Two Improved Differential Evolution Schemes for Faster Global Search

Swagatam Das
Electronics & Telecom Eng Dept.
Jadavpur University
Kolkata 700032, India
+(91) (33) 2528-2717
swagatamdas19@yahoo.co.in

Amit Konar
Electronics & Telecom Eng Dept.
Jadavpur University
Kolkata 700032, India
+(91) (33) 2416-2697
babu25@hotmail.com

Uday K. Chakraborty
Math & Computer Science Dept.
University of Missouri
St. Louis, MO 63121, USA
+1 (314) 516-6339
uday@cs.umsl.edu

ABSTRACT

Differential evolution (DE) is well known as a simple and efficient scheme for global optimization over continuous spaces. In this paper we present two new, improved variants of DE. Performance comparisons of the two proposed methods are provided against (a) the original DE, (b) the canonical particle swarm optimization (PSO), and (c) two PSO-variants. The new DE-variants are shown to be statistically significantly better on a seven-function test bed for the following performance measures: solution quality, time to find the solution, frequency of finding the solution, and scalability.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search -- *Heuristic methods*; G.1.6 [**Numerical Analysis**]: Optimization -- *Global optimization*; G.3 [**Probability and Statistics**] --- *Probabilistic algorithms*

General Terms

Algorithms

Keywords

Differential evolution, particle swarm optimization, evolutionary computation

1. INTRODUCTION

Differential evolution (DE) [10] seeks to replace the classical crossover and mutation schemes of the genetic algorithm (GA) by alternative differential operators. The DE algorithm has recently become quite popular in the machine intelligence and cybernetics community, and has, in many cases, been shown to be better than the GA or particle swarm optimization (PSO).

However, DE does suffer from the problem of premature convergence to local optima. Also, like most other stochastic optimization techniques DE is not free from the so-called "curse of dimensionality". In the present paper we propose two modifications to Storn and Price's DE ("scheme DE1") [10]. First, we introduce the concept of time-varying scale factor by which the difference vector is to be multiplied. Second, the scale factor is made to vary in a random way. The primary consideration behind these modifications is to try to avoid (or slow down) premature convergence in the early stages of the search and to facilitate convergence to the global optimum solution during the later stages of the search. Empirical simulations with well-known benchmarks show the usefulness of these modifications.

GECCO'05, June 25-29, 2005, Washington, DC, USA.

The remainder of this paper is organized as follows. In Section 2 we provide a brief outline of differential evolution. Section 3 introduces the two modifications to the basic scheme. Experimental settings for the benchmarks and the simulation strategies are explained in Section 4. Results are presented in Section 5, before concluding in Section 6.

2. THE CLASSICAL DE

DE searches for a global optimum point in an N-dimensional hyperspace. It begins with a randomly initialized population of N-dimensional real-valued parameter vectors. Each vector forms a candidate solution to the multi-dimensional optimization problem. Unlike the conventional GA, the reproduction phase in DE is implemented as follows. For each individual vector G_k^D belonging to generation D, randomly sample three other individuals G_i^D, G_j^D and G_m^D from the same generation (for distinct *i, j, k* and *m*), calculate the difference of the components of G_i^D and G_j^d, scale it by a scalar R (ϵ [0, 1]) and create a trial vector by adding the result to the chromosomes of G_k^D. For the n-th component of each parameter vector

$$\left.\begin{aligned} G_{k,n}^{D+1} &= G_{m,n}^{D} + R.(G_{i,n}^{D} - G_{j,n}^{D}) && \text{if } \mathrm{rand}_n(0,1) < \mathrm{CR} \\ &= G_{k,n}^{D}, && \text{otherwise.} \end{aligned}\right\} \quad (1)$$

where CR (ϵ [0, 1]) is the crossover constant. Parameters R and CR govern the convergence speed and robustness of DE. The algorithm is outlined below:

Procedure Differential-evolution

Begin

Initialize population;

Evaluate fitness;

While termination condition not satisfied

For j = 0 to population-size **do**

Create Difference-Offspring;

Evaluate fitness;

If an offspring is better than its parent

Then replace the parent by offspring in the next generation;

End If

End For

End While

End.

3. PROPOSED IMPROVEMENTS

Generally in population-based search and optimization methods, considerably high diversity is necessary during the early part of the search to utilize the full range of the search space. On the other hand during the latter part of the search, when the algorithm is converging to the optimal solution, fine-tuning is important to find the global optimum efficiently. Considering these issues, we propose two new strategies to improve the performance of the DE.

3.1 DE with Random Scale Factor (DERSF)

In the original DE [10] the difference vector (G_i - G_j) is scaled by a constant factor 'R'. The usual choice for this control parameter is a number between 0.4 and 1. We propose to vary this scale factor in a random manner in the range (0.5, 1) by using the relation

$$R = 0.5*(1+ \text{rand}(0, 1)) \quad (2)$$

where rand (0, 1) is a uniformly distributed random number within the range [0, 1].

The mean value of the scale factor is 0.75. This allows for stochastic variations in the amplification of the difference vector and thus helps retain population diversity as the search progresses. Even when the tips of most of the population vectors point to locations clustered near a local optimum due to the randomly scaled difference vector, a new trial vector has fair chances of pointing at an even better location on the multimodal functional surface. Therefore the fitness of the best vector in a population is much less likely to get stagnant until a truly global optimum is reached.

3.2 DE with Time Varying Scale Factor (DETVSF)

In most population-based optimization methods (except perhaps some hybrid global-local methods) it is generally believed to be a good idea to encourage the individuals (here, the tips of the trial vectors) to sample diverse zones of the search space during the early stages of the search. During the later stages it is important to adjust the movements of trial solutions finely so that they can explore the interior of a relatively small space in which the suspected global optimum lies. To meet this objective we reduce the value of the scale factor linearly with time from a (predetermined) maximum to a (predetermined) minimum value:

$$R = (R_{max} - R_{min}) * (MAXIT - iter) / MAXIT \quad (3)$$

where R_{max} and R_{min} are the maximum and minimum values of scale factor F, iter is the current iteration number and MAXIT is the maximum number of allowable iterations. The locus of the tip of the best vector in the population under this scheme may be illustrated as in Fig. 1.

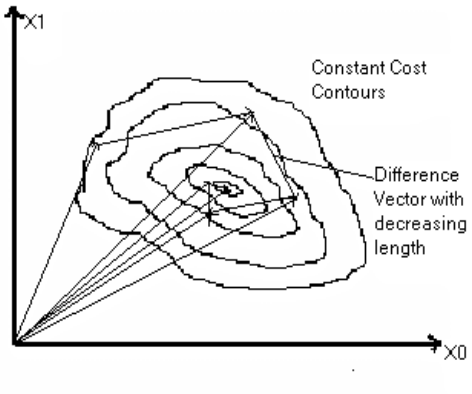


Figure 1. Illustrating the concept of time-varying scale factor in a 2-D parameter space.

4. EXPERIMENTAL SETTINGS

4.1 Benchmarks

We have used seven well-known benchmarks (Table 1) [12] to evaluate the performance of the new variants of the DE. In Table 1, *n* represents the number of dimensions (we used *n* = 5 to 30, in steps of 5). Here the proposed algorithms have been tested against (a) the original DE, (b) the canonical PSO, and (c) two other versions of PSO. The first two test functions are unimodal, having only one minimum. The others are multimodal, with a considerable number of local minima in the region of interest. All benchmark functions except f_7 have the global minimum at or very close to the origin [9], [12]. For Shekel's foxholes (f_7), the global minimum is at (-31.95, -31.95) and f_7 (-31.95, -31.95) ≈ 0.998.

Table 1. Benchmark functions for simulation

Function	Mathematical Representation
Sphere function	$f_1(x) = \sum_{i=1}^{n} x_i^2$
Rosenbrock function	$f_2(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
Rastrigin function	$f_3(x) = \sum_{i=1}^{n} [x_i^2 - 10 \cos(2\pi x_i) + 10]$
Griewank function	$f_4(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
Ackley Function	$f_5(x) = -20\exp(-0.2\sqrt{(\frac{1}{n}\sum_{i=1}^{n} x_i^2)} - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i) + 20 + e$
Schaffer's f_6 function	$f_6(x) = 0.5 + \frac{(\sin\sqrt{x^2 + y^2})^2 - 0.5}{(1.0 + 0.001(x^2 + y^2))^2}$
Shekel's Foxholes function	$f_7(x) = [\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}]^{-1}$

4.2 PSO Methods Compared

In PSO [5], a population of particles is initialized with random positions X_i and velocities V_i, and a function, f, is evaluated, using the particle's positional coordinates as input values. In an n-dimensional search space, $X_i = (x_{i1}, x_{i2}, x_{i3},...,x_{in})$ and $V_i = (v_{i1}, v_{i2}, v_{i3},...,v_{in})$. Positions and velocities are adjusted, and the function is evaluated with the new coordinates at each time-step. The basic update equations for the d-th dimension of the i-th particle in PSO may be given as

$$\left.\begin{aligned} V_{id}(t+1) &= \omega.V_{id}(t) + C_1.\varphi_1.(P_{lid} - X_{id}(t)) + C_2.\varphi_2.(P_{gd} - X_{id}(t)) \\ X_{id}(t+1) &= X_{id}(t) + V_{id}(t+1) \end{aligned}\right\} \quad (4)$$

The variables φ_1 and φ_2 are random positive numbers, drawn from a uniform distribution and defined by an upper limit φ_{max} which is a parameter of the system. C_1 and C_2 are called acceleration constants whereas ω is called inertia weight. P_{li} is the local best solution found so far by the i-th particle, while P_g represents the positional coordinates of the fittest particle found so far in the entire community.

PSO-TVIW. Shi and Eberhart [9] improved the performance of the PSO method by using a linearly varying inertia weight (ω) over iterations from a predefined maximum to a minimum value. They empirically observed that the performance could be improved by varying ω from 0.9 at the beginning of the search to 0.4 at the end of the search for most problems. We use these values while implementing this scheme. We call this version PSO-TVIW (PSO with Time Varying Inertia Weight).

PSO-RANDIW. Eberhart and Shi [3] proposed a random inertia weight for tracking dynamic systems. However, when the random inertia weight factor is used, the acceleration coefficients are kept constant at 1.494. In the rest of this paper this method is referred to as PSO-RANDIW.

4.3 Population Initialization

Since most of the test functions used in this paper have their global minimum at or near the origin of the search space, we use the asymmetric initialization method proposed by Angeline [1].

Under this scheme the population is initialized only in a certain portion of the search space. The most common dynamic ranges used in the literature are used in this paper and each dimension is confined to the same dynamic range [3], [2]. Table 2 shows the range of initialization and the range of search for each function.

Table 2. Initialization and range of search for test functions

Function	Range of search	Range of Initialization
f_1	$(-100, 100)^n$	$(50, 100)^n$
f_2	$(-100, 100)^n$	$(15, 30)^n$
f_3	$(-10, 10)^n$	$(2.56, 5.12)^n$
f_4	$(-600, 600)^n$	$(300, 600)^n$
f_5	$(-32, 32)^n$	$(15, 32)^n$
f_6	$(-100, 100)^2$	$(15, 30)^2$
f_7	$(-65.536, 65.536)^2$	$(0, 65.536)^2$

4.4 Simulation Strategy

Simulations were carried out to obtain a comparative performance analysis of the proposed methods with respect to: (a) canonical PSO (b) PSO-TVIW (c) PSO-RANDIW and (d) classical DE. Thus a total of six algorithms were considered – two new, the other four existing in the literature. All benchmarks except Schaffer's f_6 and Shekel's Foxholes were tested with dimensions 5 through 30 (in steps of 5). Schaffer and Shekel are two dimensional. Fifty independent runs of each of the six algorithms were executed, and the average and the standard deviation of the best-of-run values were recorded. Different maximum generations (G_{max}) were used according to the complexity of the problem. For all benchmarks (excluding Schaffer's f_6 and Shekel's f_7) the stopping criterion was set as reaching a fitness of 0.001. For Schaffer's f_6, the widely used error limit of 0.00001 [6], [11] was used. For Shekel's Foxholes the criterion was 0.998.

In the case of DE and its new extensions, we chose CR = 0.9. For the original DE, scale factor R = 0.8 was used. After some trial runs we found that varying the scale factor linearly from 1.2 to 0.4 gave good results for DE-TVSF. For PSO and its variants it is quite common to limit the value of each component of the velocity vector of each particle (V_{id}) to the maximum allowable value. Through empirical studies on numerical benchmarks, Eberhart and Shi [4] suggested that it is good to limit the maximum velocity V_{max} to the upper limit of the dynamic range of search, i.e., X_{max}. Therefore we too use this limit in this investigation.

4.5 Population Size

It is common practice in DE research to use a population size ten times the dimensionality of the search space. DE maintains a constant population size across generations because each population member is either replaced by a trial vector (if the trial solution is fitter than the original one) or left unaltered at each generation. However, it is quite common in the PSO literature to limit the number of particles to limit the number of particles to the range 20 to 60 [4], [7] and [8]. Van den Bergh and Engelbrecht [11] have shown that though there is a slight improvement of the optimal value with increasing swarm size, a larger swarm increases the number of function evaluations to converge to an error limit. Eberhart and Shi [4] showed that the population size has hardly any effect on the performance of the PSO method. In this paper empirical experiments for all the six algorithms have been carried out with the same population sizes (100, 200, and 300).

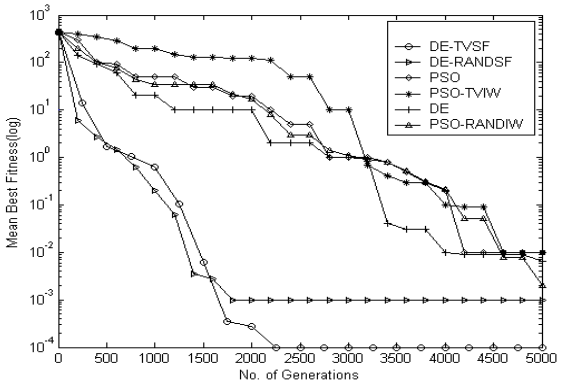
5. RESULTS

The following performance measures are used for our comparative study: (a) quality of the final solution (b) speed of convergence towards the optimal solution, (c) success rate (frequency of hitting the optimum), and (d) scalability of the algorithms against the growth of problem dimensions. Table 3 compares the algorithms on the quality of the optimum solution. The mean and the standard deviation (within parentheses) of the best-of-run values for 50 independent runs of each of the six algorithms are presented in Table 3. Missing values of standard deviation in this table indicate a zero standard deviation. The best solution in each case has been shown in bold. Table 4 shows results of unpaired *t*-tests between the better of the two new algorithms and the best of the other four in each case (standard error of difference of the two means, 95% confidence interval of

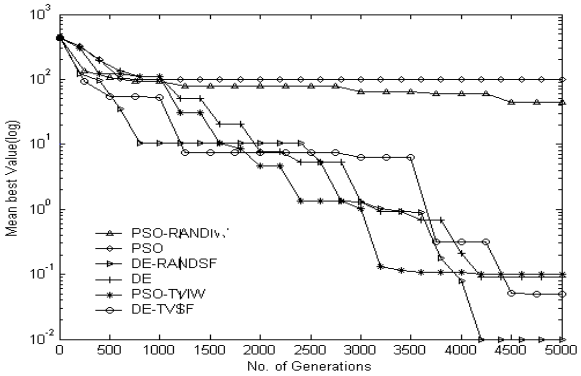
this difference, the *t* value, and the two-tailed *P* value). For all cases in Table 4, sample size = 50 and degrees of freedom = 98. It is interesting to see from Tables 3 and 4 that in a majority of cases one of the two proposed methods meets or beats the nearest competitor in a statistically meaningful way. Table 3 shows that in two cases the proposed methods' means are numerically larger (i.e., worse) than the mean of the competitor, but as Table 4 shows, one of these two differences is not statistically significant. Table 5 shows, for the same set of runs as used in Tables 3 and 4, the number of runs (out of 50) that managed to find the optimum solution (within the given tolerance) and also the average number of generations (in parentheses) needed to find that solution. In Figure 2 we have graphically presented the rate of convergence of all the methods for all the functions (in 30 dimensions). Figure 3 shows the scalability of the six methods on the first five test functions -- how the average time to convergence varies with an increase in the dimensionality of the search space. These results show that the proposed methods lead to significant improvements in most cases.

Table 3. Average and the standard deviation of the best-of-run solution for 50 runs

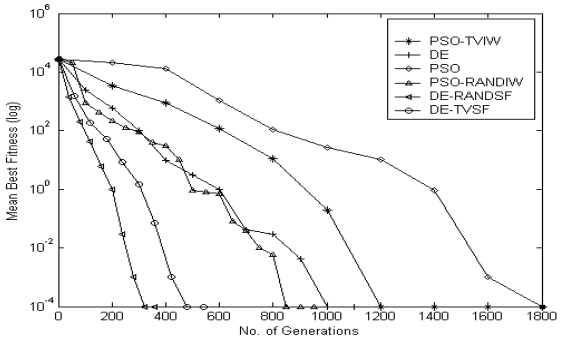
F	Dim	G_{max}	Average (Standard Deviation)					
			Canonical PSO	PSO-TVIW	PSO-RANDIW	DE	DE-TVSF	DE-RANDSF
f_1	10	1000	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
	20	2000	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
	30	4500	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
f_2	10	3000	23.535 (32.312)	20.25 (26.187)	3.254 (41.032)	2.051 (2.048)	0.0543 (0.0117)	**0.0049 (0.0121)**
	20	4000	54.21 (35.221)	32.33 (16.309)	5.322 (11.132)	1.089 (9.675)	**0.0654 (0.453)**	0.0913 (0.554)
	30	5000	78.90 (44.786)	60.561 (27.523)	12.122 (25.434)	0.834 (10.285)	0.486 (2.278)	**0.427 (1.582)**
f_3	10	3000	3.334 (2.218)	2.084 (1.443)	4.78 (2.493)	0.025 (0.0109)	0.06 (0.0045)	**0.0018 (0.0067)**
	20	4000	18.997 (6.489)	11.36 (4.128)	29.371 (8.313)	0.5367 (0.0361)	0.0021 (0.0231)	**0.0013 (0.033)**
	30	5000	10.457 (12.801)	24.536 (3.757)	8.435 (2.169)	0.043 (0.0914)	**0.0026 (0.112)**	0.0058 (0.277)
f_4	10	2500	0.1003 (0.077)	0.072 (0.041)	**0.0081 (0.037)**	0.052 (0.03)	0.021 (0.0187)	0.024 (0.180)
	20	3500	0.043 (0.0332)	0.02612 (0.0334)	0.0311 (0.017)	0.025 (0.0033)	**0.0012 (0.031)**	0.016 (0.0223)
	30	5000	0.0198 (0.166)	0.0186 (0.023)	0.0148 (0.029)	0.019 (0.0035)	0.0026 (0.0035)	**0.0019 (0.0022)**
f_5	10	2000	0.341 (3.435)	0.538 (0.9812)	0.143 (0.669)	0.0346 (0.0446)	**0.0015 (0.0154)**	0.0127 (0.0454)
	20	3500	0.457 (2.934)	0.624 (1.658)	0.677 (0.5241)	0.057 (0.0075)	0.0543 (0.0082)	**0.008 (0.0618)**
	30	5000	1.454 (0.224)	1.002 (0.5351)	0.0659 (0.768)	0.731 (0.0572)	0.0445 (0.069)	**0.0011 (0.088)**
f_6	2	1000	0.0045 (0.2322)	0.0042 (0.015)	0.0013 (0.0018)	**0.00006 (0.0011)**	0.00045 (0.016)	0.00013 (0.01225)
f_7	2	1000	1.551 (2.645)	1.934 (1.432)	1.772 (2.327)	1.002 (0.0854)	1.221 (0.0421)	**0.998 (0.000)**



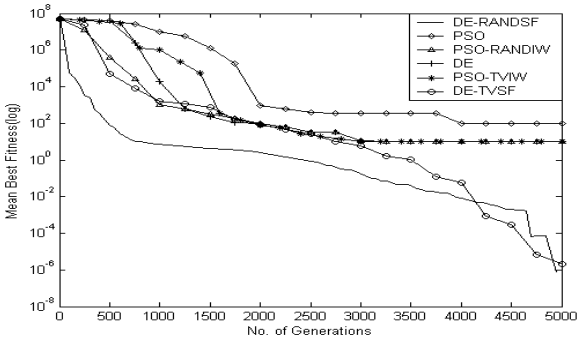
(a) Sphere Function (f_1)



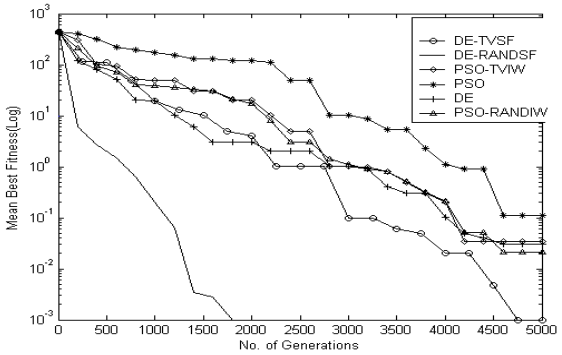
(b) Rosenbrock Function (f_2)



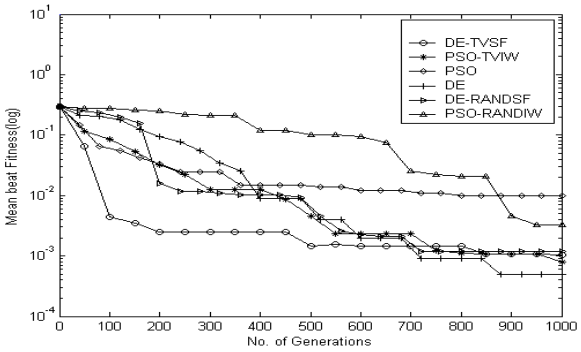
(c) Griewank Function (f_3)



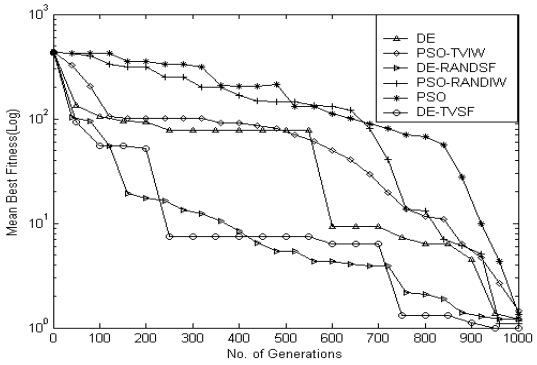
(d) Rastrigin Function (f_4)



(e) Ackley Function (f_5)

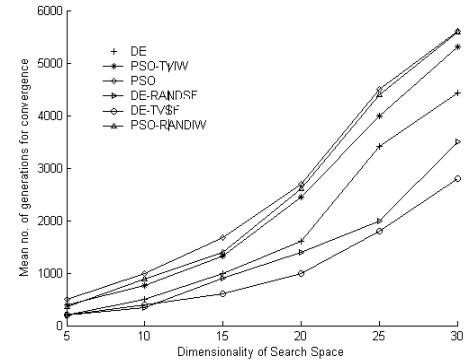


(f) Schaffer's f_6 Function (f_6)

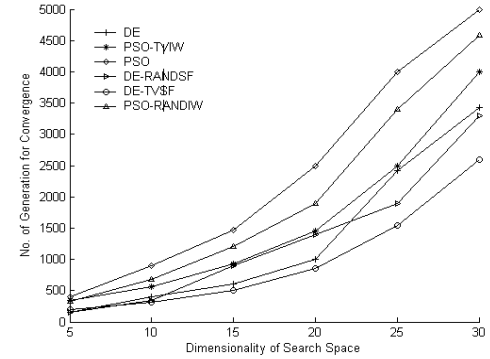


(g) Shekel's Foxholes Function (f_7)

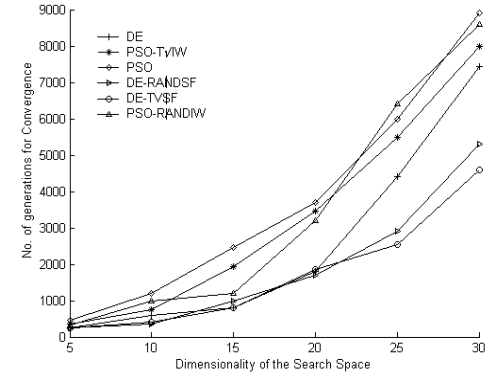
Figure 2. Progress to the optimum solution (all plots are for dimension = 30, except f_6 and f_7 which are 2D)



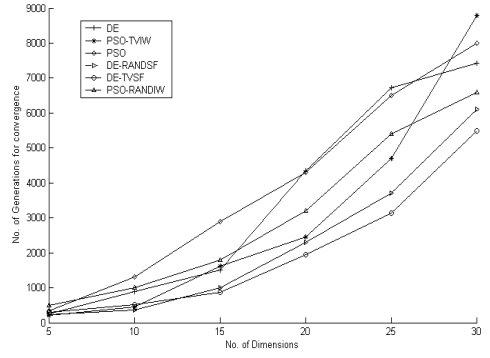
(a) Sphere Function (f_1)



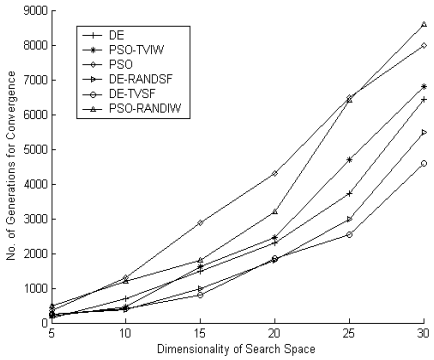
(b) Rosenbrock Function ((f_2)



(c) Griewank Function (f_3)



(d) Rastrigin Function (f_4)



(e) Ackley Function (f_5)

Figure 3. Variation of mean convergence time with increase in dimensionality of the search space.

Table 4. Results of unpaired t-tests on the data of Table 3

Fn, Dim	Std. Err	*t*	95% Conf. Intvl	Two-tailed *P*	Significance
f_2, 10	0.290	7.0644	(-2.621, -1.471)	< 0.0001	Extremely significant
f_2, 20	1.370	0.7473	(-3.742, 1.695)	0.4567	Not significant
f_2, 30	1.472	0.2766	(-3.327, 2.513)	0.7827	Not significant
f_3, 10	0.002	12.822	(-0.027, -0.020)	< 0.0001	Extremely Significant
f_3, 20	0.007	77.404	(-0.549, -0.522)	< 0.0001	Extremely Significant
f_3, 30	0.020	1.9761	(-0.081, 0.00018)	0.051	Not quite significant
f_4, 10	0.006	2.2003	(-0.025, -0.001)	0.0301	Significant
f_4, 20	0.004	5.3983	(-0.033, -0.015)	< 0.0001	Extremely significant
f_4, 30	0.004	3.1364	(-0.021, -0.005)	0.0023	Very significant
f_5, 10	0.007	4.9604	(-0.046, -0.020)	< 0.0001	Extremely significant
f_5, 20	0.009	5.5657	(-0.066, -0.032)	< 0.0001	Extremely significant
f_5, 30	0.109	0.5927	(-0.282, 0.152)	0.5547	Not significant
f_6, 2	0.002	0.0402	(-0.004, 0.003)	0.968	Not significant
f_7, 2	0.012	0.3312	(-0.028, 0.020)	0.7412	Not significant

Table 5. Number of runs (out of 50) that converged to optimality and the corresponding mean number of generations

Fn	Dim	G_{max}	No. of runs that converged to optimality (Average number of generations)					
			Canonical PSO	PSO-TVIW	PSO-RANDIW	DE	DE-TVSF	DE-RANDSF
f_1	10	1000	50 (678.4)	50 (520.9)	50 (197.4)	50 (189.7)	50 (248.3)	**50 (167.4)**
	20	2000	50 (2213.8)	50 (1453.67)	50 (776.5)	50 (743.7)	**50 (589.3)**	50 (1124.7)
	30	3000	50 (2133.5)	50 (2500.4)	50 (1852.5)	50 (2335.8)	**50 (1500.6)**	50 (1966.4)
f_2	10	3000	0	0	16 (2300.5)	18 (2257.5)	22 (2026.5)	**35 (2889.6)**
	20	4000	0	0	2 (3431)	7 (4323.6)	**16 (4067.3)**	10 (6213.7)
	30	5000	0	0	0	0	7 (4604.3)	**14 (4989.4)**
f_3	10	3000	2 (3190.5)	2 (3089)	7 (2566)	14 (3435.6)	46 (2298.2)	**48 (1675.3)**
	20	4000	0	0	0	5 (5647.9)	40 (4655.8)	**44 (3984.7)**
	30	5000	0	0	0	3 (6004.2)	**42 (5087.6)**	38 (4751.4)
f_4	10	2500	0	10 (2457.8)	**32 (2446.5)**	19 (2079.9)	12 (2066.5)	10 (2196.5)
	20	3500	10 (3386)	16 (3351)	19 (3409.7)	12 (3485.7)	**25 (3449)**	23 (2632)
	30	5000	26 (4357.5)	35 (4231.6)	12 (4254.8)	22 (3256.6)	40 (3043.2)	**46 (4582.5)**
f_5	10	2000	12 (1114.6)	10 (1974.3)	30 (1429.4)	43 (1999.4)	**45 (1231.5)**	42 (1209.6)
	20	3500	3 (3482.4)	4 (3184.5)	10 (2897.4)	24 (3461.9)	33 (3100.8)	**38 (3446.6)**
	30	5000	0	0	19 (4438)	30 (4871.4)	43 (4970)	**46 (4981.4)**
f_6	2	1000	19 (625.4)	12 (386.5)	19 (463.6)	**48 (344.8)**	46 (521.5)	43 (503.1)
f_7	2	1000	24 (549.5)	20 (892)	23 (783.3)	45 (657.8)	46 (556.4)	**50 (643.1)**

6. CONCLUSION

Two new variants of DE have been presented, based on the mechanism of controlling the scale factor. The new schemes have been shown to improve performance in a statistically meaningful way. The new DE-variants have been compared against (a) the basic DE, (b) the PSO, and (c) two PSO-variants using a seven-function test suite. The following performance metrics have been used: (a) solution quality, (b) speed of convergence, (c) frequency of hitting the optimum, and (d) scalability. The proposed variants have been shown to meet or beat the competitor in most cases.

7. ACKNOWLEDGMENTS

Partial support of UGC-sponsored projects on i) *AI and Expert Systems* and ii) *Excellence Program in Cognitive Science* is acknowledged. We are thankful to four anonymous referees for their valuable comments.

8. REFERENCES

[1] Angeline, P. J. Evolutionary optimization versus particle swarm optimization: Philosophy and the performance difference, *Lecture Notes in Computer Science (vol. 1447), Proceedings of 7th International Conference on. Evolutionary Programming* – Evolutionary Programming VII (1998) 84-89.

[2] Blackwell, T. A., Bentley, P. Improvised music with swarms, In *Proceedings of IEEE Congress on Evolutionary Computation 2002*, vol. 2, Honolulu, HI, (2002) 1462-1467.

[3] Eberhart, R. C., Shi, Y. Particle swarm optimization: Developments, applications and resources, In *Proceedings of IEEE International Conference on Evolutionary Computation*, vol. 1, (2001) 81-86.

[4] Eberhart, R. C., Shi, Y. Comparing inertia weights and constriction factors in particle swarm optimization, *Proceedings of IEEE International Congress on Evolutionary Computation*, vol. 1, (2000) 84-88.

[5] Kennedy, J., Eberhart, R. Particle swarm optimization, In *Proceedings of IEEE International Conference on Neural Networks*, (1995) 1942-1948.

[6] Kennedy, J. Stereotyping: Improving particle swarm performance with cluster analysis, *Proc. IEEE Int. Conf. Evolutionary Computation*, vol. 2, (2000) 303-308.

[7] Shi, Y., Eberhart, R. C. Comparison between genetic algorithm and particle swarm optimization, *Lecture Notes in Computer Science -- Evolutionary Programming VII*, vol. 1447, (1998) 611-616.

[8] Shi, Y., Eberhart, R. C. Parameter selection in particle swarm optimization, Lecture Notes in Computer Science Evolutionary Programming VII, vol. 1447, (1998) 591-600.

[9] Shi, Y., Eberhart, R. C. Empirical study of particle swarm optimization. In *Proceedings of IEEE International Conference on Evolutionary Computation,* vol 3, (1999) 101-106.

[10] Storn, R., Price, K. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization,* 11(4) (1997) 341–359.

[11] van den Bergh, F., Engelbrecht, P. A. Effects of swarm size on cooperative particle swarm optimizers, In *Proceedings of GECCO-2001*, San Francisco, CA, (2001) 892-899.

[12] Yao, X., Liu, Y., Lin, G. Evolutionary programming made faster, *IEEE Transactions on Evolutionary Computation*, vol 3, No 2, (1999) 82-102.