A Modified Particle Swarm Optimization Algorithm for Function Optimization

Ashok Pal*[a], S.B.Singh[a] and Kusum Deep[b]

*[a]*Department of Mathematics Punjabi University, Patiala (Pb)-India*

[b]*Department of Mathematics, Indian Institute of Technology, Roorkee-India*
*ashokpmaths@gmail.com

Abstract

Particle Swarm Optimization (PSO) is a very popular population based heuristic search algorithm for optimization developed by Eberhart and Kennedy in 1995(Kennedy *et al.,* 1995), usually requires a large number of fitness evaluations to reach the global optima. The PSO algorithm is easy to implement and has been proven to be very competitive for solving diverse global optimization problems including both benchmark and real life optimization problems in comparison to conventional methods and other meta-heuristics. In this paper we have proposed an improved particle swarm optimization algorithm named a Random Search Quadratic approximation Particle Swarm Optimization (RQPSO) algorithm tested on 15 non scalable nonlinear benchmark optimization problems taken from literature and having a number of local as well global optimal solutions. The experimental results show that the proposed algorithm improves its performance in terms of used number of function evaluations (AFE), rate of success (SR), average error(AE) and standard deviation of error(SD).

Keywords - Standard Particle Swarm Optimization (SPSO), Random Search Quadratic approximation PSO (RQPSO), Non Scalable Optimization Problems.

Introduction

Particle Swarm Optimization (PSO) is a heuristic optimization technique introduced by Kennedy and Eberhart in 1995(Kennedy *et al.,* 1995). It is inspired by the intelligent, experience-sharing, social flocking behaviour of birds that was first simulated on a computer by Craig Reynolds (Reynolds, 1987), and further studied by the biologist Frank Heppner (Reynolds and Grenander, 1990). PSO is a population-based search strategy that finds optimal solutions using a set of flying particles with velocities that are dynamically updated according to their historical performance, as well as their neighbours in the search space (Shi, 2004).PSO solves problems whose solutions can be represented as a *set of points* in an n-dimensional solution space. The term *'particles'* refers to population members, which are fundamentally described as the swarm positions in the n-dimensional solution space. Each particle is set into motion through the solution space with a velocity vector representing the particle's speed in each dimension. Each particle has a memory to store its historically best solution (i.e., its best position ever attained in the search space so far, which is also called its experience).

Some of the important features of PSO include ease of its implementation and none of any gradient information is required for it. In PSO, solution space of the problem is called a search space and each position in the search space is known as a potential solution of the problem. Particles target to find the best position (i.e. best solution) in the search space i.e. in the solution space. For a n- dimensional search space, the position of the position of the with particle is denoted as $X_i = (x_{i1}, x_{i2}, \ldots\ldots x_{in})$ Each particle maintains a memory of its previous best position $p_i = (p_{i1}, p_{i2}, \ldots\ldots p_{in})$. The best particle (i.e. the particle which gives the best fitness value) among all the particles in the swarm is denoted as $p_g = (p_{g1}, p_{g2}, \ldots\ldots p_{gn})$

and the velocity of each particle in the swarm is denoted by $V_i = (v_{i1}, v_{i2}, ...v_{in})$

The two basic equations in the working of the standard particle swarm optimization (SPSO) algorithm are the velocity and position vectors which are given as

$v_i(t+1) =$

inertial weight component *cognitive component* *social component*

$$\overbrace{wv_i(t)} + \overbrace{c_1 r_1 [p_i(t) - x_i(t)]} + \overbrace{c_2 r_2 [p_g(t) - x_i(t)]} \quad ...(1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad ....(2)$$

Where $1 \le swarm\,(i) \le S$, $1 \le \dim(D) \le n$

Equations (1) and (2) respectively are termed as velocity and position update equations in standard particle swarm optimization (SPSO).

In these equations (1) and (2), $x_i(t)$ is the position of the with particle at any time t, $v_i(t)$ is the velocity of the with particle at time t, p_i (t) is the best position found by the with particle itself in the swarm so far, $p_g(t)$is the best position found by the whole swarm so far. The inertia weight w and the acceleration constants c_1 and c_2 (Shi and Eberhart, 1998) are predefined taken from the literature by the user and r_1 & r_2 are the random numbers uniformly generated in the range of 0 and 1.

The three terms in equation (1) described as:

(i) The first term is the current speed of the particle which shows its present state and has the ability to balance the whole swarm and search a local part.

(ii) The second term in equation (1) is called the cognitive component here which shows the thought of the particle itself and causes the swarm to have a strong ability to search the whole swarm and avoid a local minimum.

(iii) The third term is known as the social component which shows the information sharing among the whole swarm and among the particles (i.e. solutions) which are known to be near to good (i.e. best) solutions. With the help of these three terms, the particles can reach to an effective and best position.

If we assume that the optimization problem is of minimization, then in SPSO how the personal and the global best values are updated at the time t respectively are given as

$$p_i(t+1) = \begin{cases} p_i(t), when f[p_i(t)] \le f[x_i(t+1)] \\ x_i(t+1), when f[p_i(t)] > f[x_i(t+1)] \end{cases} .....(3)$$

$$p_g(t+1) = \min\{f(y), f[p_g(t)]\} .....(4)$$

where $y \in [p_0(t), p_1(t), p_2(t), ..., p_n(t)]$

A Random Search Technique with Quadratic Approximation

N.H. Thanh and C. Mohan (Thanh and Mohan, 1996) determined a new trial point p in the manner of a 'Control Random Search Technique for Global Optimization using quadratic approximation'(Mohan and Shankar, 1994). In this random search technique we choose

$$p = 0.5 * \frac{\begin{matrix}[f(b_1)(b_2^2 - b_3^2) + f(b_2)(b_3^2 - b_1^2) \\ + f(b_3)(b_1^2 - b_2^2)]\end{matrix}}{\begin{matrix}[f(b_1)(b_2 - b_3) + f(b_2)(b_3 - b_1) \\ + f(b_3)(b_1 - b_2)]\end{matrix}} ..(5)$$

Where p gives the extremal point of the quadratic curve passing through the points b_1, b_2 and b_3.

The Proposed Algorithm

Kusum Deep and J.C. Bansal (Deep and Bansal, 2009) proposed an algorithm named Quadratic approximation Particle Swarm Optimization (qPSO) in which the hybridization of PSO is performed with Quadratic Approximation operator (QA), by splitting the whole swarm into two sub swarms in such a way that the PSO operators are applied on one sub swarm, whereas the QA operator is applied on the other sub swarm, ensuring that both sub swarms are updated using the global best particle of the entire swarm.

The proposed algorithm in the present paper is a hybrid of the standard particle swarm optimization (SPSO) algorithm (Maurice,2006) and a random search technique with quadratic approximation formula (Mohan and Shankar, 1994) named Random Search Quadratic approximation Particle Swarm Optimization (RQPSO) algorithm. In this proposed algorithm, a probability say having a certain value provided by the user has been fixed. In every iteration, if the uniformly generated random number r(0,1) is less than that value , then the velocity vector is generated by the equation (1) of standard PSO algorithm otherwise it is generated by equation (5) of random search technique with quadratic approximation formula (Mohan and Shankar, 1994).

The flow of the proposed algorithm is as under:

BEGIN:

Create and Initialize an n-dimensional swarm S

{ $x_i(t)$: =1to S} uniformly between 0 and 1.

Assign w some value between 0.4 to 0.9 and
set c_1 and c_2 =2.0.
For i=1 to S,
For d=1 to n,
Assign some value to P between 0 and 1.
If r (0, 1) < P , then
generate velocity vector using equation (1) of
standard PSO algorithm,
else generate it using equation (5) of controlled
random search with quadratic
approximation formula (Mohan and Shankar, 1994) .
Calculate parti cle position as

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

End- for-d;
Compute fitness of updated position; if needed,
update historical information for P_i and P_g
End-for-i;
Terminate if P_g meets problem requirements;

END

Performance Testing Criteria and Setting of Parameters

Successful Runs: A run in which at least one solution is discovered with error tolerance 0.001.
AFE: Average number of function evaluations for successful runs.
SR = Success Rate
= (No of successful runs (NR) /Total runs)*100.

= % age of successful runs to total runs.

$$\text{AE} = \text{Average Error} = \frac{\sum_{n}(f_{\min} - f_{opt})}{NR},$$

Where NR is the number of runs.

SD: Standard deviation of the error.

Table 1 : Parameter Settings**.**

S.N	Parameter	Symbol	Value
1.	Swarm Size	SS	Normally=(5-10)*DD: Dim ension
2.	Max no . of function evaluation s taken	MFE	50000
3.	Max no of function evaluation s used	AFE	Pb wise given in the table no 4 of results
4.	Number of runs	NR	30
5.	Inertia weight	W	0.4 to 0.9
6.	Acceleration Coeff .	c_1, c_2	$c_1 = c_2 = 2$
7.	Error tolerance	ε	0.001
8.	PC Configuration	Processor	Intel Dual Core
		RAM	2 GB
		Operating System	Windows XP
		Software used	C++/Visual Studio
Random numbers are generated using inbuilt rand () function for the algorithm.			

Results and Discussion

The acceleration factors C_1 and C_2 equals to 2 are used here, however the other settings were also be used in the literature but C_1 and C_2 are equals usually and ranges from 0 to 4 (Shi and Eberhart, 1998). The swarm size i.e. number of particles we used here (5 to 10)* dimension. There is no any hard and fast rule for it. For most of the problems 10 to 50 particles are large enough to get good result. For solving all the benchmark problems in this research paper a C++ code has been developed and compiled in Microsoft visual C++ compiler and the data recorded as per the tables 3 to 6 given in this section.

Conclusions

In the present research paper a modified particle swarm optimization algorithm named Random Search Quadratic Approximation Particle Swarm Optimization (RQPSO) algorithm has been proposed and tested on 15 benchmark non scalable nonlinear optimization problems taken from the literature and having number of local as well as global optimal solutions. Results are compared with a, Hybrid PSO (HPSO) (Deep and Bansal, 2009) and it has been observed that the proposed algorithm improves its performance in terms of the average number of function evaluations, success rate, average error and standard deviation in most of the cases.

Table 2 : List of Benchmark Problems.

PbNo	Problem name	Range	Sofⁿ Min(f)
1.	Easom 2D [9]	[-10,10]	-1
2.	Becker and Lago [10]	[-10,10]	0
3.	Aluffi-Pentini's [11]	[-10,10]	-0.3523
4.	Wood's [9],[12]	[-10,10]	0
5.	Miele and Cantrell [12]	[-1,1]	0
6.	Bohachevsky 1 [13]	[-50,50]	0
7.	Bohachevsky 2 [13]	[-50,50]	0
8.	Branin [14]	[-5,10], [0,15]	$5/(4\pi)$
9.	Eggcrate [15]	$[-2\pi, 2\pi]$	0
10.	Modified Rosen.[10]	[-5,5]	0
11.	Periodic[10]	[-10,10]	0.9
12.	Powell's [12]	[-1,4]	0
13.	Camel Back -3 Hump Problem[14]	[-5,5]	0
14.	Camel Back -6 Hump Problem [14]	[-5,5]	-1.0316
15.	McCormick Problem [16]	[-1.5,4], [-3,3]	-1.9133

Table 3 : Comparison of RQPSO with SPSO and HPSO in terms of rate of success (SR)(%).

PbNo	SPSO	HPSO	RQPSO	
			r(0,1)<0.4	r(0,1)<0. 1
1.	100	100	100	100
2.	100	100	100	100
3.	100	100	100	100
4.	100	100	100	100
5.	100	100	100	100
6.	100	100	100	100
7.	100	100	100	100
8.	100	100	100	100
9.	100	100	100	100
10.	60	67	60	70
11.	75	86	100	100
12.	100	100	100	100
13.	100	100	100	100
14.	100	100	100	100
15.	100	100	100	100

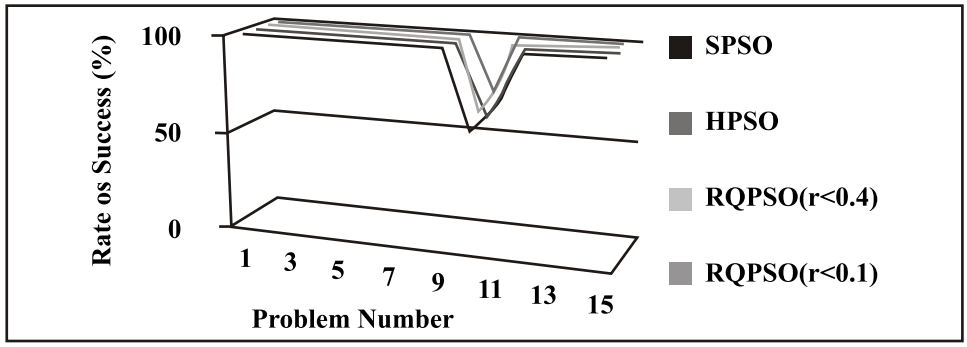

Figure 1: Comparison of RQPSO vs SPSO and HPSO based on rate of success.

Table 4 : Comparison of RQPSO with SPSO and HPSO in terms of average number of function evaluations (AFE) used.

Pb No	SPSO	HPSO	RQPSO	
			r(0,1)<0.4	r(0,1)<0.1
1.	508	548	2959	5854
2.	623	605	293	429
3.	426	409	341	403
4.	18754	17979	5416	10462
5.	226	223	254	376
6.	977	1071	2094	957
7.	1007	1065	1142	673
8.	514	591	346	394
9.	642	716	1395	816
10.	3360	2783	493	1620
11.	4665	2540	3010	2875
12.	1463	1463	2787	2564
13.	366	389	203	240
14.	590	443	346	367
15.	293	305	192	323

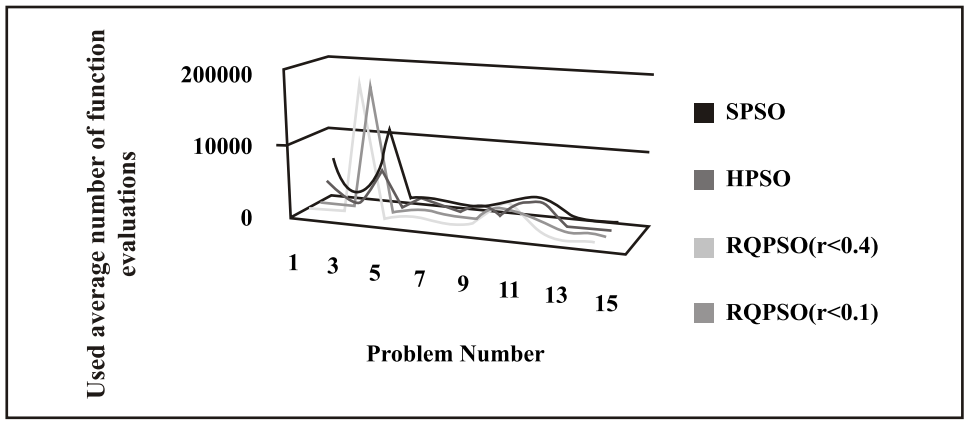

Figure 2 : Comparison of RQPSO With SPSO & HPSO based on AFE.

Table 5 : Comparison of RQPSO with SPSO and HPSO in terms of average error(AE).

Pb No	SPSO	HPSO	RQPSO	
			r(0,1)<0.4	r(0,1)<0.1
1.	0.000453	0.000459	0.000208	0.000137
2.	0.000474	0.000482	0.000436	0.000089
3.	0.000422	0.000442	0.000230	0.000263
4.	0.00096	0.000937	0.004085	0.000356
5.	0.000401	0.000343	0.000277	0.000361
6.	0.000511	0.000525	0.000399	0.000037
7.	0.000536	0.000516	0.000380	0.000204
8.	0.000476	0.000518	0.000441	0.000276
9.	0.000492	0.000526	0.000305	0.000256
10.	0.003211	0.002810	0.003273	0.002480
11.	0.010510	0.012548	0.000164	0.000122
12.	0.000652	0.000636	0.000724	0.000552
13.	0.000522	0.000453	0.000461	0.000442
14.	0.000608	0.000627	0.000422	0.000570
15.	0.000545	0.000563	0.000331	0.000509

Table 6 : Comparison of RQPSO with SPSO and HPSO based on standard deviation(SD) of error.

Pb	SPSO	HPSO	RQPSO	
			r(0,1)<0.4	r(0,1)<0.1
1.	0.000302	0.000277	0.000207	0.000250
2.	0.000276	0.000282	0.000324	0.000179
3.	0.000315	0.000295	0.000283	0.000286
4.	0.000060	0.000125	0.017242	0.085128
5.	0.000325	0.000302	0.000243	0.000342
6.	0.000297	0.000216	0.000394	0.000113
7.	0.000274	0.000287	0.000310	0.000277
8.	0.000286	0.000312	0.000317	0.000212
9.	0.000277	0.000285	0.000251	0.000265
10.	0.003331	0.003240	0.003392	0.003244
11.	0.029833	0.032300	0.000228	0.029959
12.	0.000241	0.000264	0.000206	0.000258
13.	0.000292	0.000298	0.000181	0.000359
14.	0.000217	0.000213	0.000202	0.000241
15.	0.000253	0.000272	0.000262	0.000233

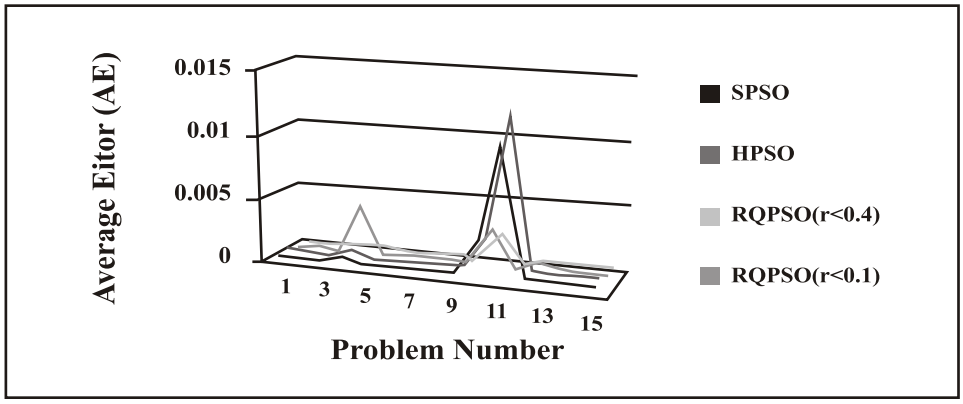

Figure 3 : RQPSO vs SPSO & HPSO based on AE.

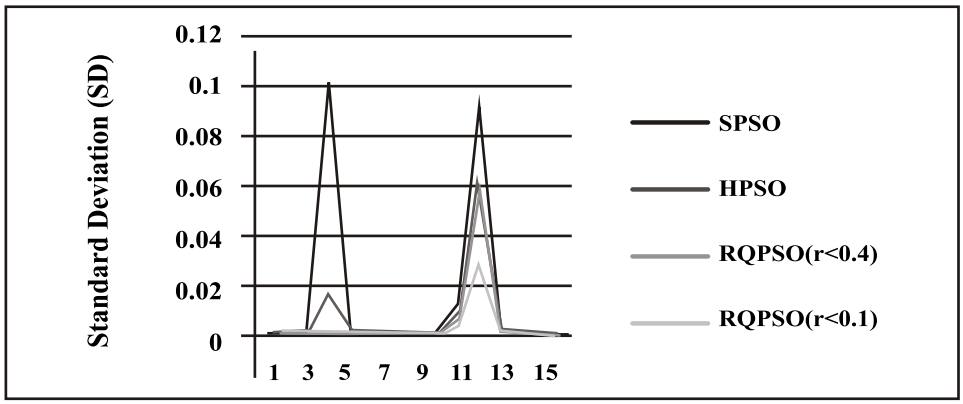

Figure 3 : RQPSO vs SPSO & HPSO based on SD.

References

Aluffi-Pentini, F., Parisi, V., Zirilli, F. 1985. Global optimization and stochastic differential equations. *Journal of Optimization Theory and Applications*, 47, 1–16,

Bohachevsky, M.E., Johnson, M.E., Stein, M.L. 1986. Generalized simulated annealing for function optimization. *Technometrics* 28, 209–217.

Deep, K., Bansal, J. C. 2009. Hybridization of particle swarm optimization with quadratic approximation. *OPSEARCH*,46(1),1-22.

Dixon, L.., Szego, G. 1978. Towards Global Optimization. *2, North Holland, New York.*

Eberhart, R. C., Shi, Y. 2000. Comparing inertia weights and constriction factors in particle swarm optimization. *Congress on Evolutionary Computing*, 1, 84-88.

Hassan, R., Cohanim, B., de Weck, O., Venter, G. 2005. A Comparison of Particle Swarm Optimization and the Genetic Algorithm. *46th AIAA / ASME/ ASCE / AHS / ASC Structures, Structural Dynamics and Materials Conference, Austin, Texas,* No. AIAA-2005-1897.

Heppner, F., Grenander, U. 1990. A stochastic nonlinear model for coordinated bird flocks. *In S Krasner, Ed., the Ubiquity of Chaos. AAAS Publications*, Washington, DC.

Kennedy, J., Eberhart, R.C. 1995. Particle swarm optimization. *Proceeding IEEE International Conference of Neural Network IV*, pp.1942-1948.

Maurice Clerc. Confinements and biases in particle swarm optimisation.Technical report, Open archive HAL http://hal.archives-ouvertes.fr/, ref. hal- 00122799, 2006.

McCormick, G.P. 1982. Applied Nonlinear Programming, Theory, Algorithms and Applications. *John Wiley and Sons, New York.*

Michalewicz, Z. 1996. Genetic Algorithms+Data Structures = Evolution Programs. *Springer- Verlag, Berlin /Heidelberg /NewYork.*

Mohan, C., Shankar K. 1994. A control random search technique for global optimization using quadratic approximation. *Asia pacific journal of operational research*, 11, 93-101,

Price, W.L. 1977. Global optimization by Controlled Random Search. *Computer Journal*, 20, 367-370.

Reynolds, C. W. 1987. Flocks, herds, and schools: a distributed behavioural model, *Computer Graphics (ACM SIGGRAPH '87 Conference Proceedings*, Vol. 21(4), 25–34.

Shi, Y. 2004. Feature article on particle swarm optimization. *IEEE Neural Network Society, Feature Article*, pp. 8- 13.

Shi, Y., Eberhart, R. C. 1998. A modified particle swarm optimizer. *Proc. of IEEE Int. Conf. On Evolutionary Computation*, pp.69-73.

Thanh, N.H., Mohan, C. 1996. Some global optimization techniques and their use in solving optimization problems in crisp and fuzzy environments. *PhD Thesis, University of Roorkee, Roorkee (India)*.

Wolfe, M.A. 1978. Numerical Methods for Unconstrained Optimization. *Van Nostrand Reinhold Company, New York.*