin with dimension 20

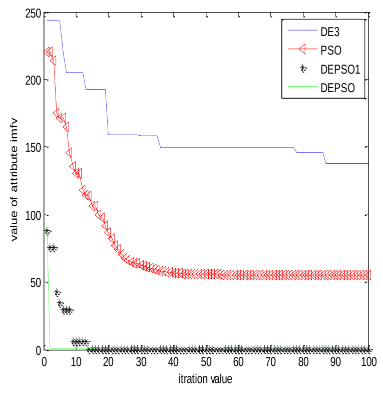

Fig 22 Griewank with dimension 5

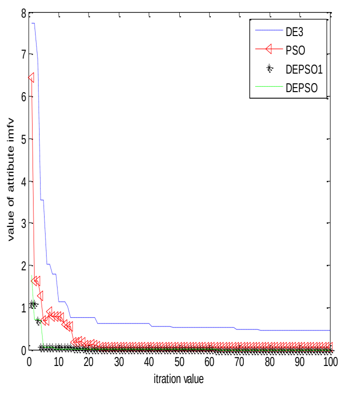

Fig 22 Griewank with dimension 10

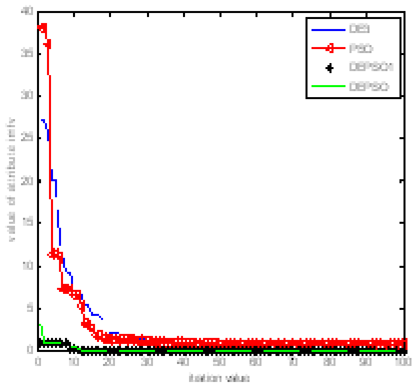

Fig 23 Griewank with dimension 20

[Performance comparison of Improved DEPSO with DEPSO, DE, PSO]

[9] Ling He, Dejia Shi, Li Wang (2009), “An Adaptive Discrete Particle Swarm Optimization for TSP Problem”, Second Asia-Pacific Conference on Computational Intelligence and Industrial Applications, PP: 393-396.

[10] R. Mallipeddi, P.N. Suganthana,, Q.K. Pan, M.F. Tasgetiren (2011), “Differential evolution algorithm with ensemble of parameters and mutation strategies”, Applied Soft Computing, vol. 11, PP: 1679-1696