# Synergy of Particle Swarm Optimization with Evolutionary Algorithms for Intelligent Search and Optimization

Swagatam Das[1] and Ajith Abraham[2]

[1] *Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700032, India.*

[2] *IITA Professorship Program,School of Computer Science, Yonsei University,134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Republic of Korea*
*ajith.abraham@ieee.org*

**Abstract.** The concept of Particle Swarms, although initially introduced for simulating human social behaviors, has become very popular these days as an efficient global optimization technique. On the other hand, a keen observation of the underlying relation between optimization and biological evolution has led to the development of an important paradigm of Computational Intelligence, marked as 'Evolutionary Algorithms' (EA). The EA algorithms are ubiquitously used for performing very complex search and optimization. In this article we focus on the possible synergies of these two powerful search techniques in order to enjoy the best of both the worlds i.e. the fast convergence and high diversity. We provide a few computer simulations undertaken for this study to demonstrate the effectiveness of the hybrid algorithms.

**Keywords:** Particle Swarm Optimization, Evolutionary Algorithms, Differential Evolution, Global Optimization, Swarm Intelligence.

## 1 Introduction

Scientists and engineers from all disciplines often have to deal with the classical problem of search and optimization. Optimization means an action of finding the best-suited solution of some problem within the given constraints and flexibilities. It has now become a well known fact among the researchers that a class of stochastic search algorithms can perform better on complex real life optimization problems as compared to the classical deterministic algorithms such as the steepest descent search [1]. Particle Swarm Optimization (PSO) [2, 3] and Evolutionary Algorithms (EA) [4-7] are two important members of the former class.

The PSO is based on the simulation of the collective behavior of a flock of birds or the movements of a school of fish. The dynamics of the search is motivated by the modes of communication among the members of such a swarm of social creatures. The particles are conceptual mathematical entities, which accelerate simultaneously along two directions – the best positions of the search space individually experienced by each of them at some point of time and the globally best position found by the entire swarm so far. Thus the particles have a tendency to fly towards the better and better regions of the search space over time, which results in the fast convergence of the search. PSO requires no gradient information of the function to be optimized, is very easy to implement in any standard programming language and uses only primitive mathematical operators throughout. Due to these features, the algorithm has become very popular among the researchers since its advent in 1995.

Evolutionary algorithms on the other hand, employ selection and mutation operators to locate the global maxima/minima of a complex objective function. It also starts with a population of agents or the trial solutions of the search problem. The selection operator forces the individual agent to find better fitness in order to survive to the next generation. The mutation operator brings about diversity in the population to avoid premature convergence or trapping in some local optima. The main concept in EA is to keep the competition in the population. It may be mentioned here that the key concept in PSO is the cooperation among the population members.

In the present article, we discuss a few hybrid algorithms that integrate evolutionary operators, such as selection and mutation, into the standard PSO algorithm. The chapter gives special emphasis on a recently developed hybrid algorithm known as PSO-DV (Particle Swarm Optimization with Differentially perturbed Velocity) [8]. The algorithm synergistically combines PSO with a very powerful member of the EA family, well-known as Differential Evolution (DE) [9, 10]. It incorporates a selection mechanism in PSO and thus saves the limited computational source by prohibiting the particles from visiting the useless regions of the search space. It also incorporates the vector differential operator borrowed from DE, in the PSO dynamics.

The rest of the chapter is organized as follows. Section 2 provides a brief outline of the classical PSO and the EA family of algorithms. In section 3, we review a few hybrid PSO-EA algorithms developed in recent past. Section 4 describes the PSO-DV algorithm in sufficient details. A performance comparison of the PSO-DV with the original PSO and DE on a few representative benchmark objective functions has been provided in section 5. Finally, the chapter is concluded in section 6.

## 2 Brief Introduction to PSO and EA

PSO is in principle such a multi-agent parallel search technique. Particles are conceptual entities which fly through the multi-dimensional search space. At any particular instant each particle has a position and a velocity. The position vector of a particle with respect to the origin of the search space represents a trial solution of the search problem. In classical PSO, a population of particles is initialized with random positions $\vec{X}_i$ and velocities $\vec{V}_i$, and a function, *f*, is evaluated, using the particle's positional coordinates as input values. In a D-dimensional search space, $\vec{X}_i$ = ($X_{i1}$, $X_{i2}$, $X_{i3}$ ...$X_{iD}$) and $\vec{V}_i$ = ($V_{i1}$, $V_{i2}$, $V_{i3}$ ...$V_{iD}$). Positions and velocities are adjusted, and an objective function is evaluated with the new coordinates at each time-step. The fundamental velocity and position update equations for the d-th dimension of the i-th particle in the swarm may be given as

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

The variables $\varphi_1$ and $\varphi_2$ are random positive numbers, drawn from a uniform distribution and restricted to an upper limit $\varphi_{max}$ that is a parameter of the system. $C_1$ and $C_2$ are called acceleration constants whereas $\omega$ is called inertia weight. $P_i$ 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.

Below we illustrate the general principle of an EA with a simple pseudo-code. Here P(t) denotes a population of chromosomes (trial solutions of the problem at hand) at time t. The procedure initializes a population P(t) randomly at iteration t = 0. The function: Evaluate P(t) determines the fitness of the chromosomes by employing a specially constructed fitness measuring function. The 'while' loop includes three main steps. First it increases the iteration index by 1. Next it selects a population P(t) from P(t-1) based on the results of fitness evaluation. The function: Alter P(t) evolves P(t) by some complex non-linear operations. The while loop then re-evaluates P(t) for the next iteration, and continues evolution until the terminating condition is reached.

```
Procedure Evolutionary-Computation
Begin
  t←0;
  Initialize P(t);
  Evaluate P(t);
  While (terminating condition not reached) do
   Begin
    t←t+1;
    Select P(t) from P(t-1);
    Alter P(t);
    Evaluate P(t);
   End While;
End.
```

The main functioning loop of an EA can also be illustrated in figure 1. Depending upon the alteration scheme of P(t) and the selection mechanisms, the EAs can have a wide range of variants. A comprehensive foundation of the various forms of EA can be found in [5, 10].

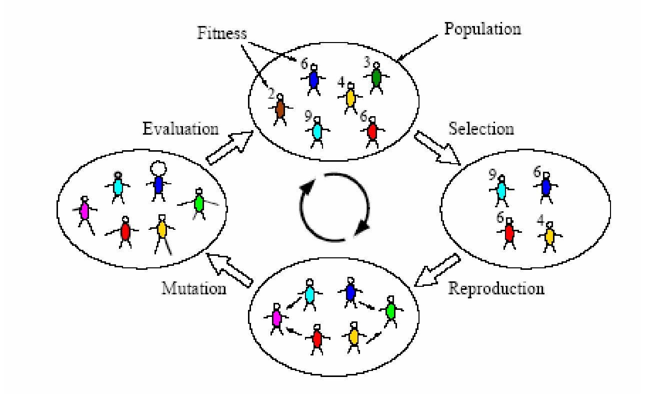


**Fig. 1.** The main loop of an EA

## 3 The Hybrid PSO-EA Algorithms – An Outline

Both the social and cognitive components of PSO focus more on the cooperation of the particles. With memory, each particle can track its personal best performance and that achieved in its neighborhood throughout its life time. However, the particles in PSO are not eliminated even when they come up with worst fitness (in terms of the objective function value) and thus waste the limited computational resources. On the other hand, the individuals in EA compete for survival, but the winning survivors hardly retain sufficient history. Clearly, the advantage of one algorithm can compensate for the other's shortcoming. Therein lays the motivation to develop hybrid algorithms based on PSO and EA.

Based on the complementary properties of PSO and EA, Cai *et al.* [11, 12] proposed a promising hybrid algorithm, which they successfully used in training the feed-forward neural networks. In each generation, the hybrid algorithm selects half of the PSO population as the winners (elites) according to the fitness value, and discards the rest half as losers. These elites are enhanced, sharing the information in the community and benefiting from their learning history using a standard PSO procedure. The enhanced elites then serve as parents for an EA mutation operation to produce the same amount of offspring to fill up the vacuum that the discarded individuals left in the population size. The offspring also inherit the social and cognitive information from the corresponding parents and carry this to the next generation if they become the winners. Cai *et al.* applied the hybrid PSO-EA algorithms in neural network learning and showed that it performs better than both PSO and EA in terms of speed and accuracy. Some other approaches to combine PSO with EAs can be traced in the works like [13]. In what follows, we describe a novel hybrid algorithm based on PSO and Differential Evolution (DE). The DE is a recent addition to the vast realm of EAs. Shortly after its advent in 1995, the algorithm appeared as an attractive alternative to the classical EAs including the Genetic Algorithms (GAs) in several real world search problems.

## 4 A Synergism of PSO and DE – the PSO-DV Algorithm

The canonical PSO has been subjected to empirical [14-16] and theoretical [17, 18] investigations by several researchers. In many occasions the convergence is premature, especially if the swarm uses a small inertia weight ω [19] or constriction coefficient [17]. 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 [20, 21, and 16].

In an attempt to circumvent the problems mentioned in the previous section, Das *et al.* coupled a differential operator with the velocity-update scheme of PSO [8]. 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:**

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

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

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

where CR is the crossover probability, $\delta_d$ 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. Creation of the trial location is illustrated in figure 2.

Now, a new trial location $Tr_i$ is created for the particle by adding the updated velocity to the previous position $X_i$:

$$\vec{Tr}_i = \vec{X}_i(t) + \vec{V}_i(t+1) \quad \textbf{(4)}$$

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 $f(\vec{X})$, then the target particle is relocated as follows:

$$\left.\begin{aligned} \vec{X}_i(t+1) &= \vec{Tr} && \text{if } f(\vec{Tr}_i) < f(\vec{X}_i(t)) \\ \vec{X}_i(t+1) &= \vec{X}_i(t) && \text{Otherwise} \end{aligned}\right\} \quad \textbf{(5)}$$

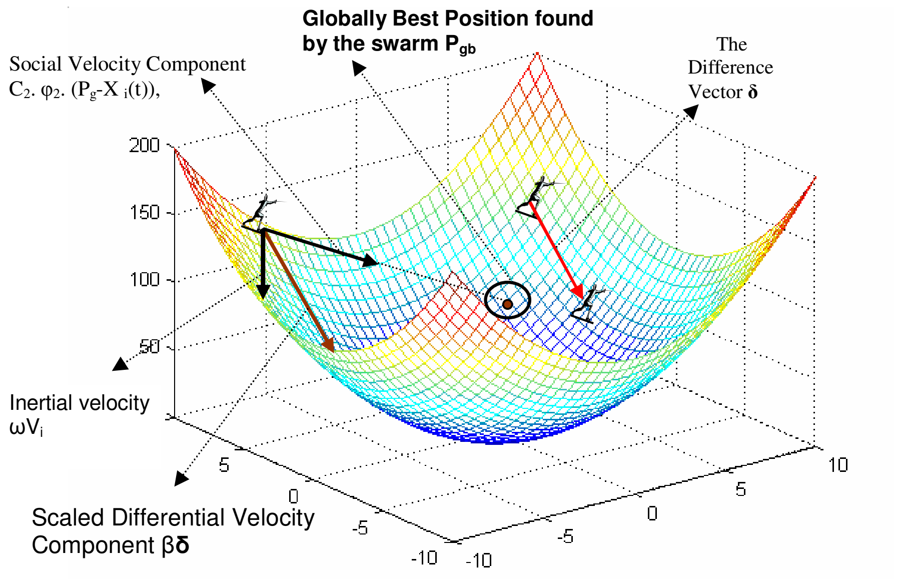


**Fig. 2.** Illustrating PSO-DV on a two dimensional function surface

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 $|P_{lid}-X_{id}|$ 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":

**If** $((\vec{X}_i(t) = \vec{X}_i(t+1) = \vec{X}_i(t+2) = .... = \vec{X}_i(t+N))$ and $(f(\vec{X}_i(t+N))$

**then**

for ( r = 1 to n)

$$X_{ir}(t+N+1) = X_{min} + rand_r(0, 1)*(X_{max}-X_{min}) \qquad \textbf{(6)}$$

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 pseudo code for this new method, called PSO-DV (Particle Swarm Optimization with Differentially perturbed Velocity), is presented below:

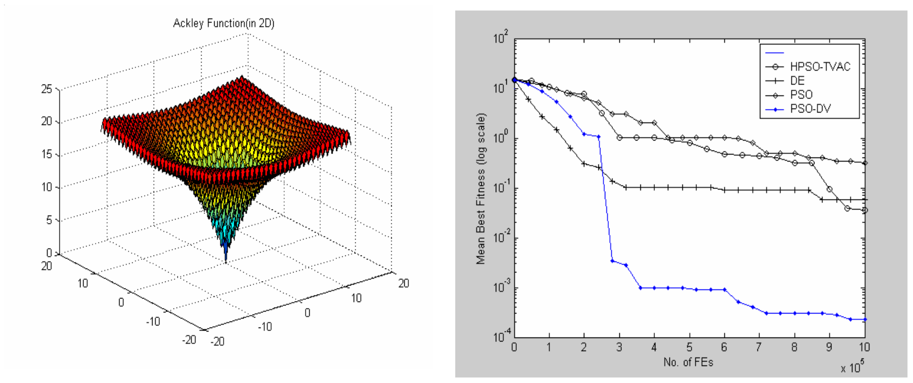
```
Procedure PSO-DV
begin
 initialize population;
 while stopping condition not satisfied do
```

```
  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 $\delta = \vec{X}_k - \vec{X}_j$ ;
    for d = 1 to no_of_dimensions
     if $rand_d$ (0, 1) < CR
       $V_{id}(t+1) = \omega.V_{id}(t) + \beta.\delta_d + C_2.\ \varphi_2.\ (P_{gd}-X_{id}(t))$;
     else $V_{id}(t+1) = V_{id}(t)$;
     endif
    endfor
    create trial location as $\vec{Tr}_i = \vec{X}_i(t) + \vec{V}_i(t+1)$ ;
     if ( $f(\vec{Tr}_i) < f(\vec{X}_i(t))$ ) then $\vec{X}_i(t+1) = \vec{Tr}$ ;
       else $\vec{X}_i(t+1) = \vec{X}_i(t)$ ;
      endif
  endfor
    for i = 1 to no_of_particles
     if $X_i$ stagnates for N successive generations
              for r = 1 to no_of_dimensions
               $X_{ir}(t+1) = X_{min} + rand_r(0, 1)*(X_{max}-X_{min})$
               end for
      end if
     end for
 end while
end
```

## 5 Experimental Results

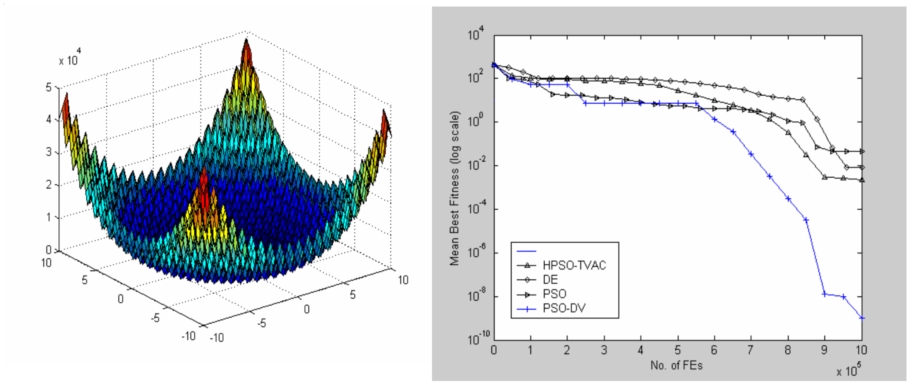
In this section we present a performance comparison of the PSO-DV algorithm with classical PSO, DE and a recently proposed improved version of PSO over a test bed of three well-known multi-modal benchmark functions. The PSO-variant that we consider are known as HPSO-TVAC (Self Organizing Hierarchical PSO with Time Varying Acceleration Coefficient) [22]. For DE and PSO-DV, we chose a population size equal to ten times the dimension of the test function, whereas for the two PSO algorithms, we used a swarm of 40 particles. The functions were tested for 10, 20, 30, and 40 dimensions. Through empirical studies on numerical benchmarks, Eberhart and Shi [23] suggested that it is good to limit the maximum velocity, $V_{max}$, to the upper limit of the dynamic range of search, $X_{max}$. We used this limit in this investigation. In case of the HPSO-TVAC algorithm, the acceleration-coefficient $C_1$ was varied from 0.35 to 2.4 and that for $C_2$ should be from 2.4 to 0.35 in equation (1). Moreover, the re-initialization velocity was set to change from $V_{max}$ to $0.1V_{max}$. For the PSO-DV algorithm, we used $\beta = 0.8$. In PSO-DV, the value of the parameter N depends on the nature of the test function. The classical PSO algorithm is applied with inertia factor ω = 0.749, $C_1 = C_2 = 1.494$. In case of the classical DE, we choose the conventional parameter set up [9]: CR = 0.9 and scale factor F = 0.8.

In figure 3 to 5, we have graphically presented the rate of convergence of all the methods on three test functions (in 30 dimensions). Figure 6 shows the scalability of the six methods on three test functions -- how the average time to convergence varies with an increase in the dimensionality of the search space. We used the number of Fitness Function Evaluations (FEs) as a measure of computation time instead of 'generations' or 'iterations', since different algorithms perform different amounts of work in their inner loops through each iteration.



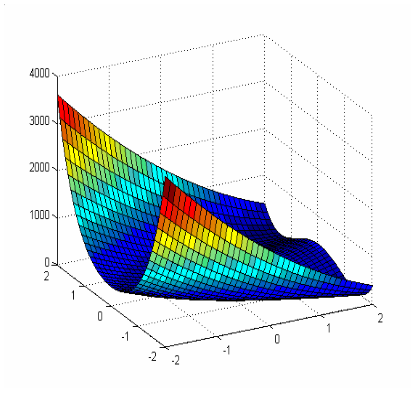
(a) (b)

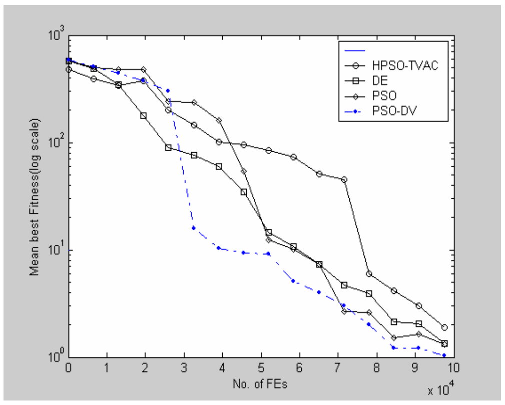
**Fig. 3.** (a) Fitness landscape of the 2-dimensional Ackley function (b) Progress toward the optima with computation time, for four competitive algorithms.



(a) (b)

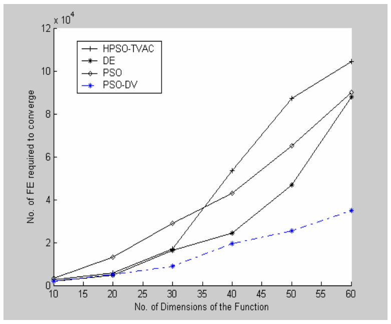
**Fig. 4.** (a) Fitness landscape of the 2-dimensional Griewank function (b) Progress toward the optima with computation time, for four competitive algorithms.


(a)

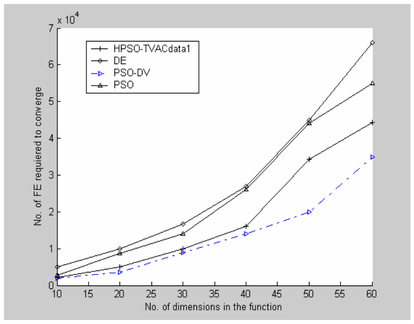


(b)

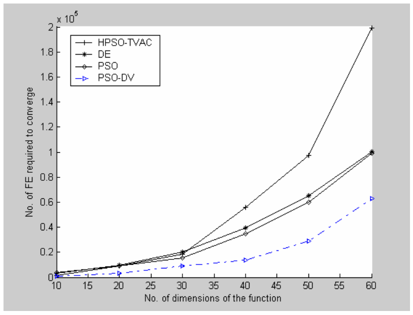
**Fig. 5.** (a) Fitness landscape of the 2-dimensional Rosenbrock function. (b) Progress toward the optima with computation time, for four competitive algorithms.



(a)



(b)



(c)

**Fig. 6.** Variation of mean convergence time with increase in dimensionality of the search space (a) Ackley Function (b) Griewank Function (c) Rosenbrock function.

As can be perceived from the figures 3 to 5, in all the test cases, PSO-DV outperforms the original PSO, DE and a state-of-the-art variant of the PSO, in terms of both accuracy and speed. Detailed experimental results reported in [8] reveal that over seven standard benchmark functions, PSO-DV meet or beat all other competitive algorithms in a statistically meaningful way. The particles in PSO-DV were able to land at or very near to the global optima of the fitness landscape at a much faster rate. A close scrutiny of figure 5 shows that PSO-DV is least affected by the increase of the dimensionality of the search space as compared to its competitors.

## 6 Conclusions

PSO and EA mark two promising families of algorithms for global search and optimization of current interest. In this chapter the combination of the search capabilities of these two methods has been explored. A novel synergy of PSO and DE has been presented in sufficient details. The hybrid algorithm inherits both cooperative and competitive characteristics from PSO and EA. It shares the searching information within the promising individuals, which leads to faster convergence, while replacing the worse individuals from the offspring of elites or prohibiting the particles from visiting the points far away from the global optima as we have seen for the PSO-DV. In this way, the search of the particles soon becomes more focussed near the global optima thus saving the computational cost. The purpose of incorporating DE-type mutation to PSO is to increase the diversity of the population and thus enable the particles to escape from local optima. Future research may focus on incorporating other kinds of mutation and crossover strategies (like the simplex crossover, the parent centric recombination and so on [24]) from EA domain in PSO. Some PSO concepts like that of the neighbourhood topologies may also be incorporated in EAs like DE. Preliminary work towards this direction has been undertaken in [25].

## References

1. Snyman, A.: Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms. Springer Publishing. (2005).
2. Kennedy, J.: Eberhart, R. Particle swarm optimization, In Proceedings of IEEE International Conference on Neural Networks, (1995) 1942-1948.
3. Kennedy, J.: Eberhart, R. and Shi, Y., Swarm Intelligence, Morgan Kaufmann Academic Press (2001).
4. Fogel, L. J., Owens A. J. and Walsh, M. J.: Artificial Intelligence Through Simulated Evolution. New York: Wiley, (1966).
5. Yao, X.: Evolutionary Computation: Theory and Applications, World Scientific Press (1999).
6. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
7. Schwefel, H.-P. : Evolution and Optimum Seeking. New York, NY: Wiley, 1st edition (1995).

8. Das, S., Konar, A. and Chakraborty, U. K.: An Improved Particle Swarm Optimization Algorithm for Faster Global Search in ACM-SIGEVO Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2005), Washington DC, (2005).
9. Storn, R., Price, K.: Differential evolution – A Simple and Efficient Heuristic for Global continuous spaces, Journal of Global Optimization, 11(4) (1997) 341–359.
10. DeJong, K. A.: Evolutionary Computation, a Unified Approach, MIT Press, (2002).
11. Cai, X., Zhang, N., Venayagamoorthy G. K. and Wunsch II D. C.: Time series prediction with recurrent neural networks using hybrid PSO-EA algorithm, Proc. of INNS-IEEE International Joint Conference on Neural Networks (IJCNN), Vol. 2, (2004) 1647-1652, Budapest, Hungary.
12. Cai, X., Zhang, N., Venayagamoorthy G. K. and Wunsch II D. C.: Time series prediction with recurrent neural networks using hybrid PSO-EA algorithm, Neurocomputing, accepted for publication, 2005.
13. Løvbjerg, M., Rasmussen, T., K. and Krink, T. Hybrid Particle Swarm Optimizer with Breeding and Subpopulations. In: Proceedings of the third Genetic and Evolutionary Computation Conference (GECCO-2001), vol. 1, (2001) 469-476.
14. 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.
15. Kennedy, J.: Bare bones particle swarms, In Proceedings of IEEE Swarm Intelligence Symposium, (2003) 80-87.
16. Xie, X. F., Zhang, W. J., Yang Z. L.: A dissipative particle swarm optimization, In Proceedings of Congress on Evolutionary Computation (2002), 1456-1461.
17. Clerc, M. and 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.
18. Trelea, C. I.: The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters (2003), 85(6), 317–325.
19. Eberhart, R. C. and Shi, Y.: Particle swarm optimization: Developments, applications and resources, In Proceedings of IEEE International Conference on Evolutionary Computation, vol. 1 (2001), 81-86.
20. Higashi, N. and Iba, H.: Particle swarm optimization with Gaussian mutation, In IEEE Swarm Intelligence Symposium (2003) 72-79.
21. Xie, X., F, Zhang, W., J. and Yang, Z, L.: Adaptive particle swarm optimization on individual level, In Proceedings of International Conference on Signal Processing (2002), 1215-1218.
22. Ratnaweera, A. and 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.
23. Eberhart, R. C. and 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.
24. Deb, K., Anand, A. and Joshi, D.: A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization, Evolutionary computation, 10(4), (2002) 371 – 395.
25. Chakraborty, U. K., Das, S. and Konar, A.: DE with Local Neighborhood proceedings of Congress on Evolutionary Computation (CEC 2006), Vancouver, BC, Canada, IEEE Press.