Improving Particle Swarm Optimization with Differentially Perturbed Velocity

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
This paper introduces a novel scheme of improving the performance of particle swarm optimization (PSO) by a vector differential operator borrowed from differential evolution (DE). Performance comparisons of the proposed method are provided against (a) the original DE, (b) the canonical PSO, and (c) three recent, high-performance PSO-variants. The new algorithm is shown to be statistically significantly better on a seven-function test suite 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 --- *Probabilistic algorithms*

General Terms
Algorithms

Keywords
Particle swarm optimization, differential evolution, evolutionary computation

1. INTRODUCTION
Particle swarm optimization (PSO) [8] is a stochastic optimization technique that draws inspiration from the behavior of a flock of birds or the collective intelligence of a group of social insects with limited individual capabilities. In this paper we present an improved PSO algorithm by proposing a new scheme of adjusting the velocities of the particles in PSO with a vector differential operator. The canonical PSO updates the velocity of a particle using three terms -- a previous velocity term that provides the particle with the necessary momentum, a *social* term that indicates how the particle is stochastically drawn towards the globally best position found so far by the entire swarm, and a *cognitive* term that reflects the personal thinking of the particle, i.e., how much it is drawn towards the best position so far encountered in its own course. In the proposed scheme the cognitive term is omitted; instead the particle velocities are perturbed by a new term containing the weighted difference of the position vectors of any two distinct particles randomly chosen from the swarm. A *survival of the fittest* mechanism has also been introduced in the swarm.

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

2. CANONICAL PSO AND SOME OF ITS SHORTCOMINGS
In PSO a population of particles is initialized with random positions X_i and velocities V_i, and a fitness 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 velocity and position update equations for the d-th dimension of the i-th particle in the swarm may be given as follows:

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

The variables φ_1 and φ_2 are random positive numbers, drawn from a uniform distribution, and with an upper limit φ_{max} which is a parameter of the system. C_1 and C_2 are called acceleration constants, and ω is the inertia weight. P_{li} is the best solution found so far by an individual particle, while P_g represents the fittest particle found so far in the entire community. The canonical PSO has been subjected to empirical [1], [7], [15] and theoretical [3], [13] investigations by several researchers. In many occasions the convergence is premature, especially if the swarm uses a small inertia weight ω [15] or constriction coefficient [3]. From equations (1), we see that if V_{id} is small, and if $|P_{lid}-X_{id}|$ and $|P_{gd}-X_{id}|$ too are small enough, V_{id} cannot attain a large value in the upcoming generations. That would mean a loss of exploration power. Such a case can occur even at an early stage of the search process, when the particle is the global best, causing both $|P_{lid}-X_{id}|$ and $|P_{gd}-X_{id}|$ to be zero, and V_{id} gets damped quickly with the ratio w. Also, the swarm suffers from loss of diversity in later generations if P_{lid} and P_{gd} are close enough [6], [16], [15].

3. PROPOSED ALGORITHM
In an attempt to circumvent the problems mentioned in the previous section, we have tightly coupled a differential operator (borrowed from differential evolution [12]) with the velocity-update scheme of PSO. The operator is invoked on the position vectors of two randomly chosen particles (population-members), not on their individual best positions. Further, unlike the PSO scheme, a particle is actually shifted to a new location only if the

new location yields a better fitness value, i.e., a selection strategy has been incorporated into the swarm dynamics.

In the proposed algorithm, for each particle i in the swarm two other distinct particles, say j and k ($i \neq j \neq k$), are selected randomly. The difference between their positional coordinates is taken as a difference vector **δ:**

$$\vec{\delta} = \overrightarrow{X_k} - \overrightarrow{X_j} \quad (2)$$

Then the d-th velocity component ($1 < d < n$) of the target particle i is updated as

$$\left.\begin{array}{ll} V_{id}(t+1) = \omega.V_{id}(t) + \beta.\delta_d + C_2.\,\varphi_2.\,(P_{gd}\text{-}X_{id}(t)), & \\ & \text{if } rand_d(0,1) < CR \\ = V_{id}(t), & \text{otherwise} \end{array}\right\} \quad (3)$$

where CR is the crossover probability, δ_d is the d-th component of the difference vector **δ** defined in (2), and β is a scale factor in [0, 1]. In essence the cognitive part of the velocity update formula in (1) is replaced with the vector differential operator to produce some additional exploration capability. Clearly, for CR < 1, some of the velocity components will retain their old values. Now, a new trial location Tr_i is created for the particle by adding the updated velocity to the previous position X_i:

$$\overrightarrow{Tr_i} = \overrightarrow{X_i}(t) + \overrightarrow{V_i}(t+1) \quad (4)$$

The particle is placed at this new location only if the coordinates of the location yield a better fitness value. Thus if we are seeking the minimum of an n-dimensional function $f(x_1, x_2, ..., x_n) = f(\vec{X})$, then the target particle is relocated as follows:

$$\left.\begin{array}{ll} \overrightarrow{X_i}(t+1) = \overrightarrow{Tr_i} & \text{if } (f(\overrightarrow{Tr_i}) < f(\overrightarrow{X_i}(t))) \\ \overrightarrow{X_i}(t+1) = \overrightarrow{X_i}(t) & \text{otherwise} \end{array}\right\} \quad (5)$$

Therefore, every time its velocity changes, the particle either moves to a better position in the search space or sticks to its previous location. The current location of the particle is thus the best location it has ever found. In other words, unlike the classical PSO, in the present scheme, P_{lid} always equals X_{id}. So the cognitive part involving $|P_{lid}\text{-}X_{id}|$ is automatically eliminated in our algorithm. If a particle gets stagnant at any point in the search space (i.e., if its location does not change for a predetermined number of iterations), then the particle is shifted by a random mutation (explained below) to a new location. This technique helps escape local minima and also keeps the swarm "moving":

$$\text{If}((X_i(t) = X_i(t+1)) = X_i(t+2) = ... = X_i(t+N))$$

$$\text{and}\,(f(\overrightarrow{X_i}(t+N)) \neq f^*))\ \text{then}$$

$$\text{for}\ (r = 1 \text{ to } n)\ X_{ir}(t+N+1) = X_{min} + rand_r(0,1)*(X_{max}\text{-}X_{min}) \quad (6)$$

where f* is the global minimum of the fitness function, N is the maximum number of iterations up to which stagnation can be tolerated and (X_{max}, X_{min}) define the permissible bounds of the search space. The pseudocode for this new method, called PSO-DV (Particle Swarm Optimization with Differentially perturbed Velocity), is presented below:

Procedure PSO-DV

begin
initialize population;
while stopping condition not satisfied do
for i = 1 to no_of_particles
evaluate fitness of particle;
update P_{gd};
select two other particles j and k ($i \neq j \neq k$) randomly;
construct the difference vector as $\vec{\delta} = \overrightarrow{X_k} - \overrightarrow{X_j}$;
for d = 1 to no_of_dimensions
if $rand_d(0, 1) < CR$
$V_{id}(t+1) = \omega.V_{id}(t) + \beta.\delta_d + C_2.\,\varphi_2.\,(P_{gd}\text{-}X_{id}(t))$;
else $V_{id}(t+1) = V_{id}(t)$;
endif
endfor
create trial location as $\overrightarrow{Tr_i} = \overrightarrow{X_i}(t) + \overrightarrow{V_i}(t+1)$;
if $(f(\overrightarrow{Tr_i}) < f(\overrightarrow{X_i}(t)))$ **then** $\overrightarrow{X_i}(t+1) = \overrightarrow{Tr_i}$
else $\overrightarrow{X_i}(t+1) = \overrightarrow{X_i}(t)$;
endif
endfor
for i = 1 to no_of_particles
if X_i stagnates for N successive generations
for r = 1 to no_of_dimensions
$X_{ir}(t+1) = X_{min} + rand_r(0, 1)*(X_{max}\text{-}X_{min})$
end for
end if
end for
end while
end

4. EXPERIMENTAL SETTINGS

4.1 Benchmarks

We have used seven well-known benchmarks [17] to evaluate the performance of the proposed algorithm. Here the proposed algorithm has been tested against the original DE, canonical PSO and three other recent variants of PSO. The benchmarks used are presented below (*n* represents the number of dimensions; we used up to 30 dimensions for the first five functions):

Sphere :

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \qquad \text{with} \quad -100 \le x_i \le 100$$

Rosenbrock :

$$f_2(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

$$\text{with} \quad -50 \le x_i \le 50$$

Rastrigin :

$$f_3(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$$

$$\text{with} \quad -5.12 \le x_i \le 5.12$$

Griewank :

$$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$\text{with} \quad -600 \le x_i \le 600$$

Ackley :

$$f_5(x) = -20\exp(-0.2\sqrt{(\frac{1}{n}\sum_{i=1}^{n} x_i^2)} - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i) + 20 + e$$

with $-32 \le x_i \le 32$

Schaffer's f_6:

$$f_6(x) = 0.5 + \frac{(\sin\sqrt{x^2 + y^2})^2 - 0.5}{(1.0 + 0.001(x^2 + y^2))^2}$$

with $-100 \le x_1, x_2 \le 100$

Shekel's Foxholes:

$$f_7(x) = [\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}]^{-1}$$

with $-65.536 \le x_1, x_2 \le 65.536$

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 the f_7 function have the global minimum at the origin or very near to the origin [11]. For Shekel's foxholes (f_7), the global minimum is at (-31.95, -31.95), with f_7(-31.95, -31.95) ≈ 0.998.

4.2 PSO Variants Used for Comparison

4.2.1 PSO-TVIW

Shi and Eberhart [11] improved the performance of the PSO method by using a linearly varying inertia weight (ω) (over iterations) from a predetermined maximum to a predetermined 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. We use these values while implementing their scheme. We call this version PSO-TVIW (PSO with Time Varying Inertia Weight).

4.2.2 HPSO-TVAC

Ratnaweera *et al.* [10] have suggested a parameter automation strategy for PSO where the cognitive component is reduced and the social component increased (by varying the acceleration coefficients C_1 and C_2 in (1)) linearly with time. They suggested another modification, named "self-organizing hierarchical particle swarm optimizer", in conjunction with the previously mentioned time varying acceleration coefficients (HPSO-TVAC). In this method the inertial velocity term is kept at zero and the modulus of the velocity vector is reinitialized to a random velocity, known as "re-initialization velocity", whenever the particle gets stagnant ($V_{id} = 0$) in some region of the search space. This way, a series of particle swarm optimizers are generated automatically inside the main particle system according to the behavior of the particles in the search space, until some stopping criterion is met. Following [10], in the present paper the re-initialization velocity is kept proportional to the maximum allowable velocity, V_{max}.

4.2.3 MPSO-TVAC

In this variant [10] of PSO, the velocity of a randomly selected particle is perturbed by a random mutation step-size if the global best-so-far solution does not improve for a predetermined number of generations. Following [10], we keep the mutation step-size proportional to the maximum allowable velocity. The acceleration coefficients are made to vary linearly with time here too.

4.3 Population Initialization

Since most of the test functions used in this paper have the global minimum at or near the origin of the search space, we use the asymmetric initialization method proposed by Angeline [1]. In this scheme the population is initialized only in a certain portion of the search space. The most common dynamic ranges found in the literature are used in this paper and all dimensions are confined to the same dynamic range [4], [2]. Table 1 shows the range of initialization and the range of search for each function.

Table 1. Initialization and dynamic range of search

Function	Range of search	Range of Initialization
f_1	$(-100, 100)^n$	$(50, 100)^n$
f_2	$(-50, 50)^n$	$(15, 30)^n$
f_3	$(-5.12, 5.12)^n$	$(2.56, 5.12)^n$
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 new method with respect to: (a) canonical PSO, (b) PSO-TVIW, (c) MPSO-TVAC, (d) HPSO-TVAC, and (e) classical DE. Thus a total of six algorithms were considered – one new, the other five 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's f_6 and Shekel's foxholes are two-dimensional. For a given function of a given dimension, fifty independent runs of each of the six algorithms were executed, and the average best-of-run value and the standard deviation were obtained. Different maximum generations (G_{max}) were used according to the complexity of the problem. For all benchmarks (excluding f_6 and f_7) the stopping criterion was set as reaching a fitness of 0.001. However, for Schaffer's f_6, the widely used error limit of 0.00001 [9], [14] was used and for Shekel's foxholes the criterion was 0.998.

In the case of DE, we chose the crossover constant CR = 0.9 and the scale factor R = 0.8. 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 [5] suggested that it is good to limit the maximum velocity, V_{max}, to the upper limit of the dynamic range of search, Xmax. We used this limit in this investigation. For MPSO-TVAC, we set the mutation probability to 0.4 and the mutation step-size was changed from V_{max} to $0.1V_{max}$ during the search. In the case of HPSO-TVAC, the re-initialization velocity was set to change from V_{max} to $0.1V_{max}$. For the new algorithm PSO-DV, we used β = 0.8. In PSO-DV, the value of the parameter N depends on the nature of the test function (see Table 2).

4.5 Population Size

It is common practice in DE research to use a population size ten times the dimensionality of the search space. We take up the same convention here for the DE. Eberhart and Shi [5] showed that the population size has hardly any effect on the performance of the

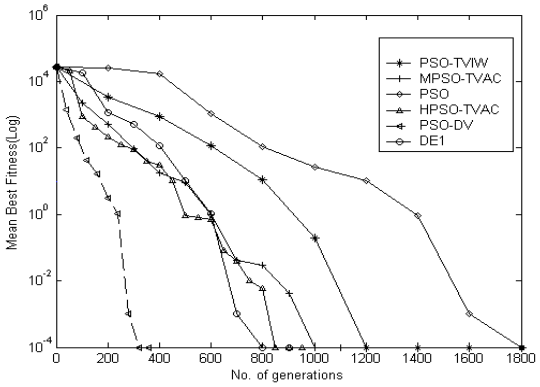
Table 2. Average and standard deviation of the best-of-run solution obtained for 50 runs of each of the six different methods.

F	Dim	N	G_{max}	Average (Standard Deviation)					
				PSO	PSO-TVIW	MPSO-TVAC	HPSO-TVAC	DE	PSO-DV
f_1	10	50	1000	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
	20	100	2000	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
	30	150	4500	**0.001**	**0.001**	**0.001**	**0.0001**	**0.001**	**0.001**
f_2	10	75	3000	21.705 (40.162)	16.21 (14.917)	1.234 (4.3232)	1.921 (4.330)	2.263 (4.487)	**0.0063 (0.0561)**
	20	150	4000	52.21 (124.32)	32.53 (75.309)	52.432 (117.178)	98.749 (120.175)	18.934 (9.453)	**0.0187 (0.554)**
	30	250	5000	76.87 (86.136)	61.56 (78.923)	32.222 (56.944)	17.134 (46.945)	0.0986 (0.338)	**0.0227 (0.182)**
f_3	10	50	3000	2.334 (2.297)	2.1184 (1.563)	1.78 (2.793)	0.039 (0.061)	0.006 (0.0091)	**0.0014 (0.0039)**
	20	100	4000	13.812 (3.491)	16.36 (4.418)	11.131 (0.91)	0.2351 (0.1261)	0.0053 (0.0032)	**0.0028 (0.0017)**
	30	150	5000	6.652 (21.811)	24.346 (6.317)	50.065 (21.139)	1.903 (0.894)	0.099 (0.112)	**0.0016 (0.277)**
f_4	10	50	2500	0.1613 (0.097)	0.092 (0.021)	**0.00561 (0.047)**	0.057 (0.045)	0.054 (0.0287)	0.024 (0.180)
	20	100	3500	0.2583 (0.1232)	0.1212 (0.5234)	0.0348 (0.127)	0.018 (0.0053)	0.019 (0.0113)	**0.0032 (0.0343)**
	30	150	5000	0.0678 (0.236)	0.1486 (0.124)	0.0169 (0.116)	0.023 (0.0045)	0.005 (0.0035)	**0.0016 (0.0022)**
f_5	10	50	2500	0.406 (1.422)	0.238 (1.812)	0.169 (0.772)	0.0926 (0.0142)	**0.00312 (0.0154)**	0.00417 (0.1032)
	20	100	3500	0.572 (3.094)	0.318 (1.118)	0.537 (0.2301)	0.117 (0.025)	0.029 (0.0067)	**0.0018 (0.028)**
	30	150	5000	1.898 (2.598)	0.632 (2.0651)	0.369 (2.735)	0.068 (0.014)	0.0078 (0.0085)	**0.0016 (0.0078)**
f_6	2	40	1000	0.0059 (1.672)	0.0068 (0.128)	0.0087 (0.3215)	0.00198 (0.0071)	0.00065 (0.0048)	**0.00021 (0.0015)**
f_7	2	40	1000	1.235 (2.215)	1.239 (1.468)	1.321 (2.581)	1.328 (1.452)	1.032 (0.074)	**0.9991 (0.0002)**

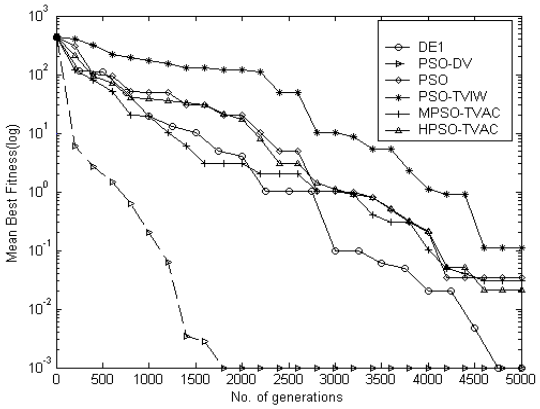
PSO method. Van den Bergh and Engelbrecht [14] have shown that though there is a slight improvement in the solution quality with increasing swarm sizes, a larger swarm increases the number of function evaluations to converge to an error limit. The present paper uses the "ten times" rule of population size for all the six algorithms.

5. RESULTS

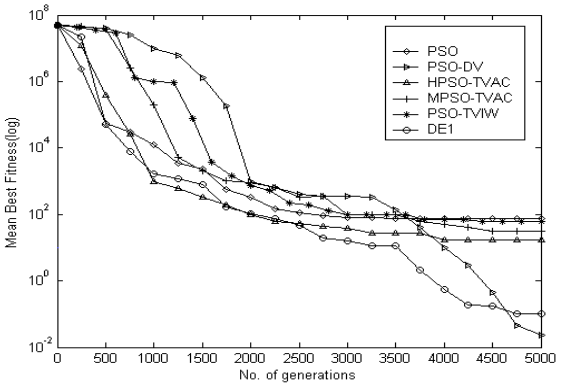
The following performance measures are used for our comparative study: (a) quality of the final solution, (b) speed of convergence towards the optimal solution, (c) success rate (frequency of hitting the optimum), and (d) scalability of the algorithms against the growth of problem dimensions. Table 2 compares the algorithms on the quality of the best solution. The mean and the standard deviation (within parentheses) of the best-of-run solution for 50 independent runs of each of the six algorithms are presented in Table 2. 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 3 shows results of unpaired *t*-tests between the best algorithm and the second best 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 3, sample size = 50 and degrees of freedom = 98. It is interesting to see from Tables 2 and 3 that in most cases the proposed method meets or beats the nearest competitor in a statistically meaningful way. Table 2 shows that only in two cases is the proposed method's mean numerically larger (i.e., worse) than the mean of the competitor (MPSO-TVAC or DE), but as Table 3 shows, this difference is *not* statistically significant. Table 4 shows, for the same set of runs as used in Tables 2 and 3, 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 1 we have graphically presented the rate of convergence of all the methods for all the functions. Figure 2 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 method leads to significant improvements in most cases.



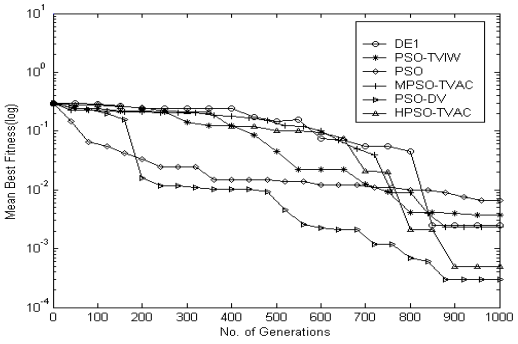
(a) Sphere Function



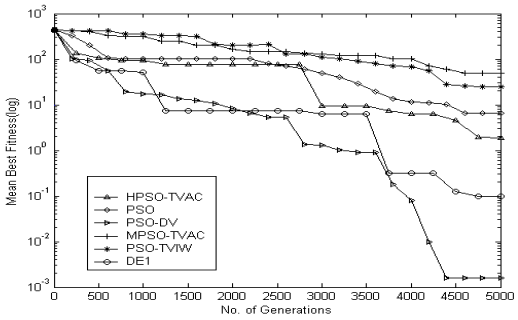
(b) Griewank Function



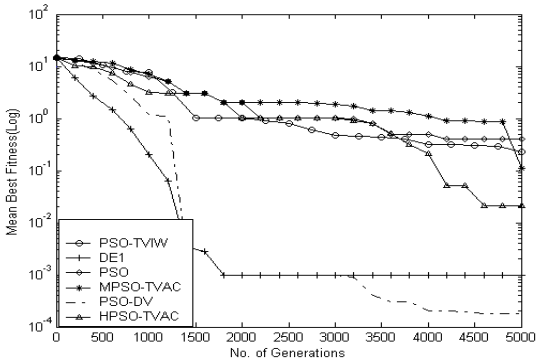
(c) Rosenbrock Function



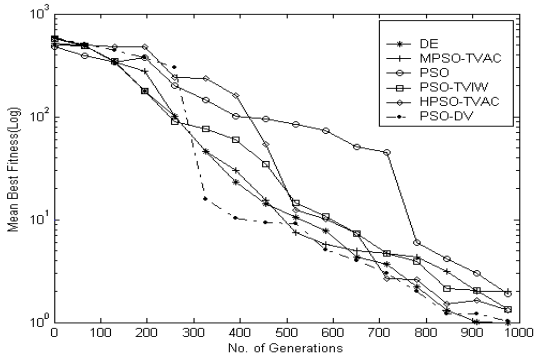
(d) Schaffer's f_6 function



(e) Rastrigin Function

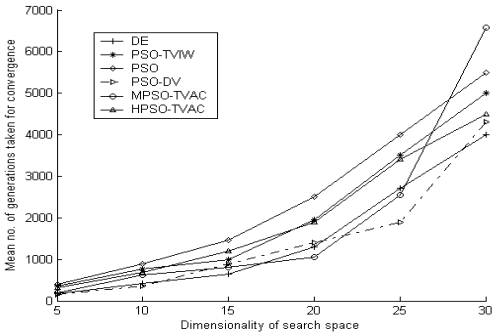


(f) Ackley Function

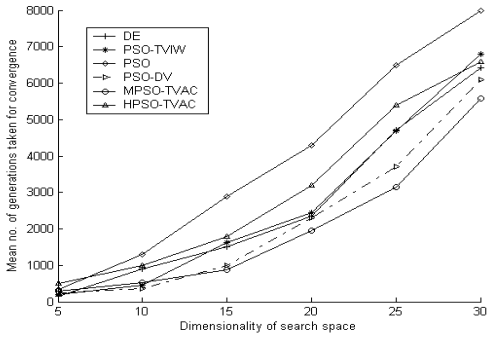


(g) Shekel's Foxholes Function

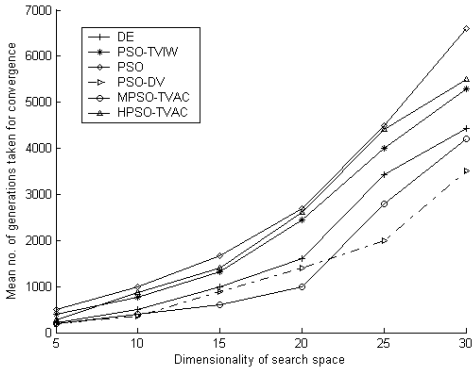
Figure 1. Variation of the mean best value with time (all the graphs are for dimension = 30 except for Schaffer's f6 and Shekel's Foxholes which are 2D)



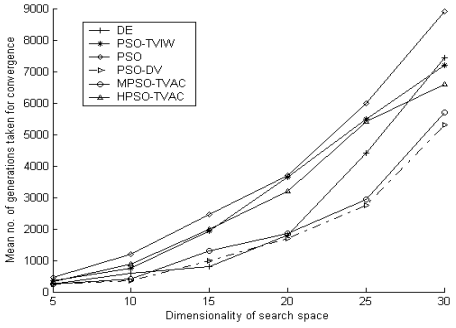
(a) Sphere Function



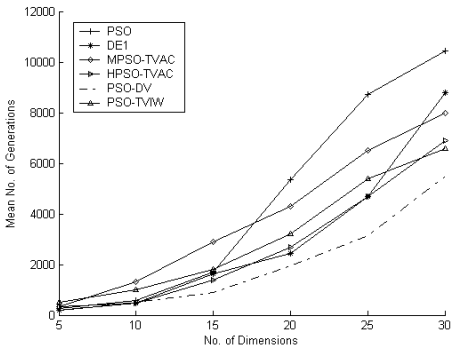
(b) Griewank Function



(c) Rastrigin Function



(d) Rosenbrock Function



(e) Ackley Function

Figure 2. Variation of mean convergence time with increase in dimensionality of the search space (the dashed line represents the new algorithm).

Table 3. Results of unpaired t-tests on the data of Table 2

Fn, Dim	Std. Err.	*t*	95% Conf. Intvl	Two-tailed *P*	Significance
f_2, 10	0.611	2.0079	(-2.4411, -0.0143)	0.0474	Significant
f_2, 20	1.339	14.1249	(-21.573, -16.258)	< 0.0001	Extremely significant
f_2, 30	0.054	1.3981	(-0.1836, 0.0318)	0.1653	Not significant
f_3, 10	0.001	3.2854	(-0.0074, -0.0018)	0.0014	Very significant
f_3, 20	0.001	4.8786	(-0.0035, -0.0015)	< 0.0001	Extremely significant
f_3, 30	0.042	2.3051	(-0.1813, -0.0135)	0.0233	Significant
f_4, 10	0.026	0.6990	(-0.0706, 0.0338)	0.4862	Not significant
f_4, 20	0.005	3.0153	(-0.0245, -0.0051)	0.0033	Very significant
f_4, 30	0.001	5.8156	(-0.0046, -0.0022)	< 0.0001	Extremely significant
f_5, 10	0.015	0.0712	(-0.0303, 0.0282)	0.9434	Not significant
f_5, 20	0.004	6.6804	(-0.0353, -0.0191)	< 0.0001	Extremely significant
f_5, 30	0.002	3.8002	(-0.0094, -0.0030)	0.0003	Extremely significant
f_6, 2	0.001	0.6187	(-0.0019, 0.0010)	0.5376	Not significant
f_7, 2	0.010	3.1437	(-0.0537, -0.0121)	0.0022	Very significant

Table 4. Number of runs (out of 50) to optimality and the corresponding mean number of generations

Fn	Dim	G_{max}	No. of runs converged to the optimality criterion (Average number of generations)					
			PSO	PSO-TVIW	MPSO-TVAC	HPSO-TVAC	DE	PSO-DV
f_1	10	1000	50 (618.4)	50 (820.9)	50 (667.4)	50 (349.7)	50 (348.3)	**50 (267.4)**
	20	2000	50 (2893.8)	50 (1353.6)	50 (1316.5)	50 (443.7)	50 (522.3)	**50 (434.7)**
	30	3000	50 (2193.5)	50 (2600.4)	50 (952.5)	50 (765.8)	50 (911.6)	**50 (506.9)**
f_2	10	3000	0	0	24 (2852.4)	18 (2441.5)	22 (2006.7)	**34 (2981.5)**
	20	4000	0	7 (3345)	2 (2331.5)	10 (3389.7)	15 (4067.3)	**16 (2213.7)**
	30	5000	0	0	0	10 (4793.8)	7 (5604.3)	**25 (4198.4)**
f_3	10	3000	2 (3100.5)	2 (3082)	35 (1296.4)	40 (2435.6)	36 (2298.2)	**40 (1675.3)**
	20	4000	0	0	34 (2398.5)	39 (5647.9)	35 (4655.8)	**43 (3984.7)**
	30	5000	0	0	13 (4386.8)	47 (6124.2)	43 (4087.6)	**48 (4751.4)**
f_4	10	2500	0	8 (2351.5)	5 (2393.8)	**19 (2079)**	12 (2066.5)	18 (2196.5)
	20	3500	13 (3286)	18 (3051.5)	29 (3329.7)	38 (2385.7)	35 (3249.4)	**41 (3007.8)**
	30	5000	34 (4087)	25 (4674.6)	8 (4541.5)	32 (4256.6)	46 (4043.2)	**48 (2209.4)**
f_5	10	2000	16 (1714.5)	28 (1714.3)	37 (1226.4)	40 (1640.5)	**45 (1631.5)**	41 (1582.6)
	20	3500	5 (3432.4)	7 (3284.5)	13 (2832.8)	27 (3411.0)	36 (3132.5)	**47 (2536.9)**
	30	5000	0	5 (4903.6)	15 (4448.3)	26 (3985.4)	40 (4973.6)	**43 (4956.3)**
f_6	2	1000	19 (625.4)	12 (386.5)	35 (423.6)	18 (344.8)	36 (621.5)	**41 (503.1)**
f_7	2	1000	19 (457.8)	26 (892.7)	33 (848.3)	45 (617.8)	43 (756.4)	**48 (343.1)**

6. CONCLUSION

A new, efficient PSO algorithm has been presented and has been shown to improve performance in a statistically meaningful way. The new method has been compared against (a) the basic DE, (b) the PSO, and (c) three best-known PSO-variants, using a seven-function test suite, on the following performance metrics: (a) solution quality, (b) speed of convergence, (c) frequency of hitting the optimum, and (d) scalability.

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 also thankful to four anonymous reviewers 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, 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] Clerc, M., Kennedy, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space, In *IEEE Transactions on Evolutionary Computation* (2002) 6(1): 58-73.

[4] 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.

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

[6] Higashi, N., Iba, H. Particle swarm optimization with Gaussian mutation, In *IEEE Swarm Intelligence Symposium* (2003) 72-79.

[7] Kennedy, J. Bare bones particle swarms, In *Proceedings of IEEE Swarm Intelligence Symposium*, (2003) 80-87.

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

[9] Kennedy, J. Stereotyping: Improving particle swarm performance with cluster analysis, In *Proceedings of IEEE International Conference on Evolutionary Computation*, vol. 2 (2000), 303-308.

[10] Ratnaweera, A., Halgamuge, K.S. Self organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, In *IEEE Transactions on Evolutionary Computation* (2004) 8(3): 240-254.

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

[12] 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.

[13] Trelea, C. I. The particle swarm optimization algorithm: convergence analysis and parameter selection, *Information Processing Letters* (2003), 85(6), 317–325.

[14] 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.

[15] Xie, X. F., Zhang, W. J., Yang, Z. L. A dissipative particle swarm optimization, In *Proceedings of IEEE Congress on Evolutionary Computation* (2002), 1456-1461.

[16] Xie, X. F., Zhang, W. J., Yang, Z. L. Adaptive particle swarm optimization on individual level, *In Proceedings of International Conference on Signal Processing* (2002), 1215-1218.

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