Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Operations Research 33 (2006) 859–871

computers & operations research

www.elsevier.com/locate/cor

Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization

A. Chatterjee[a,*], P. Siarry[b]

[a]*Electrical Engineering Department, Jadavpur University, Kolkata 700 032, India*
[b]*Université Paris XII, Faculté des Sciences (LERISS), 94010 Créteil, France*

Available online 17 September 2004

Abstract

The particle swarm optimization (PSO) is a relatively new generation of combinatorial metaheuristic algorithms which is based on a metaphor of social interaction, namely bird flocking or fish schooling. Although the algorithm has shown some important advances by providing high speed of convergence in specific problems it has also been reported that the algorithm has a tendency to get stuck in a near optimal solution and may find it difficult to improve solution accuracy by fine tuning. The present paper proposes a new variation of PSO model where we propose a new method of introducing nonlinear variation of inertia weight along with a particle's old velocity to improve the speed of convergence as well as fine tune the search in the multidimensional space. The paper also presents a new method of determining and setting a complete set of free parameters for any given problem, saving the user from a tedious trial and error based approach to determine them for each specific problem. The performance of the proposed PSO model, along with the fixed set of free parameters, is amply demonstrated by applying it for several benchmark problems and comparing it with several competing popular PSO and non-PSO combinatorial metaheuristic algorithms.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Combinatorial metaheuristics; Particle swarm; Nonlinear inertia weight; Fixed parameter set

1. Introduction

Particle swarm optimization (PSO) has very recently evolved as an important branch of combinatorial metaheuristic techniques which operate on a population of potential solutions to explore the search

* Corresponding author.
E-mail addresses: cha_ami@yahoo.co.in (A. Chatterjee), siarry@univ-paris12.fr (P. Siarry).

space for optimization. Like ant colony optimization (ACO) technique [1], particle swarm was originally formulated [2] to study the interesting concept of a social behavior (bird flocking or fish schooling) in a simplified manner in simulation. However, very soon the potential of this technique was realized to develop into a powerful optimization tool [3] which can be successfully applied in the fields of both continuous and discrete optimization problems. Nowadays PSO and ACO have developed to be real competitors for other comparatively older and well-established techniques for population-based evolutionary computation, notably Genetic Algorithms, Evolutionary Programming, Genetic Programming etc. and other prominent non-population based metaheurisitc techniques e.g. simulated annealing and tabu search. Among PSO and ACO, PSO has particularly gained prominence due to its relative ease of operation and capability to quickly arrive at an optimal/near-optimal solution. However it was pointed out that although PSO can show significant performance in the initial iterations, the algorithm might encounter problems in reaching optimum solutions efficiently for several function approximation problems [4]. Since then the researchers have engaged themselves to improve upon the original PSO variation to increase accuracy of solution without sacrificing the speed of solution significantly. An improved PSO algorithm was proposed when Shi and Eberhart [5] introduced an inertia weight making the influence of the previous velocity on the new velocity a flexible one. The impact of the choice of a proper maximum allowable velocity in each dimension ($v_{\max}$) along with the linearly varying inertia weight was studied in [6] where maximum allowable velocity serves as a free controllable parameter that can be used to limit (or allow) global exploration capability. The studies also showed that choice of a $v_{\max}$ equal to maximum allowable excursion of any particle in that dimension i.e. $x_{\max}$, in absence of any knowledge regarding absolute value of $v_{\max}$, is a reasonable choice. However this PSO version was tested for a single benchmark problem and hence it could not give enough guidelines for selection of parameters like inertia weight and maximum velocity for a large variety of problems.

Another interesting variation of PSO has been reported in [7]. They introduced the new concept of constriction coefficient which should control each of the three components of the velocity update relation to limit any explosion of the particles beyond limits. This paper also presented an excellent, rigorous analysis of the particles' trajectory in PSO to search the problem space. They also reported that the latest PSO version [8] which is a modification of the type 1" version has shown some very encouraging results.

A hybrid-type PSO model has been reported in [9] where velocity and position update relations employed the concepts of breeding and subpopulation. A new concept of breeding probability has been introduced to select the parent particles for breeding and arithmetic crossovers on the basis of the position of the parents to determine the offspring. However the hybrid PSO model becomes computationally heavy due to the additional burden of breeding and subpopulation. Attempts have also been made to apply PSO optimization in different areas of applications e.g. in neural networks and control applications. Very recently PSO has been successfully applied to design a model based predictive controller for the environmental temperature control of a greenhouse [10].

The present paper makes a humble attempt to determine a generalized framework for implementation of PSO where suitable free parameters are suggested for PSO to show reasonably satisfactory performance for a wide range and variety of test functions. The determination of suitable parameter values and empirical rules for parameter selection saves the user of the tedious trial and error based approaches for determination of these parameters, which are till now believed to be problem dependent. The system also proposes a new improved dynamic adaptation technique for PSO models where the inertia weight is varied dynamically as nonlinear functions of iterations to improve performance. As the main objective was to achieve faster speed of convergence or better solution accuracy with minimum incremental computational burden, the function mappings are kept simple as far as practicable. Performance of the proposed PSO model is compared with some other available PSO models reported in the literature available as well as some other well established combinatorial metaheurisitc techniques to establish its usefulness.

Rest of the paper is organized as follows. Section 2 describes the classical PSO model along with its different variations. Section 3 describes our proposed improved PSO model with dynamic adaptation. Section 4 analyses the behavior of classical PSO model along with our proposed model for one classical test function i.e. sphere function. Section 5 presents the performance of the proposed PSO model vis-à-vis other PSO models and other evolutionary programming models. Section 6 presents the conclusions.

2. Particle swarm optimization (PSO)—static PSO models

In particle swarm optimization a group of particles, referred to as the population or potential solutions, is flown in a multidimensional search space to determine the optimum position of them. This optimum position is usually characterized by the optimum of a fitness function. Each “particle” i is represented by a vector in multidimensional space to characterize its position ($\mathbf{x}_i$) and another vector to characterize its velocity ($\mathbf{v}_i$) at the current time step. In order to pursue the optimum of the fitness function (f), velocity $\mathbf{v}_i$ and hence position $\mathbf{x}_i$ of each particle are adjusted in each time step. The updated velocity in each time step $\mathbf{v}_{i\text{new}}$ is a function of three major components: the old velocity vector of the same particle ($\mathbf{v}_{i\text{old}}$) (component I), difference of the ith particle’s best position found so far (called $\mathbf{p}_i$) and the current position ($\mathbf{x}_i$) (component II) and difference of the best position of any particle within the context of the topological neighborhood of ith particle found so far (called $\mathbf{p}_g$) and current position of the ith particle ($\mathbf{x}_i$) (component III). Now, each of components II and III are stochastically weighted and added to component I [7] to update velocity of each particle with enough oscillations that should empower each particle to search for a better pattern within the problem space. The PSO algorithm is shown in Fig. 1. In traditional

```
Initialize no. of particles
DO
        FOR $i$ = 1 to no. of particles
        IF $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ THEN $\mathbf{p}_i = \mathbf{x}_i$
        $\mathbf{p}_g = \min(\mathbf{p}_{\text{neighbors}})$
            FOR d = 1 to Dimension
                Calculate new velocity of the particle
                Calculate new position of the particle
          ENDFOR
        ENDFOR
UNTIL termination criteria is satisfied
```

Fig. 1. The particle swarm optimization (PSO) algorithm.

PSO model the velocity and position update relations, in the *d*th dimension, are given as [6].

$$v_{i\,d\,\text{new}} = \underset{\text{component I}}{v_{i\,d\,\text{old}}} + \underset{\text{component II}}{\varphi_i(p_{id} - x_{id})} + \underset{\text{component III}}{\varphi_g(p_{gd} - x_{gd})} \tag{1a}$$

$$\text{if } (v_{i\,d\,\text{new}} > v_{\max}) \text{ then } v_{i\,d\,\text{new}} = v_{d\,\max}, \tag{1b}$$

$$\text{if } (v_{i\,d\,\text{new}} < -v_{\max}) \text{ then } v_{i\,d\,\text{new}} = -v_{d\,\max}, \tag{1c}$$

$$x_{i\,d\,\text{new}} = x_{i\,d\,\text{old}} + v_{i\,d\,\text{new}}, \tag{1d}$$

$$v_{i\,d\,\text{old}} = v_{i\,d\,\text{new}}, \tag{1e}$$

$$x_{i\,d\,\text{old}} = x_{i\,d\,\text{new}}, \tag{1f}$$

φ_i and φ_g are responsible for introducing stochastic weighting to components II and III, respectively, and they are popularly chosen to vary within [0, 2]. This traditional PSO model showed quick, aggressive movement narrowing down towards the solution region but often encountered problem in fine tuning the search to determine the supreme solution. Since then two important modifications of PSO attempted to rectify this problem by introducing a judicious mix of aggressive, coarse updating during early iterations and relaxed, fine updating during later iterations. One of the important improvements of PSO models introduced “inertia weight” to the component I i.e. $v_{i\,d\,\text{old}}$ [5,6]. Hence the velocity update rule was given as

$$v_{i\,d\,\text{new}} = w(v_{i\,d\,\text{old}}) + \varphi_i(p_{id} - x_{id}) + \varphi_g(p_{gd} - x_{id}) \tag{2}$$

as an improvement over (1a). The choice of the inertia weight (*w*) can be judiciously made along with a proper choice of $v_{d\,\max}$ to optimize the contribution of global and local exploration capabilities [6].

Another important modification is proposed in form of “constriction coefficients” to restrict a particle to take smaller updating in each step [7]. It was reported that the random nature of variation of the control parameters in components II and III of the velocity update relation may result in some kind of a “drunkard’s walk” which may cause explosion of the particles towards infinity (unless it is limited by a suitable $v_{\max}$). This constriction coefficient influences each of the three components simultaneously. In its simple, generalized form, they update velocity relation as:

$$v_{i\,d\,\text{new}} = \chi(v_{i\,d\,\text{old}} + \varphi_i(p_{id} - x_{id}) + \varphi_g(p_{gd} - x_{id})), \tag{3}$$

where χ is the constriction coefficient.

3. Proposed PSO model with dynamic adaptation

While introducing the concept of inertia weight (*w*), Shi and Eberhart observed that a reasonable choice for *w* should decrease with a higher choice of $v_{\max}$. In fact, when $v_{\max}$ is assigned the same value as $x_{\max}$, a reasonable *w* under this condition coincides with that value obtained for *w*, when $v_{\max}$ is independently chosen a high value. As a general remark, Shi and Eberhart opined that a better performance would be obtained if the inertia weight were chosen a time varying, linearly decreasing quantity, rather than being a constant value and supported their statement with a single case study. A higher value of the inertia weight implied larger incremental changes in velocity per time step which meant exploration of new search areas

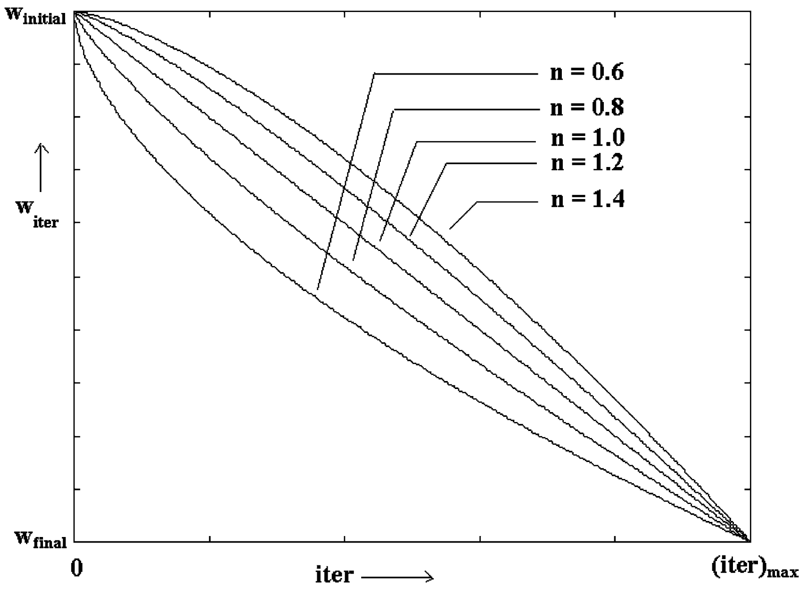

Fig. 2. Variations of inertia weights with iterations for different values of nonlinear modulation index (*n*).

in pursuit of a better solution. However, smaller inertia weight meant less variation in velocity to provide slower updating for fine tuning a local search. It was inferred that the system should start with a high inertia weight for coarse global exploration and the inertia weight should linearly decrease to facilitate finer local explorations in later iterations [5,6]. This should help the system to approach the optimum of the fitness function quickly. The present paper proposes a new nonlinear function modulated inertia weight adaptation with time for improved performance of PSO algorithm. This dynamic adaptation for PSO algorithm proposes to update the velocity relation in the *d*th dimension according to the following relation:

$$v_{i\,d\,\mathrm{new}} = w_{\mathrm{iter}}(v_{i\,d\,\mathrm{old}}) + \varphi_i(p_{id} - x_{id}) + \varphi_g(p_{gd} - x_{id}) \tag{4}$$

with position updating and velocity clamping rules kept unchanged as in (1b)–(1f). The stochastic weighting factors φ_i and φ_g are also kept unchanged. The important modification is the determination of the inertia weight w_{iter} as a nonlinear function of the present iteration number (iter) at each time step. The proposed adaptation of w_{iter} is given as:

$$w_{\mathrm{iter}} = f(\mathrm{iter}) = \left\{ \frac{(\mathrm{iter}_{\mathrm{max}} - \mathrm{iter})^n}{(\mathrm{iter}_{\mathrm{max}})^n} \right\} (w_{\mathrm{initial}} - w_{\mathrm{final}}) + w_{\mathrm{final}}, \tag{5}$$

where w_{initial} is the initial inertia weight at the start of a given run, w_{final} the final inertia weight at the end of a given run, when iter $= (\mathrm{iter})_{\mathrm{max}}$, $(\mathrm{iter})_{\mathrm{max}}$ the maximum number of iterations in a given run, iter the iteration number at the present time step, w_{iter} the inertia weight at the present time step and *n* the nonlinear modulation index.

Fig. 2 shows typical inertia weight variations with iterations for different values of *n*, on either side of unity. With $n = 1$, the system becomes a special case of linearly adaptive inertia weight with time, as proposed by Shi and Eberhart.

The system starts with a high initial inertia weight (w_{initial}) which should allow it to explore new search areas aggressively and then decreases it gradually according to (5) following different paths for different values of n to reach w_{final} at iter $=$ (iter)$_{\max}$. The proposed algorithm also attempts to derive a reasonable set of choice for the free parameters of any given system i.e. $\{w_{\text{initial}}, w_{\text{final}}, n\}$ on the basis of a fixed (iter)$_{\max}$. The objective is to arrive at an attractive solution for any given problem with the known, fixed free parameters applying our proposed PSO variation which should require less computational burden and time compared to trial and error approaches. However, a user of a given problem may attempt to arrive at an even improved performance by further fine tuning the free parameters.

4. Behavior of the proposed PSO model

To illustrate the behavior of our proposed PSO model, vis-à-vis classical PSO models, we have implemented it for the famous Sphere function which with any arbitrary n-dimensions has a single minimum:

$$\text{SM}(\mathbf{x}) = \sum_{d=1}^{n} x_d^2. \tag{6}$$

This function is also known as De Jong's f1 function and its global minimum is $\text{SM} = 0$ at $\mathbf{x} = (0, 0, 0, \ldots, 0)$. For our purpose of analysis we have considered a 6th order Sphere model i.e. $n = 6$ and the initial search space is limited to $x_d \in [-5.12, 5.12]$. We have carried out our experimentation for this function with 20 particles comprising the particle swarm population and maximum number of iterations in each run was assigned a value of 2000. In each simulation, the algorithm was implemented for 100 runs and the results reported are on the basis of the average performance obtained in each simulation, over 100 runs. The maximum allowable velocity in each dimension is taken as $v_{d\max} = (x_{d\max} - x_{d\min})$ i.e. under the assumption that any particle can have a maximum excursion of the full range available for that particle in a given dimension, in a single time step. The choice of 100 runs for each simulation and to present the average results is based on the concept that in each of these 100 runs, φ_i and φ_g will be generated with different random seeds in each of the (iter)$_{\max}$ iterations and the algorithm will provide true stochastic flavor in component II and component III throughout the experiment.

We start our experimentation on the Sphere model with $n = 1$, fixed initial inertia weight (w_{initial}) and corresponding variations of the slope of linear adaptations in w. As we are considering a special case of (5) with $n = 1$, the slope of w is given as:

$$m = \frac{w_{\text{initial}} - w_{\text{final}}}{(\text{iter})_{\max}}. \tag{7}$$

For the same w_{initial} and (iter)$_{\max}$ (=2000), choice of different 'm' will indicate choice of different w_{final} in each simulation. Table 1 will present these results for our proposed model. In each case, number of evaluations (evals) indicate the average number of iterations and the percentage of success of the test (%OK) indicates the average percent success taken over 100 runs to arrive at the global optimum. In each experiment the population size comprises of 20 particles. A test is considered to be 100% successful if the following relation holds good:

$$|f^{\alpha} - f^*| < \varepsilon_1 \times f^* + \varepsilon_2, \tag{8}$$

Table 1
Comparison of the iterations taken by the PSO model to achieve the desired minimum for the Sphere function with $n = 1$, for varying sets of $\{w_{\text{initial}}, m\}$

Maximum/starting weight (w_{initial})	Slope ($\times 10^{-4}$) (m)	%OK (%)	Eval (iter)
0.7	−2.5	100	861
0.6	−2.5	100	627
	−2.5	100	476
0.5	−3.0	100	459
	−3.5	100	448
	−4.5	100	352
0.4	−3.5	100	362
	−3.0	100	370
	−2.5	100	376
	−4.0	100	318
0.3	−3.0	100	316
	−2.5	100	316
	−2.0	100	318
	−3.5	100	310
0.2	−2.5	100	295
	−2.0	100	317
	−1.5	100	320
	−3.0	100	331
0.1	−2.0	100	322
	−1.5	100	325
	−1.0	100	304

where, f^* is the global optimum of the objective function under consideration, f^α the optimum of the objective function obtained by the algorithm under consideration, and $\varepsilon_1 = \varepsilon_2$ accuracy coefficients $= 10^{-4}$ (assumed in this literature).

One can see that in each case the algorithm converges and the testing was found 100% successful. The results demonstrate that a higher choice of w_{initial} i.e. $w_{\text{initial}} = 0.4 - 0.7$ consumes more iterations to give the required solution. Hence, experimentation with different values of *m* for same w_{initial} is avoided here. These experiments are conducted for smaller values of $w_{\text{initial}} (0.1 - 0.3)$ where encouraging results are obtained for any *m*. Different values of *m* are implemented to fine tune the best result for a given set of $\{w_{\text{initial}}, m\}$ which in turn determines the best set of $\{w_{\text{initial}}, w_{\text{final}}\}$ for a given $(\text{iter})_{\max}$. Table 1 demonstrates that the best results are obtained with $w_{\text{initial}} = 0.2$ and $m = -2.5 \times 10^{-4}$ when average number of iterations taken to arrive at the solution gets its minimum value of 295. Now w_{final} can be calculated for this special case of $n = 1$ as:

$$w_{\text{final}} = w_{\text{initial}} + m \times (\text{iter})_{\max}, \tag{9}$$

Hence w_{final} is calculated as -0.3. Once the best set of $\{w_{\text{initial}}, m\}$ i.e. $\{w_{\text{initial}}, w_{\text{final}}\}$ is obtained for the linear case of $n = 1$, we try to obtain better performance with the nonlinear case of $n \neq 1$ having w_{initial} and w_{final} kept fixed to their values obtained for the linear case. Table 2 shows the performance evaluation of the resultant system with different values of *n*, keeping $w_{\text{initial}} = 0.2$, $w_{\text{final}} = -0.3$ and $(\text{iter})_{\max} = 2000$. The objective is to find the best mathematical fitting among different possible variations

Table 2
Comparison of the iterations required by the PSO model to achieve the desired minimum for the Sphere function with varying n (fixed parameters: $w_{\text{initial}} = 0.2$, $w_{\text{final}} = -0.3$, $m = -2.5 \times 10^{-4}$ and $(\text{iter})_{\max} = 2000$)

Modulation index (n)	%OK (%)	Eval (iter)
0.1	100	299
0.3	100	299
0.5	100	299
0.8	100	296
1.0	100	295
1.2	100	286
1.4	100	289
1.5	100	294
1.6	100	300
2.0	100	304

of w, as plotted previously in Fig. 2. From Table 2, we can observe that the best results are obtained with $n = 1.2$ when 'eval' reaches its minimum value of 286. Thus determination of a suitable nonlinear variation of w with iterations helped us to improve the resultant performance. We propose to keep these three parameters fixed for different problems i.e. $w_{\text{initial}} = 0.2,\ \ m = -2.5 \times 10^{-4}$ and $n = 1.2$ and observe the resultant performance. w_{final} for each problem can be determined applying (8) for the special case of $n = 1$ when maximum number of iterations ($(\text{iter})_{\max}$) are specified. Once w_{initial} and w_{final} are obtained for the linear case, they remain unchanged for the nonlinear cases also, for a given problem.

The possible reason for $n = 1.2$ as an encouraging choice for the exponent in (5) can be given as below. As opposed to Shi and Eberharts' linear model, our model with $n = 1.2$ chooses a higher value of w during the early iterations which facilitates to take larger steps in the solution space. This helps the algorithm to search the solution space more aggressively to look for "better areas". However one cannot increase the value of n indefinitely much beyond unity, as this will surely cause large oscillations in the performance of the system and hence it will be very difficult to control it. A choice of $n = 1.2$ could also ensure the fact that during the later iterations w was decreased more rapidly than the linear case, which was very suitable for the algorithm to determine the optimal region among the already discovered promising sub optimal regions. If $n > 1.2$ is chosen, then even more rapid decrease in the value of w during later iterations may not be helpful because the algorithm would not have determined the promising sub optimal regions during the early iterations effectively, due to the excessive oscillations. Obviously the value of 1.2 is a typical one and it can be suitably determined for each case individually. However, the reasoning will remain valid for each such suitable optimum value of n determined.

5. Performance evaluation

The performance of the proposed PSO model is tested for a number of analytical benchmark functions which have been extensively used to compare both PSO-type and non-PSO-type metaheuristic algorithms. Table 3 reports the results obtained for 12 such benchmark problems for which the proposed PSO model has been applied with the same parameter set of $\{w_{\text{initial}} = 0.2,\ \ m = -2.5 \times 10^{-4}$ and $n = 1.2\}$ and the results are compared with Shi and Eberhart's model of linearly adapting inertia weights with a parameter set of

Table 3
Iterations required by the proposed function modulated PSO model (i.e. $n = 1.2$) vis-à-vis linearly modulated PSO model (i.e. $n = 1$), for a set of benchmark problems

Function	PSO with $n = 1$	PSO with $n = 1.2$
SM	295	286
R_2	349	349
B_2	67	67
Boha-1	21	21
Boha-2	38	36
Boha-3	25	24
MG	47	46
GP	79	74
DJ	22	22
Z_2	17	16
Z_5	81	81
Z_{10}	404	373

Table 4
Iterations required by the proposed PSO model vis-à-vis other continuous optimization algorithms to achieve optima for the benchmark problems given in Table 2

Function	Proposed PSO	SCGA [11]	RCGA [12]	CGA [13]	DSSA [14]	ECTS [15]	DE [16]
SM	286	—	—	750	—	338	392
R_2	349	222	596	960	306	480	615
B_2	67	—	—	—	—	—	—
Boha-1	21	460 (99%)	—	430	252	—	—
Boha-2	36	471 (99%)	493	—	—	—	—
Boha-3	28	468	—	—	—	—	—
MG	46	—	—	—	—	—	—
GP	74	191	270	410	261	—	—
DJ	22	187	395	750	273	—	—
Z_2	16	170	437	620	186	—	—
Z_5	81	998	1115	1350	914	—	—
Z_{10}	373	1829	2190	6991	12501	—	—

$\{w_{\text{initial}} = 0.2,\ m = -2.5 \times 10^{-4}$ and $n = 1\}$ for each benchmark problem. The results are obtained with 20 particles employed in each iteration and 2000 iterations in each run. For a large number of problems e.g. Sphere model, Bohachevsky and Zakharov functions, Goldstein and Price function etc., our model shows faster convergence in relatively smaller number of iterations. For remaining benchmark problems e.g. two-dimensional Rosenbrock, B_2, De Jong's function etc. both models give similar performance. In general, significant improvement in performance is obtained for higher dimensional problems e.g. six-dimensional Sphere model or the 10-dimensional Zakharaov function.

Table 4 compares the performance of our proposed PSO model with eight other popular (non-PSO) metaheuristic algorithms. The results in each case indicate in how many iterations the convergence of the solution or success was met, for a given benchmark problem. These metaheuristic algorithms were

Table 5
Mean best fitness function values over 100 runs for the proposed PSO model with varying population size, employed for a popular set of benchmark functions

Function	Population size				
	20	40	60	80	100
De Jong's f1	0.046453	0			
De Jong's f2	0				
De Jong's f4—no noise	0.010359	0.000094	0		
Shaffer's f6	0.000004	0			
Griewank function	0.158316	0.033107	0.021448	0.018524	0.015275
Rosenbrock function	179.613642	81.799508	72.278469	54.454777	53.085024
Rastrigin function	61.425623	43.881103	37.635133	34.963201	29.769177

chosen from a wide cross section of different popular strategies possible e.g. genetic algorithm, simulated annealing, tabu search etc. Those cells which are marked as '$-$' indicate that for that benchmark problem the corresponding algorithm was not tested. The results in Table 4 indicate superiority in terms of speed of convergence for our proposed PSO model for most of the benchmark functions, without sacrificing accuracy.

Armed with this success of testing our algorithm vis-à-vis other non-PSO models, we then conducted the performance comparison vis-à-vis other PSO models for another set of 7 benchmark problems considered in [7]. These benchmark problems are relatively more complex, mainly due to larger dimensional size. Out of the 7 functions considered 5 were with dimensions of $n = 30$. The performance was tested with 2000 maximum number of iterations in each run ($(\text{iter})_{\max}$) and 100 runs in each experiment. Table 5 presents mean best fitness function values obtained over all the runs for each function. These functions are so considered that the global optimum for each function is 0.0. To demonstrate the efficacy and usefulness of employing our model with fixed set of free parameters i.e. $\{w_{\text{initial}} = 0.2,\ m = -2.5 \times 10^{-4}$ and $n = 1.2\}$ we employed these same parameters for each function. The results are reported using 20, 40, 60, 80 or 100 particles as necessary to obtain the mean best result. Once the results obtained with 20 particles are not found satisfactory, we have opted for 40 particles. If the results are not even satisfactory for 40 particles, we have chosen 60 particles and so on. Only the more complex benchmark functions like Griewank, Rosenbrock and Rastrigin functions required 100 particles for very high problem dimension ($n = 30$).

Table 6 presents the comparisons of the mean best result obtained using our algorithm vis-à-vis other popular PSO models [7] and Angeline's evolutionary method [4]. Fig. 3 presents representative variation of fitness function values with iterations for three higher dimensional problems ($n = 30$). Considering performances for all functions, one can conclude that among the possible solutions available till now, Type 1", Type 1 and E&S algorithms give almost comparable performance. In fact, among them it seems that E&S algorithm has evolved as the best candidate. It can be easily observed that our algorithm arrives at the global optimum value of 0.0 for each of the f1, f2 and f4 functions and performs at par with Type 1", Type 1 and E&S algorithms. For the f6 algorithm our model could beat all other competing algorithms as no other algorithm could arrive at 0.0 but our model could. Performances of our proposed algorithm for Griewank and Rosenbrock functions show slightly poor results compared to Type 1", Type 1 and

Table 6
Mean best fitness function values attained by the proposed PSO model vis-à-vis other particle swarm and Angeline's evolutionary algorithm for the popular set of benchmark functions in Table 5

Function	$V_{\max} = 2$	$V_{\max} = 4$	Type 1"	Type 1	Exp. Version	E& S	Angeline	Proposed PSO model
De Jong's f1	15.577775	59.301901	0	0	0	0	9.8808	0
De Jong's f2	0.000500	0.001326	0	0	0	0	—	0
De Jong's f4—no noise	271.107996	4349.13751	0	0	0	0	—	0
Shaffer's f6	0.000464	0.000247	0.001459	0.002915	29.17301	0.00016	—	0
Griewank function	0.562339	0.968623	0.003944	0.008614	0.038923	0.00209	0.4033	0.015275
Rosenbrock function	2770.88259	37111.7070	50.193877	39.118488	47.75395	50.7981	1610.36	53.085024
Rastrigin function	223.834812	299.771716	82.95618	81.689550	63.22260	57.1941	46.4689	29.76918

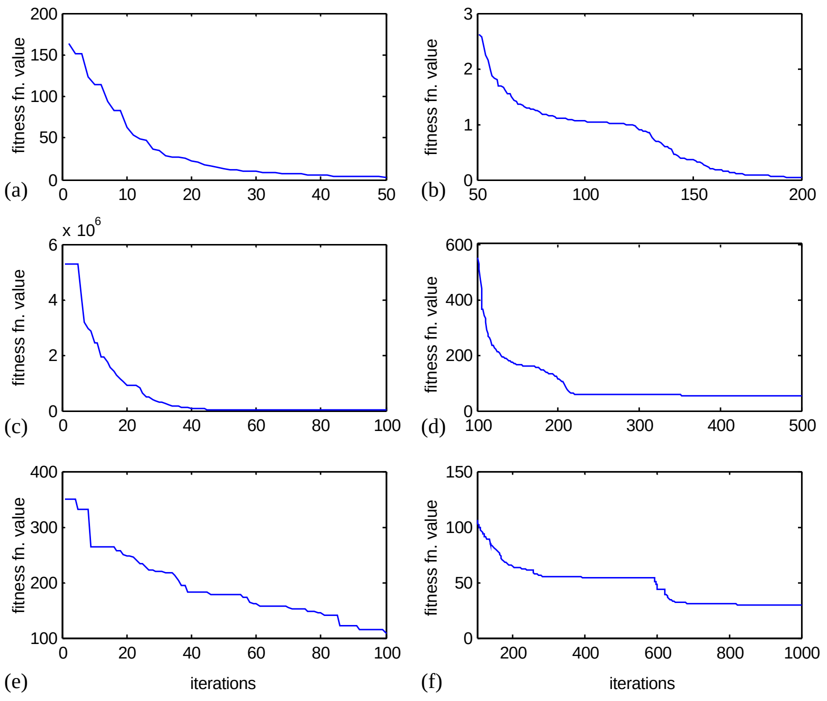

Fig. 3. Fitness function values plotted against iterations for higher dimensional ($n = 30$) problems: (a), (b) for Griewank function; (c), (d) for Rosenbrock function, (e), (f) for Rastrigin function.

E&S algorithms. Again for the Rastrigin function, our model performed better than all other algorithms comfortably. So for most of these functions, our proposed model performed either better than or at least at par with most of the popular PSO models. We have already demonstrated how our proposed model could obtain superior results for all but one benchmark functions in Table 4, when compared to popular non-PSO models. Hence, it can be concluded that the proposed algorithm can be considered as one of the most efficient PSO algorithms proposed so far. Another important point under consideration should be that all the results in our proposed algorithm were reported using the same set of free parameters $\{w_{\text{initial}}, m, n\}$.

The significantly encouraging results obtained are expected to tempt many users to relieve themselves of any additional burden of employing trial and error-based approaches for each and every problem to determine the best set of free parameters. In fact, if the user is not happy with the results of a given optimization problem employing our algorithm with our proposed free parameters, he/she can always exercise the option of manually tuning these parameters to obtain a better solution.

6. Conclusion

A new variant of PSO is proposed here which employs nonlinear variation of inertia weight. This nonlinear variation is adopted so that the proposed algorithm can employ aggressive, coarse tuning during initial iterations to achieve better search of the solution space to quickly arrive near optimum solution and to gradually employ mild, fine tuning during later iterations so that the optimum solution can be approached with better accuracy. Different nonlinear variations are tested with different modulation index, n for a classical, benchmark problem of Sphere model in six-dimensions. Finally the algorithm proposes a fixed set of free parameters $\{w_{\text{initial}}, m, n\}$ for its reasonably satisfactory operation which saves a user of the time consuming, tedious operation of finding these parameters best suited for each and every optimization problem separately. The performances of the proposed PSO algorithm, employing the fixed parameter set for each problem domain, have been tested for a large number of benchmark functions and they have been compared with a number of popular PSO algorithms and other non-PSO algorithms. The results are found to be mostly encouraging.

One of the important factors in successful implementation of the proposed algorithm is the proper choice of exponent n. Although $n = 1.2$ has shown encouraging results for several benchmark problems, there can be other functions where this choice may not be the best one. Several experimentations within the perview of this study showed that a choice of n within [0.9-1.3] is normally satisfactory. Hence in our opinion, if an user wants to verify whether $n = 1.2$ is the best solution for his/her problem, he/she can vary n from 0.9 to 1.3 in steps of 0.1 and can settle for the best solution. An even superior solution can be the employment of an online, dynamical determination of n after each iteration, rather than choosing the best static value. This adaptive determination of n can be potentially carried out on the basis of incremental variation of fitness function value, after each iteration. The authors wish to pursue this development as an immediate direction for the future research.

References

[1] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. In: Proceedings of ECAL’91—European Conference on Artificial Life. Paris, France: Elsevier Publishing, 1998. p. 134–42.

[2] Eberhart RC, Kennedy J. A new optimiser using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Piscataway, NJ, Nagoya, Japan: IEEE Service Center; 1995. p. 39–43.

[3] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. vol. IV. Piscataway, NJ: IEEE Service Center; 1995. p. 1942–8.

[4] Angeline PJ. Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. Evolutionary Programming VII, Lecture Notes in Computer Science, vol. 1447. Berlin: Springer; 1998. p. 601–10.

[5] Shi Y, Eberhart RC. A modified particle swarm optimiser. In: Proceedings of the IEEE International Conference on Evolutionary Computation. Piscataway NJ: IEEE Press, 1998. p. 69–73.

[6] Shi Y, Eberhart RC. Parameter selection in particle swarm optimization. In: Evolutionary Programming VII. Lecture Notes in Computer Science, vol. 1447. Berlin: Springer; 1998. p. 591–600.

[7] Clerc M, Kennedy J. The particle Swarm—explosion, stability and convergence in a multidimensional complex space. IEEE Trans Evol Comput 2002;6(1):58–73.
[8] Eberhart RC, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation. Washington, DC, 2000. p. 84–8.
[9] Lovbjerg M, Rasmussen TK, Krink T. Hybrid particle swarm optimiser with breeding and subpopulations. In: Proceedings of the Third Genetic and Evolutionary Computation Congress, 2001.
[10] Coelho JP, Oliviera PB, de M Cunha Greenhouse air temperature control using the particle swarm optimization algorithm. In: 15th Triennial World Congress. IFAC. Barcelona, Spain, 2002.
[11] Hedar A, Fukushima M. Simplex coding genetic algorithm for the global optimization of nonlinear functions. Technical Report, Department of Applied Mathematics and Physics, Kyoto University, Japan, 2002.
[12] Bessaou M, Siarry P. A genetic algorithm with real-value coding to optimize multimodal continuous functions. Struct Multidisc Optim 2001;23:63–74.
[13] Chelouah R, Siarry P. A continuous genetic algorithm designed for the global optimization. J Heuristics 2000;6:191–213.
[14] Hedar A, Fukushima M. Hybrid simulated annealing and direct search method for nonlinear global optimization. Technical Report 2001-013, Department of Applied Mathematics and Physics, Kyoto University, Japan, 2001.
[15] Chelouah R, Siarry P. Tabu search applied to global optimization. Euro J Oper Res 2000;123:256–70.
[16] Storn R, Price K. Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR95-012, International computer science institute, Berkeley, CA, 1995.