OPSEARCH 46(1):3–24

THEORY AND METHODOLOGY

Hybridization of particle swarm optimization with quadratic approximation

Kusum Deep, Jagdish Chand Bansal

Department of Mathematics
Indian Institute of Technology Roorkee
Roorkee - 247667 India
kusumfma@iitr.ernet.in, jcbansal@gmail.com

Abstract

Particle swarm optimization (PSO) has been extensively used in recent years for the optimization of nonlinear optimization problems. Two of the most popular variants of PSO are PSO-W (PSO with inertia weight) and PSO-C (PSO with constriction factor). Efforts have also been made to hybridize PSO with other methodologies to improve its performance. In this paper we present the hybridization of PSO with quadratic approximation operator (QA). The hybridization is performed by splitting the whole swarm into two subswarms in such a way that the PSO operators are applied on one subswarm, whereas the QA operator is applied on the other subswarm, ensuring that both subswarms are updated using the global best particle of the entire swarm. Based on this concept, two algorithms, namely qPSO-W and qPSO-C have been developed and their performance is evaluated with respect to PSO-W and PSO-C on the basis of 15 benchmark test problems and 3 real life problems taken from literature. The numerical and graphical results are a proof that the hybridized approach is a definite improvement in terms of efficiency, reliability and robustness.

Keywords

Particle swarm optimization, Quadratic approximation, Nonlinear optimization, Hybridization

1. Introduction

Particle swarm optimization (PSO) technique is considered as one of the modern heuristic algorithms for optimization introduced by James Kennedy and Eberhart in 1995. It is based on the social behavior metaphor [1]. It is a population-based optimization technique, which is sometimes regarded as an alternative tool to genetic algorithm (GAs) and other evolutionary

Paper accepted on February 17, 2009

algorithms (EAs) and gained a lot of attention in the recent years. As compared to EAs, PSO is a stochastic search technique with reduced memory requirement, computationally effective and easier to implement. Also PSO has a more global searching ability at the beginning of the run and has greater local search ability near the end of the run [2].

The "No free Lunch Theorem" by Wolpert and Macready [3], shows that there is no single method which can solve all the problems optimally. As a result research on hybrid optimization algorithms has gained momentum over the past few years. ''Hybridization" is an approach that combines the capabilities of two strong concepts in such a way that the good traits of both are adopted. Hybrid strategy is generally regarded as an efficient strategy (requiring fewer evaluations) which is also generally more effective (identifying higher quality solutions) for solving complex optimization problems. A number of approaches of hybridization of PSO have been recently reported in the literature.

A large number of PSO variants have been developed by combining certain aspects from evolutionary computation (EC) and ant colony optimization (ACO) with the PSO. In the case of EC hybrids, ideas have been borrowed from specific EC paradigms, including GA, evolutionary programming (EP), evolutionary strategies (ES), differential evolution (DE) and cartesian genetic programming (CGP).

Another trend is to merge or combine the PSO with the other techniques, especially the EC techniques. Evolutionary operators like selection, crossover and mutation have been incorporated into the PSO. By applying the selection operation in PSO, the particles with the best performance are copied into the next generation; so that the PSO can always keep the best performing particles [4]. By incorporating crossover operation, information can be swapped between two individuals to have the ability to "fly" to the new search area [5]. Among the three evolutionary operators, the mutation operators are the most commonly applied evolutionary operators in PSO. The purpose of applying mutation to PSO is to increase the diversity of the population and to provide the ability to the PSO to escape local minima [6–11]. Juang [12] also incorporated mutation alongside crossover and elitism. This process imitates the natural phenomenon of maturation and outperformed both PSO and GA in the study. Another approach is to prevent particles from moving too close to each other so that the diversity can be maintained and therefore escape from being trapped into local minima is achieved. In [7], the particles are relocated when they are too close to each other. Blackwell and Bentley [13]

and Krink et al. [9], designed collision-avoiding mechanisms to prevent particles from colliding with each other and increase the diversity of the swarm.

Besides incorporating evolutionary operations into PSO, different approaches to combine PSO with the evolutionary algorithms have also been reported. Robinson et al. [14] obtained better results by applying PSO first followed by GA in their profiled corrugated horn antenna optimization problem. Jian and Chen [15] introduced a PSO hybrid with the GA recombination operator and dynamic linkage discovery to optimize difficult real number optimization problems. Dynamic linkage discovery is a technique based on the notion that if the links between the basic building blocks of the objective function can be discovered, then optimization of that problem can be improved. In [16], genetical swarm optimization is presented by combining PSO and GA. In each iteration the population is divided into two parts and these are evolved with the two techniques respectively. They are then recombined in the updated population, which is again divided randomly into two parts in the next iteration for another run of genetic or particle swarm operators. Poli et al. [17–18] proposed a hybrid PSO based on genetic programming (GP). GP is used to evolve new laws for the control of particles' movement for specific classes of problems. In [19], either PSO algorithm, GA or hill-climbing search algorithm is applied to a different subpopulation of individuals in which each individual is dynamically assigned according to some pre-designed rules. In [20], DE is combined with PSO. Particles fly according to position update equation, but occasionally DE is applied to replace one poorly performed particle with a better one while retaining its velocity. Zhang and Xie [21], in their DEPSO use DE and canonical PSO operators in alternate generations. The hybrid was found to be successful for some functions, but not all, with results indicating that DEPSO improves on PSO in problems with higher dimensionality. In [22], ACO is combined with PSO. A list of best positions found so far is recorded and the neighborhood best is randomly selected from the list instead of the current neighborhood best.

Non-evolutionary techniques have also been incorporated into PSO. In [23], a cooperative particle swarm optimizer (CPSO) is implemented. The CPSO employs cooperative behavior to significantly improve the performance of the original PSO algorithm using multiple swarms to optimize different components of the solution vector cooperatively. In [5], the population of particles is divided into subpopulations which breed within their own subpopulation or with a member of another with

some probability so that the diversity of the population can be increased. Parsopoulos and Vrahatis [24], incorporate deflection and stretching techniques as well as a repulsion technique are into the original PSO to avoid particles moving toward the already found global minima so that the PSO can have more chances to find as many global minima as possible. In [25], a "dissipative particle swarm" is designed by adding negative entropy into the PSO to discourage premature convergence. Liu and Abraham [26] have hybridized a turbulent PSO (TPSO) with a fuzzy logic controller to produce a fuzzy adaptive TPSO (FATPSO). The TPSO uses the principle that PSO's premature convergence is caused by particles stagnating about a suboptimal location. Arumugam et al. [2] have used extrapolation technique to update the particles' best position along with PSO (ePSO) for solving optimization problems. A detailed review on the hybrid PSO can be found in [27].

More and more hybrid algorithms are being designed and implemented with the hope of further improving their performance. In this paper we propose a novel, effective and efficient algorithm based on the hybridization of PSO and quadratic approximation operator, namely quadratic approximation particle swarm optimization (qPSO).

In the proposed hybrid algorithm, quadratic approximation operator is used to update a part of the swarm while the remaining of the swarm is updated by PSO as usual. In each iteration, a predetermined number of particles of the swarm are updated using the minima of quadratic surface passing through the global best and two random particles chosen from the entire swarm while others use PSO to update their positions. qPSO and standard PSO are simulated to test their efficacy by solving a set of 15 benchmark problems and the results are analyzed through various statistical parameters and performance index. Three real life applications are then considered to test the robustness of the proposed method.

The remaining paper is organized as follows. Section 2 describes standard PSO and some of its variations. The proposed qPSO is explained in Section 3. In Section 4, the testing of the proposed method through a set of 15 benchmark problems is carried out and the simulation results are compared with those obtained via PSO and some of its variants. Three real life applications are solved in Section 5 and the experimental results are presented. Finally, in Section 6, the conclusion is drawn based on the analysis.

2. Standard PSO and some of its variants

The idea behind PSO is based on the simulation of the social behavior of bird flock and fish schools. PSO is a swarm intelligence method for global

optimization problems. It differs from well-known evolutionary algorithms as in evolutionary algorithms a population of potential solutions is used to probe the search space, but no operators, inspired by evolution procedures, are applied on the population to generate new promising solutions. Instead in PSO, each individual, namely *particle*, of the population, called *swarm*, adjusts its trajectory towards its own previous best position (pbest), and towards the previous best position of any member of its topological neighborhood (gbest). Two variants of the PSO have been developed, one with a global neighborhood and the other with a local neighborhood. According to the global variant, each particle moves towards its best previous position and towards the best particle in the whole swarm. On the other hand, in the local variant, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood.

Working of PSO may be briefly described as under:

Suppose the search space is *D*-dimensional, then the *i-th* particle of the swarm can be represented by a *D*-dimensional vector, $X_i = (x_{i1}, x_{i2}, ...,x_{iD})^T$. The velocity (position change) of this particle can be represented by another *D*-dimensional vector $V_i = (v_{i1}, v_{i2},...,v_{iD})^T$. The best previously visited position of the *i-th* particle is denoted as $P_i = (p_{i1}, p_{i2}, ...,p_{iD})^T$. Defining *g* as the index of the best particle in the swarm, the swarm is manipulated according to the following two equations:

Velocity update equation:

$$v_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{1}$$

Position update equation:

$$x_{id} = x_{id} + v_{id} \tag{2}$$

where $d = 1, 2... D; i = 1, 2... S$, where *S* is the size of the swarm; c_1 and c_2 are constants, called cognitive and social scaling parameters respectively (usually, $c_1 = c_2$; r_1, r_2 are random numbers, uniformly distributed in *[0, 1]*). Equations (1) and (2) define the initial version of PSO algorithm. A constant, *Vmax*, is used to arbitrarily limit the velocities of the particles and improve the resolution of the search.

The pseudo code of PSO is shown below:

Algorithm PSO:
For t= 1 to the max. bound of the number on iterations,
For i=1 to the swarm size,

For d=1 to the problem dimensionality,
Apply the velocity update equation:
Update Position
End- for-d;
Compute fitness of updated position;
If needed, update historical information for Pi and Pg;
End-for-i;
Terminate if Pg meets problem requirements;
End-for-t;
End algorithm.

The maximum velocity Vmax, serve as a constraint to control the global exploration ability of particle swarm. A larger Vmax facilitates global exploration, while a smaller Vmax encourages local exploitation. The concept of an inertia weight was also developed to better control exploration and exploitation. The motivation was to be able to eliminate the need for Vmax. The inclusion of an inertia weight in the particle swarm optimization algorithm was first reported in the literature in 1998 [28–29].

After some experience with the inertia weight, it was found that although the maximum velocity factor, Vmax, couldn't always be eliminated, the particle swarm algorithm works well if Vmax is set to the value of the dynamic range of each variable (on each dimension). The resulting velocity update equation becomes:

$$v_{id} = w * v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id})$$

Eberhart and Shi [30] indicates that the optimal strategy is to initially set w to 0.9 and reduce it linearly to 0.4, allowing initial exploration followed by acceleration toward an improved global optimum.

In 1999, Clerc has introduced a constriction factor, χ, which improves PSO's ability to constrain and control velocities [31]. χ is computed as:

$$\chi = \frac{2}{\left|2 - \phi - \sqrt{\phi(\phi - 4)}\right|} \tag{3}$$

where $\phi = c_1 + c_2, \phi > 4$, and the velocity update equation is then

$$v_{id} = \chi * \left(v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id})\right)$$

Eberhart and Shi [30] found that χ, combined with constraints on *Vmax*, significantly improved the performance of PSO.

It was observed that PSO usually suffers from premature convergence, tending to get stuck in local optima, low solution precision and so on. In order to overcome these shortcomings and get better results, numerous improvements to PSO have been proposed. In this paper we propose another hybrid version of PSO which uses quadratic approximation operator.

3. Proposed quadratic approximation particle swarm optimization (qPSO)

3.1 Motivation

Deep and Das [32], hybridize a binary GA by incorporating QA operator as an additional operator for local search. This showed a substantial improvement in the performance of GA. PSO has the efficiency to solve a wide variety of problems with a larger percentage of success. Mohan and Shankar [33] proved that random search technique (RST) which uses QA operator provides fast convergence rate but once stuck in a local optima, it is generally difficult to come out of it. Perhaps social knowledge concept of PSO could help RST in coming out of the local optima. As compared to GAs, the PSO has much more profound intelligent background and could be performed more easily. These two facts motivated us to hybridize PSO and QA with the expectation of faster convergence (from QA) and improved results (from PSO).

3.2 Quadratic approximation operator

QA is an operator which determines the point of minima of the quadratic hyper surface passing through three points in a D-dimensional space. It works as follows:

1. Select the particle R_1, with the best objective function value. Choose two random particles R_2 and R_3 such that out of R_1, R_2 and R_3, at least two are distinct.
2. Find the point of minima R^* of the quadratic surface passing through R_1, R_2 and R_3, where

$$R^* = 0.5\left(\frac{\left(R_2^2 - R_2^2\right)f\left(R_1\right) + \left(R_3^2 - R_1^2\right)f\left(R_2\right) + \left(R_1^2 - R_2^2\right)f\left(R_3\right)}{\left(R_2 - R_3\right)f\left(R_1\right) + \left(R_3 - R_1\right)f\left(R_2\right) + \left(R_1 - R_2\right)f\left(R_3\right)}\right) \tag{4}$$

where $f(R_1)$, $f(R_2)$ and $f(R_3)$ are the objective function values at R_1, R_2 and R_3 respectively. The calculations are to be done component-wise using

(4) to obtain R^*. QA operator has been successfully applied to solve application problems as well [34].

3.3 The process of hybridization

In each iteration, the whole swarm S is divided into two subswarms (say S_1 and S_2). From one generation to the next generation, S_1 is evolved using PSO, whereas S_2 is evolved using QA. Figure 1 shows the idea that stands behind qPSO and the way to integrate the two techniques. qPSO consists in a strong co-operation of QA and PSO, since it maintains the integration of the two techniques for the entire run.

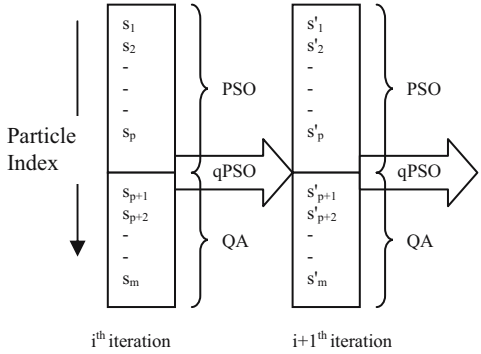

Fig. 1. Transition from i^{th} iteration to $i+1^{th}$ iteration

It is possible to understand the key steps of the qPSO, in a better way through an intuitive flowchart shown in Fig. 2. In the figure, ITER stands for iteration number and R_1, R_2 and R_3 has same meaning as in Section 3.2. It should be noted that R_1 used in QA and gbest used in PSO both are the global best position of the entire swarm (let us call it GBEST) i.e R_1 = GBEST and gbest = GBEST. The strength of the qPSO lies in the facts that both PSO and QA use the GBEST simultaneously or in other words, subswarm S_1 and S_2 share their best positions with each other and for transition from one iteration to the next, both updating schemes use the entire swarm's information. However, in updating a particle's position by QA, no information about its current position is applied as in PSO but the presence of memory of the corresponding subswarm preserves the best performed particles. So in $i+1^{th}$ iteration QA will not produce worse solution than that in i^{th} iteration.

We consider two versions of PSO, namely PSO-W (PSO with time varying inertia weight) and PSO-C (PSO with constriction factor) in order to compare the performance of qPSO. Thus two versions of qPSO come into existence, qPSO-W (qPSO with time varying inertia weight) and qPSO-C (qPSO with constriction factor). Percentage of swarm to be updated by PSO or QA is an important parameter of qPSO. We call this parameter as *coefficient of hybridization (CH)*. *CH* is the percentage of swarm which is evolved using QA in each iteration. Thus, if CH = 0, then the algorithm is pure PSO (the whole swarm is updated by PSO operators), and if CH = 100 then the algorithm is pure QA (the whole swarm is updated by QA operator) while for 0<CH<100 the corresponding percentage of swarm is evolved by QA and the rest with PSO.

4. Testing with benchmark functions

From the standard set of benchmark problems available in the literature [32], 15 important problems have been selected to test the efficacy of the proposed methods. These problems are of continuous variables and have different degree of complexity and multimodality. The set of test functions includes unimodal and multimodal functions which are scalable (the problem size can be varied as per the user's choice). The problem size for all problems is taken to be 30. All the problems are of minimization type having minimum at 0.

4.1 Selection of parameters

In case of many algorithms values of some parameters have to be provided by the user. PSO also has some parameters. In literature, different values of these parameters are used. In this paper we use the parameter setting as suggested in [35–37]. We set swarm size S =50. The inertia weight w is set to reduce linearly from 0.8 to 0.4. Constriction coefficient χ is calculated from equation (3). For PSO-C the cognitive and social parameters c_1 and c_2 are 2.8 and 1.3, respectively; for PSO-W both are set to 2. All PSOs are global versions i.e. each particle flies through the search space. Maximum velocity Vmax set equal to 0.5*(Xmax-Xmin), where Xmax and Xmin are the upper and lower bounds of the decision variables. Two criteria are applied to terminate the simulation of the algorithms: reaching maximum number of iterations (which was set as 1000) and the second criterion was getting a minimum error (0.001 for this study). For fair comparison we used the parameters of qPSO-W and qPSO-C algorithms as in original PSO-W and

PSO-C (i.e. parameters for PSO-C and qPSO-C are same and for PSO-W and qPSO-W are same). One of the questions concerning the proposed method is what percentage of the swarm (i.e. coefficient of hybridization) should be updated using QA? After a few preliminary tests, the value of CH = 30% is adopted for our study which yielded the best results for considered test functions.

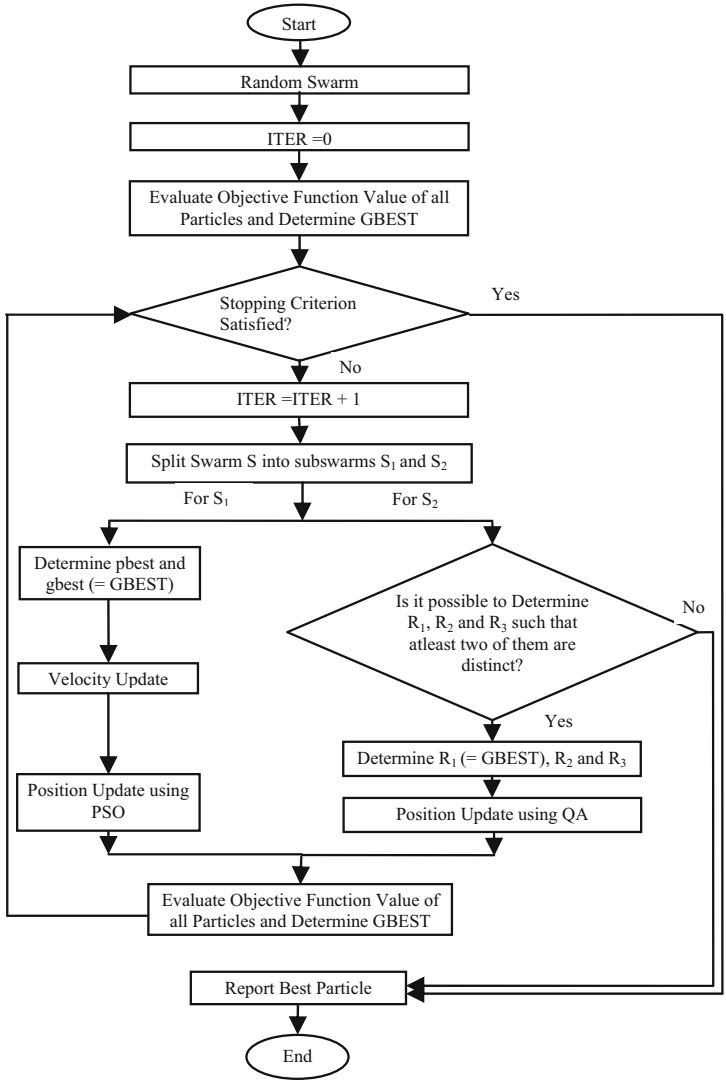

Fig. 2. Flowchart of qPSO process

4.2 Comparisons and discussions

This section focuses on the efficiency of qPSO as tested against 15 benchmark functions with 30 variables, which are taken from [32]. To avoid attributing the optimization results to the choice of a particular initial population and to conduct fair comparisons, we perform each test 100 times, starting from various randomly selected points in the hyper-rectangular search domain given in the literature and the results are recorded. The four PSOs (PSO-W, PSO-C, qPSO-W and qPSO-C) are implemented in C++. From the recorded simulated results statistical analyses are carried out and presented in Table 1. For each method the mean error, minimum error (Min Error), standard deviation (SD), success rate (SR) and the average number of function evaluations required to fulfill the termination condition (Mean Eval) are calculated from 100 simulated runs and are compared. A success was counted when the condition $f_{\min} - f_{\text{opt}} \leq 0.001$ was met, where $f_{\min}$ is the best solution found when an algorithm terminates and f_{opt} is the known global minimum of the problem.

The first goal of the analysis is to observe if the hybridization shows an improvement over the original PSOs for proposed two versions or not. From Table 1, it is clear that from the point of view of success rate qPSO-W is better than PSO-W in 9 problems, and worse in 1 problem while both perform same in 4 problems. Also on the same criteria qPSO-W is better than qPSO-C in 1 problem and worse in 6 problems while qPSO-W and qPSO-C both perform same in 7 problems. On the criteria of success rate qPSO-C is better than PSO-C in 5 problems while in remaining 9 problems both the methods have same performance. Clearly, qPSOs are better than their respective original versions and qPSO-C has the highest success rate than any method considered. Thus on the criteria of success rate qPSO-C stands first. If we consider mean number of function evaluations then qPSO-W is better than PSO-W in 9 problems, worse in 2 problems and has same performance as PSO-W in 3 problems. qPSO-C is better than PSO-C in 9 problems, worse in 2 problems and same in 3 problems. Also, if we compare hybridized PSOs then qPSO-C takes less number of functions evaluations to converge than qPSO-W in 10 problems, higher in 1 problem while same in 3 problems. From above discussion we can conclude that hybridization shows an improvement over the original PSOs for both versions.

In order to compare the consolidated performance of PSO-W, PSO-C, qPSO-W and the qPSO-C algorithm, the value of a performance index *PI* [38] is computed for these four algorithms. This index gives a

Table 1: Comparison of all PSO variations

Sr. No.	Test function	Algorithm	Performance				
			SR (%)	Mean eval	Mean error	Min error	SD
1.	Sphere (De Jong's f1)	PSO-W	100	14634	0.000932	0.000728	6.22E-05
		PSO-C	100	6908	0.000902	0.000533	8.86E-05
		qPSO-W	100	13526	0.000921	0.000741	6.64E-05
		qPSO-C	100	7137	0.000914	0.000653	7.78E-05
2.	Axis parallel hyper ellipsoidal	PSO-W	99	16338	1.6167	0.000553	5.125
		PSO-C	100	6202	0.000868	0.000573	9.33E-05
		qPSO-W	100	11098	0.000873	0.000259	0.000119
		qPSO-C	100	5826	0.000863	0.000546	0.0001
3.	Griewank	PSO-W	36	40224	0.010633	0.000693	0.012185
		PSO-C	52	29722	0.009344	0.000656	0.013928
		qPSO-W	46	37441	0.009474	0.000593	0.011201
		qPSO-C	61	22828	0.001098	0	0.011458
4.	Rosenbrock	PSO-W	0	50000	65.18	9.68558	45.9
		PSO-C	0	50000	34.7602	2.91574	28.6507
		qPSO-W	0	50000	49.4333	1.89108	43.2537
		qPSO-C	0	50000	28.0618	1.04074	22.6275
5.	Rastrigin	PSO-W	0	50000	70.1388	28.8581	23.6204
		PSO-C	0	50000	32.602	13.9294	9.25124
		qPSO-W	0	50000	53.5691	17.467	19.3696
		qPSO-C	0	50000	27.8087	11.9711	7.28571

Table 1 (cont.): Comparison of all PSO variations

6.	Ackley	PSO-W	100	30349	0.00096	0.00079	4.16E-05
		PSO-C	62	27468	0.472819	0.000784	0.633273
		qPSO-W	95	32034	0.439787	0.000841	2.79734
		qPSO-C	70	29960	0.342396	0.000175	0.573362
7.	Levy and Montalvo 1	PSO-W	70	25527	0.146521	0.000716	0.315224
		PSO-C	94	8338	0.007081	0.000612	0.024425
		qPSO-W	80	20789	0.046482	9.94E-05	0.124963
		qPSO-C	100	7145	0.000904	0.0003	9.31E-05
8.	Levy Montalvo 2	PSO-W	75	23200	0.067338	0.000672	0.387977
		PSO-C	83	13961	0.004699	0.000161	0.022203
		qPSO-W	76	23588	0.003726	2.28E-06	0.006102
		qPSO-C	100	9594	0.000783	2.28E-06	0.000278
9.	Ellipsoidal	PSO-W	4	49011	183.89	0.00083	225.551
		PSO-C	100	8668	0.00091	0.000693	7.65E-05
		qPSO-W	100	16845	0.000926	0.000722	5.96E-05
		qPSO-C	100	6117	0.000902	0.000567	8.57E-05
10.	Cosine mixture	PSO-W	47	34648	0.10684	0.000803	0.116119
		PSO-C	35	35289	0.148096	0.000697	0.138301
		qPSO-W	51	31571	0.09633	0.000197	0.103971
		qPSO-C	78	33587	0.037216	0	0.073387

Table 1 (cont.): Comparison of all PSO variations

11.	Exponential	PSO-W	99	9977	0.00485	0.000661	0.039058
		PSO-C	100	4543	0.00091	0.000737	7.00E-05
		qPSO-W	100	8814	0.000914	0.000623	7.53E-05
		qPSO-C	100	4121	0.000911	0.000615	7.72E-05
12.	Zakharov's	PSO-W	0	50000	64.2914	21.7452	21.344
		PSO-C	0	50000	1.4593	0.177401	1.03659
		qPSO-W	0	50000	2.21792	0.208495	3.13948
		qPSO-C	0	50000	0.048674	0.001707	0.056663
13.	Cigar	PSO-W	54	3997	46.0005	0.000709	49.8393
		PSO-C	100	14785	0.000904	0.000663	6.92E-05
		qPSO-W	96	29508	4.00088	0.000571	19.5957
		qPSO-C	100	11601	0.000913	0.000622	7.87E-05
14.	Brown3	PSO-W	26	40813	2.59024	0.000825	2.28925
		PSO-C	100	6224	0.0009	0.000622	8.33E-05
		qPSO-W	79	20268	0.760712	0.000157	1.77801
		qPSO-C	100	5921	0.000915	0.000659	7.20E-05
15.	Schewefel 3	PSO-W	62	35633	4.900587	0.000795	6.998875
		PSO-C	100	13805	0.000925	0.000125	0.000111
		qPSO-W	100	23655	0.000942	0.000367	7.20E-05
		qPSO-C	100	15803	0.000904	2.64E-05	0.000178

weighted importance to the success rate, the mean objective function value as well as the average number of function evaluations. The value of this performance index for a computational algorithm under comparison is given by

$$PI = \frac{1}{N_p}\sum_{i=1}^{N_p}\left(k_1\alpha_1^i + k_2\alpha_2^i + k_3\alpha_3^i\right)$$

Where $\alpha_1^i = \frac{Sr^i}{Tr^i}$; $\alpha_3^i = \begin{cases} \frac{Mf^i}{Af^i}, & if\ Sr^i > 0 \\ 0 & if\ Sr^i = 0 \end{cases}$ and $\alpha_3^i = \frac{Mo^i}{Ao^i}$.

$i =$ 1,2,..., N_p

$Sr^i =$ Number of successful runs of i^{th} problem

$Tr^i =$ Total number of runs of i^{th} problem

$Mf^i =$ Minimum of average number of function evaluations of successful runs used by all algorithms in obtaining the solution of i^{th} problem

$Af^i =$ Average number of function evaluations of successful runs used by an algorithm in obtaining the solution of i^{th} problem

$Mo^i =$ Minimum of mean objective function value obtained by all the algorithms for the i^{th} problem

$Ao^i =$ Mean objective function value obtained by an algorithm for the i^{th} problem

$N_p =$ Total number of problems analyzed.

k_1, k_2 and k_3 ($k_1 + k_2 + k_3$ and $k_1, k_2, k_3 \leq 1$) are the weights assigned to success rate, average number of function evaluations of successful runs and mean objective function value, respectively. From above definition it is clear that PI is a function of k_1, k_2 and k_3. Since $k_1 + k_2 + k_3 = 1$ one of k_i, $i = 1,2,3$ could be eliminated to reduce the number of dependent variables from the expression of PI. But it is still difficult to analyze the behavior of PI, because the surface plots of PI for PSOs and qPSOs are overlapping and it is difficult to visualize them. So, we adopt the same methodology as given in [38] i.e. equal weights are assigned to two terms at a time in the PI expression. This way PI becomes a function of one variable. The resultant cases are as follows:

(i) $k_1 = W,\ k_2 = k_3 = \frac{1-W}{2},\ 0 \leq W \leq 1$

(ii) $k_2 = W,\ k_1 = k_3 = \dfrac{1-W}{2},\ 0 \le W \le 1$

(iii) $k_3 = W,\ k_1 = k_2 = \dfrac{1-W}{2},\ 0 \le W \le 1$

The graphs corresponding to each of the cases (i), (ii) and (iii) are shown in Figures 3(a)–3(c) respectively. In these figures the horizontal axis represents the weight W and the vertical axis represents the performance index PI.

In case (i), average number of function evaluations of successful runs and the average objective function value are given equal weights. PI's of all four algorithms are superimposed in the Fig. 3(a) for comparison and to get a ranking of the performance of the four algorithms. It is observed that for qPSO-C the value of PI is more than all the remaining three PSOs. The remaining PSOs perform in the order PSO-C > qPSO-W > PSO-W.

In case (ii), equal weights are assigned to the percentage of success and average number of function evaluations of successful runs. From Fig. 3(b), it is clear that all PSOs perform same as in case (i).

In case (iii), equal weights are assigned to the percentage of success and average objective function value. Again the same conclusion is drawn from Fig. 3(c).

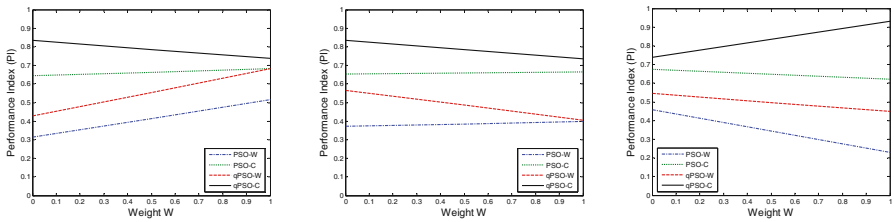

Fig. 3. Performance index; (a) for case (i), (b) for case (ii) and (c) for case (iii)

As an overall conclusion from PI we can say that the qPSOs are better than their respective pure versions.

Further, we would like to view the comparative decrease in the objective function value by PSOs and the qPSOs. For this a typical run of each of the method for each problem is shown in Fig. 4. The X-axis represents the iteration number and the Y-axis represents the error. It is observed that the qPSOs converges very rapidly to the minima as compared to PSOs in most of the problems.

5. Solution of some real life problems

In order to check the robustness of proposed methods, three real life problems namely Girder design [39], pressure vessel [40] and water distribution [41] have been solved using original PSOs and their hybrid versions developed by us. The actual optimum values of these real life problems are not known. Therefore the experiments were carried out by running all the PSO versions for 50000 functions evaluations. The average objective function value, the best objective function value and standard deviation were obtained and given in Table 4. Constraints of these problems were handled using penalty function approach [41].

In the case of girder design problem, as such all PSO algorithms obtained the same minimum objective function value. However, mean objective function value obtained in case of qPSO-C was better than all other versions of PSO. Mean objective function value obtained by qPSO-W was better than value obtained by its pure version i.e. PSO-W. In pressure vessel problem again the minimum value obtained was same by all PSO versions but the mean objective function value obtained by qPSO-C was the best. On the criteria of mean objective function value the qPSO-W is better than PSO-W, in the case of water distribution problem the best mean objective function value is obtained by qPSO-C. However, in this problem qPSO-W shows inferior performance as compared to PSO-W.

6. Conclusion

In this paper a novel PSO algorithm based on the hybridization of PSO with quadratic approximation operator, namely, qPSO is introduced. Two of the most popular variants of PSO algorithm namely PSO with linearly decreasing inertia weight (PSO-W) and PSO with constriction factor (PSO-C) are considered for hybridization. The two resultant hybridized variants proposed are qPSO-W and qPSO-C. The hybridization is performed by splitting the swarm into two subswarms in such a way that one subswarm is evolved using PSO, whereas another is evolved using QA. The performance of these variants is evaluated on the basis of 15 benchmark problems taken from literature. Based on the numerical results it is clear that the proposed hybrid PSO variants outperform the original PSO variants in terms of efficiency, reliability and robustness. The various performance criteria are consolidated into a performance index which clearly indicates the efficacy of the proposed algorithms.

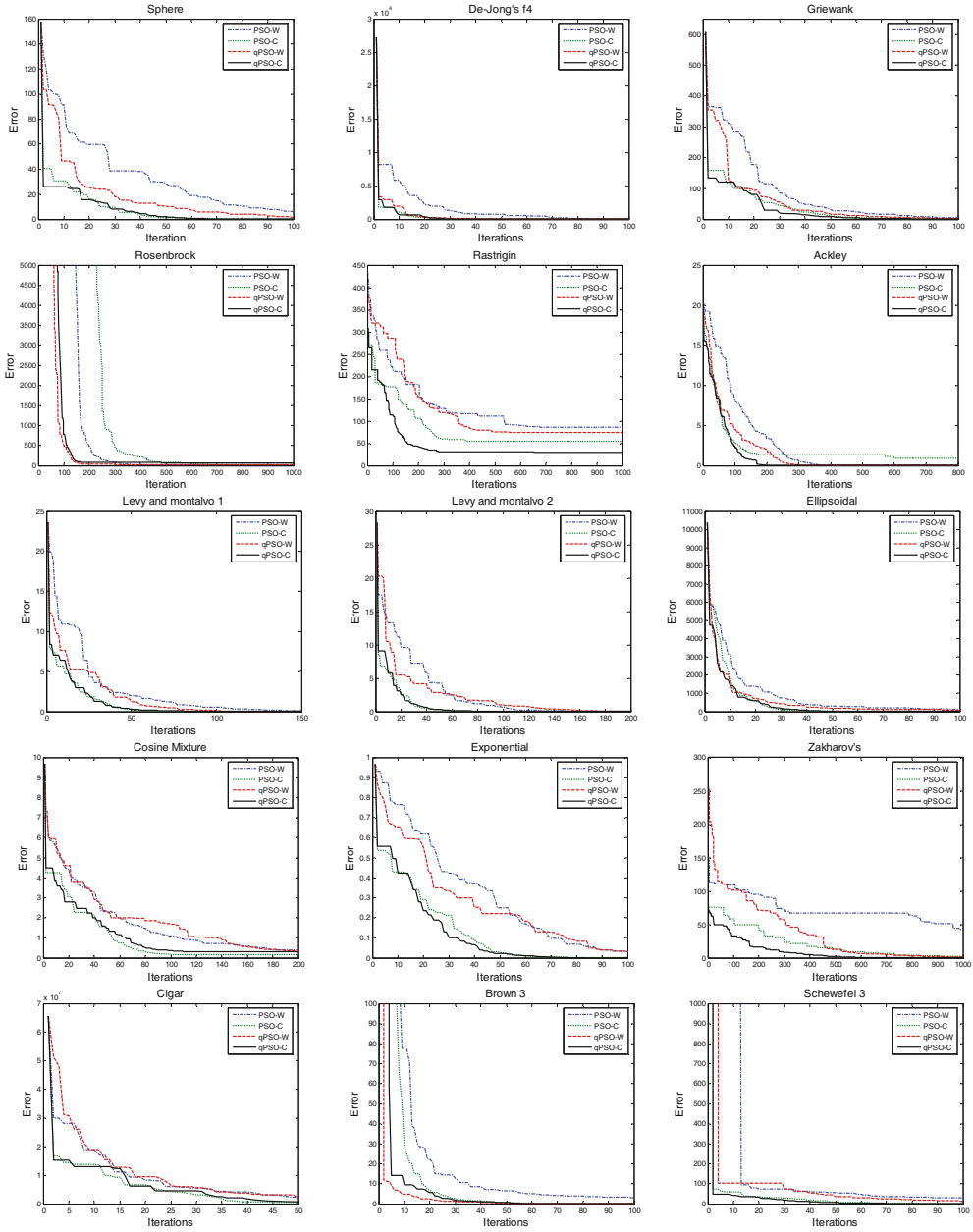

Fig. 4. Convergence graphs of benchmark problems

Table 4: Results of real life problems

Problem	Algorithm	Average bjective function value	Best objective function value	Standard deviation
Girder design	PSO-W	0.905716	0.896857	0.006524
	PSO-C	0.900075	0.896858	0.004062
	qPSO-W	0.902558	0.896856	0.006554
	qPSO-C	0.898947	0.896856	0.002578
Pressure vessel	PSO-W	7049.35	7019.03	254.166
	PSO-C	7023.44	7019.03	33.0765
	qPSO-W	7022.33	7019.03	16.9434
	qPSO-C	7021.42	7019.03	11.6045
Water distribution	PSO-W	2087746.114	2087600.322	170.4204655
	PSO-C	2087794.271	2087600.436	259.2144961
	qPSO-W	2087866.586	2087600.818	338.4060376
	qPSO-C	2087671.680	2087600.374	105.4344862

In order to demonstrate the versatility of the proposed algorithms, PSOs and qPSOs are also used to solve three real life application problems arising in the field of engineering. Based on the numerical results it is concluded that the proposed algorithms are very promising in determining the global optimal solution of nonlinear optimization problems.

Acknowledgment

Authors acknowledge Prof. Chander Mohan for his valuable comments and suggestions. The second author gratefully acknowledges funding from University Grant Commission (UGC), India under grant number 6405-11-61.

References

[1] Kennedy, J., Eberhart, R.: Particle swarm optimization. Proceedings IEEE International Conference Neural Networks, **4**, 1942–1948 (1995)

[2] Arumugam, M., Senthil, Rao, M.V.C., Tan Alan W.C.: A novel and effective particle swarm optimization like algorithm with extrapolation technique. Applied Soft Computing, **9**, 308–320 (2009)

[3] Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computations, **1**, 67–82 (1997)

[4] Angeline, P.J.: Using selection to improve particle swarm optimization. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Anchorage, Alaska, USA, 84–89 (1998)

[5] Løvbjerg, M., Rasmussen, T., Krink, T.: Hybrid particle swarm optimizer with breeding and subpopulations. Proceedings of the 3rd Genetic and Evolutionary Computation Conference (GECCO-2001), **1**, 469–476 (2001)

[6] Miranda, V., Fonseca, N.: New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control. The 14th Power Systems Computation Conference (PSCC'02), Seville, Spain (2002)

[7] Løvbjerg, M., Krink, T.: Extending particle swarms with self-organized criticality. Proceedings of the 4th Congress on Evolutionary Computation (CEC-2002), 1588–1593 (2002)

[8] Blackwell, T., Bentley, P.J.: Don't push me! Collision-avoiding swarms. IEEE Congress on Evolutionary Computation, 2002 Honolulu, Hawaii USA, 1691–1696 (2002)

[9] Krink, T., Vesterstrøm, J.S., Riget, J.: Particle swarm optimization with spatial particle extension. Proceedings of the 4th Congress on Evolutionary Computation (CEC-2002), 1474–1479 (2002)

[10] Higashi, N., Iba, H.: Particle swarm optimization with Gaussian mutation. Proceedings of the IEEE swarm intelligence symposium 2003 (SIS 2003), Indianapolis, Indiana, USA, 72–79 (2003)

[11] Ratnaweera, A., Halgamuge, S.K., Watson, H.C.: Self-organizing hierarchical particle swarm optimizer with time varying accelerating coefficients. IEEE Transactions on Evolutionary Computation **8**, No. 3, 240–255 (2004)

[12] Juang, C. F.: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans Syst Man Cybern – Part B: Cybern **34**(2), 997–1006 (2004)

[13] Blackwell, T., Bentley, P.J.: Don't push me! collision-avoiding swarms. IEEE Congress on Evolutionary Computation, 2002 Honolulu, Hawaii USA, 1691–1696 (2002)

[14] Robinson, J., Sinton, S., Rahmat-Samii, Y.: Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. IEEE International Symposium on Antennas & Propagation. San Antonio, Texas, 314–317 (2002)

[15] Jian, M., Chen, Y.: Introducing recombination with dynamic linkage discovery to particle swarm optimization. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2006), 85–86 (2006)

[16] Grimaccia, F., Mussetta, M., Zich, R.E.: Genetical swarm optimization: self-adaptive hybrid evolutionary algorithm for electromagnetics. Antennas and Propagation, IEEE Transactions on, **55**(3), 781–785 (2007)

[17] Poli, R., Di, Chio. C., Langdon, W.B.: Exploring extended particle swarms: a genetic programming approach. In: Beyer, H.-G. et al. (eds.), GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, Washington, DC, pp. 169–176. New York: ACM (2005)

[18] Poli, R., Langdon, W.B., Holland, O.: Extending particle swarm optimization via genetic programming. In: Keijzer, M. et al. (eds.) Lecture notes in computer science. Proceedings of the 8th European Conference on Genetic Programming, Lausanne, Switzerland. vol. 3447, pp. 291–300. Berlin: Springer (2005)

[19] Krink, T., Løvbjerg, M.: The lifecycle model: combining particle swarm optimisation, genetic algorithms and hillclimbers. Proceedings of Parallel Problem Solving from Nature (PPSN), Granada, Spain, 621–630 (2002)

[20] Hendtlass, T.: A combined swarm differential evolution algorithm for optimization problems. Proceedings of the 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Lecture Notes in Computer Science, vol. 2070, pp. 11–18 Springer-Verlag (2001)

[21] Zhang, WJ., Xie, XF.: DEPSO: hybrid particle swarm with differential evolution operator. IEEE International Conference on Systems, Man and Cybernetics (SMCC), Washington DC, USA, 3816–3821 (2003)

[22] Hendtlass, T., Randall, M.: A survey of ant colony and particle swarm metaheuristics and their application to discrete optimization problems. Proceedings of The Inaugural Workshop on Artificial Life (AL'01), 15–25 (2001)

[23] Van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization", IEEE Transactions on Evolutionary Computation, **8**(3) 225–239 (2004)

[24] Parsopoulos, K.E., Vrahatis, M.: On the computation of all global minimizers through particle swarm optimization. IEEE Transactions on Evolutionary Computation, **8**(3), 211–224 (2004)

[25] Xie, X., Zhang, W., Yang, Z.: A dissipative particle swarm optimization. IEEE Congress on Evolutionary Computation, 2002, Honolulu, Hawaii USA, 1456–1461 (2002)

[26] Liu, H., Abraham, A.: Fuzzy adaptive turbulent particle swarm optimization. Proceedings of 5th International Conference on Hybrid Intelligent Systems (HIS'05), Rio de Janeiro, Brazil, 6–9 November 2005, 39–47 (2005)

[27] Banks, A., Vincent, J., Anyakoha, C.: A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Natural Computing: an International Journal, **7**(1), 109–124 (2008)

[28] Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. Proceedings of the IEEE International Conference on Evolutionary Computation, Piscataway, NJ: IEEE Press, 69–73 (1998)

[29] Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. The 7th Annual Conference on Evolutionary Programming, San Diego, USA, 591–600 (1998)

[30] Eberhart, R.C., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. Congress on Evolutionary Computing, **1**, 84–88 (2000)

[31] Clerc, M.: The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Proceedings, 1999 ICEC, Washington, DC, 1951–1957 (1999)

[32] Deep, K., Das, K.N.: Quadratic approximation based hybrid genetic algorithm for function optimization. Applied Mathematics and Computation, **203**(1) 86–98 (2008)

[33] Mohan, C., Shanker, K. (now Deep, K.): A random search technique for global optimization based on quadratic approximation. Asia Pacific Journal of Operations Research, **11** 93–101 (1994)

[34] Bharti: Controlled random search technique and their applications, Ph.D. Thesis. Department of Mathematics, University of Roorkee, Roorkee, India (1994)

[35] Ali, M.M., Kaelo, P.: Improved particle swarm algorithms for global optimization. Applied Mathematics and Computation, **196**, 578–593 (2008)

[36] Zhang, Li-ping., YU, Huan-jun, HU, Shang-xu: Optimal choice of parameters for particle swarm optimization. Journal of Zhejiang University Science, **6A**(6), 528–534 (2005)

[37] Schutte, Jaco. F., Groenwold, Albert. A.: A study of global optimization using particle swarms. Journal of Global Optimization, **31**, 93–108 (2005)

[38] Deep, K., Thakur, M.: A new crossover operator for real coded genetic algorithms. Applied Mathematics and Computation **188**(1), pp. 895–911 (2007)

[39] Arora, J.S.: Introduction to Optimum Design, 2/e Academic Press, Elsevier. UK (2006)

[40] Clerc, M.: Particle Swarm Optimization. ISTE Ltd., USA (2007)

[41] Bhave, P.R.: Optimal Design of Water Distribution Networks. Narosa Publishing House, New Delhi, India (2003)