A New Particle Swarm Optimization Algorithm Incorporating Reproduction Operator for Solving Global Optimization Problems

Millie Pant[1], Radha Thangaraj[1] and Ajith Abraham[2]
[1]*Department. of Paper Techology, IIT Roorkee, India*
[2]*Center of Excellence for Quantifiable Quality of Service,*
Norwegian University of Science and Technology, Norway
millifpt@iitr.ernet.in, t.radha@ieee.org, ajith.abraham@ieee.org

Abstract

This paper presents a new variant of Basic Particle Swarm Optimization (BPSO) algorithm named QI-PSO for solving global optimization problems. The QI-PSO algorithm makes use of a multiparent, quadratic crossover/reproduction operator defined by us in the BPSO algorithm. The proposed algorithm is compared it with BPSO and the numerical results show that QI-PSO outperforms the BPSO algorithm in all the sixteen cases taken in this study.

1. Introduction

Evolutionary Algorithms (EA) and Particle Swarm Optimization (PSO) are perhaps the two most common population based, stochastic search techniques for solving continuous global optimization problems. Both the techniques have been quite efficient in solving complex optimization problems (test cases as well as real life problems). On one hand, EA are inspired from the metaphor of natural biological evolution using the concepts of selection, mutation and reproduction. On the other hand, PSO uses the complex social cooperative and competitive behavior exhibited by different species like birds, bees humans etc. some of the similarities between the EA and PSO as pointed out by Angeline [1] may be given as:

- Both are population based search techniques.
- Neither requires the auxiliary knowledge of the problem.
- In both the algorithms, solutions belonging to the same population interact with each other during the search process to move towards the better optima.
- The quality of the solutions are improved using techniques inspired from real world phenomenon like human genetics in case of EA and cooperative behavior in case of PSO.

Both the algorithms have their share of weaknesses and strengths and One of the simplest methods to enhance the performance of both the algorithms is to exploit the strengths of both the algorithms together in a single algorithm. A number of techniques suggesting a combo of these two methods are available in literature. Out of the EA operators' viz. selection, crossover and mutation, the one that has been used most frequently is the mutation operator. Some of the commonly used mutation operators used in PSO algorithms are Gaussian, Cauchy, linear, nonlinear operators etc. For a detailed description, the reader is suggested [2] - [7]. However, for selection and reproduction operator only a few examples are available (see for instance Angeline [8], Clerc [9]).

In this paper we have suggested the use of a new crossover operator named quadratic crossover operator in a BPSO algorithm. The most potent particle (one having the lowest objective function value) is selected as the leader of the swarm (tribe). The leader is then allowed to mate with two other randomly chosen particles of the swarm (tribe). The child (or new particle) lies at the point of minima of the quadratic curve passing through the three (different) chosen particles (leader and two mates).

The remaining paper is structured as follows: in Section 2, we have briefly explained the Basic Particle Swarm Optimization, in Section 3; we have defined the proposed QI-PSO algorithm. Section 4 deals with experimental settings. Section 5 gives numerical results and finally the paper concludes with Section 6.

0-7695-2946-1/07 $25.00 © 2007 IEEE
DOI 10.1109/HIS.2007.20

2. Basic Particle Swarm Optimization

PSO is a relatively newer addition to a class of population based search technique for solving numerical optimization problems. The particles or members of the swarm fly through a multidimensional search space looking for a potential solution. Each particle adjusts its position in the search space from time to time according to the flying experience of its own and of its neighbors (or colleagues).

For a D-dimensional search space the position of the ith particle is represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. Each particle maintains a memory of its previous best position $P_{besti} = (p_{i1}, p_{i2} \ldots p_{iD})$. The best one among all the particles in the population is represented as $P_{gbest} = (p_{g1}, p_{g2} \ldots p_{gD})$. The velocity of each particle is represented as $V_i = (v_{i1}, v_{i2}, \ldots v_{iD})$. In each iteration, the P vector of the particle with best fitness in the local neighborhood, designated g, and the P vector of the current particle are combined to adjust the velocity along each dimension and a new position of the particle is determined using that velocity. The two basic equations which govern the working of PSO are that of velocity vector and position vector given by:

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

$$x_{id} = x_{id} + v_{id} \quad (2)$$

The first part of (1) represents the inertia of the previous velocity, the second part is the cognition part and it tells us about the personal experience of the particle, the third part represents the cooperation among particles and is therefore named as the social component [10]. Acceleration constants c_1, c_2 [11] and inertia weight w [12] are the predefined by the user and r_1, r_2 are the uniformly generated random numbers in the range of [0, 1]. The computational steps of BPSO algorithm are illustrated below.

```
Step1: Initialization.
  For each particle i in the population:
  Step1.1: Initialize X[i] with Uniform distribution.
  Step1.2: Initialize V[i] randomly.
  Step1.3: Evaluate the objective function of X[i],
           and assigned the value to fitness[i].
  Step1.4: Initialize Pbest[i] with a copy of X[i].
  Step1.5: Initialize Pbest_fitness[i] with a copy
           of fitness[i].
  Step1.6: Initialize Pgbest with the index of the
           particle with the least fitness.

Step2: Repeat until stopping criterion is reached:
  For each particle i:
  Step 2.1: Update V[i] and X[i] according
            to equations (1) and (2).
  Step2.2: Evaluate fitness[i].
  Step2.3: If fitness[i] < Pbest_fitness[i] then
        Pbest[i] =X[i], Pbest_fitness[i] =fitness[i].
  Step2.4: Update Pgbest by the particle with current
           least fitness among the population.
```

3. Proposed QI-PSO Algorithm

The proposed QI-PSO algorithm is a simple and modified version of BPSO in which we have induced the concept of reproduction (borrowed from EA).

The basic idea of this paper is loosely based on the concept of polygamy (still prevalent in some of the tribal regions of India), where the leader of the tribe is allowed to have more than one partner. The QI-PSO algorithm works in the following manner. In every iteration, a leader of the swarm (tribe) is selected according to the performance and the one giving the least function value is taken to be the leader of the tribe. Once the leader is elected, then its partners (or mates) are chosen randomly from the remaining members of the tribe. For producing a new offspring from the union of three parents, we have defined a new crossover operator.

The quadratic crossover operator, based on quadratic interpolation [13], suggested in this paper is a nonlinear crossover operator which produces an offspring lying at the minimum of the quadratic curve passing through the three selected parents. The QI-PSO algorithm starts like the usual BPSO using equations (1) and (2). After that the crossover operator is invoked and the new particle is accepted in the swarm only if it is better than the worst particle present in the swarm. The process is repeated iteratively until a better solution is obtained. It uses a = X_{min}, (the leader having minimum function value) and two other randomly selected particles {b, c} (a, b and c are different particles) from the swarm (tribe) to determine the coordinates of the new particle $\tilde{x}^i = (\tilde{x}^1, \tilde{x}^2, \ldots\ldots \tilde{x}^n)$, where

$$\tilde{x}^i = \frac{1}{2} \frac{(b^{i^2} - c^{i^2}) * f(a) + (c^{i^2} - a^{i^2}) * f(b) + (a^{i^2} - b^{i^2}) * f(c)}{(b^i - c^i) * f(a) + (c^i - a^i) * f(b) + (a^i - b^i) * f(c)} \quad (3)$$

The nonlinear nature of the quadratic crossover operator used in this work helps in finding a better solution in the search space. The computational steps of the algorithm are illustrated below.

```
Initialize the swarm
Do
   For each particle
        Update velocity
        Update position
        Update personal best
        Update Global best.
   End for
   Find a New Particle using equation (3)
   If the New Particle is better than Worst Particle then
        Replace the Worst Particle by the New Particle
While (stopping condition is reached)
```

Table 1. Numerical benchmark problems

Function	Dim	Range	Min. value
$f_1(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	30/50	[-5.12,5.12]	0
$f_2(x)=\sum_{i=1}^{n}x_i^2$	30/50	[-5.12,5.12]	0
$f_3(x)=\frac{1}{4000}\sum_{i=0}^{n-1}x_i^2-\sum_{i=0}^{n-1}\cos(\frac{x_i}{\sqrt{i+1}})+1$	30/50	[-600,600]	0
$f_4(x)=-\sum_{i=1}^{n}x_i\sin(\sqrt{\lvert x_i\rvert})$	30/50	[-500,500]	-418.9829*n
$f_5(x)=\frac{\pi}{n}\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(y_{i+1}\pi)]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ *	30/50	[-50,50]	0
$f_6(x)=(0.1)\{\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}((x_i-1)^2(1+\sin^2(3\pi x_{i+1})))+(x_n-1)(1+\sin^2(2\pi x_n))\}+\sum_{i=0}^{n-1}u(x_i,5,100,4)$ *	30/50	[-50,50]	-1.1428
$f_7(x)=\sum_{i=0}^{n-1}\lvert x_i\rvert+\prod_{i=0}^{n-1}\lvert x_i\rvert$	30/50	[-10,10]	0
$f_8(x)=\max\lvert x_i\rvert,\ \ 0\le i<n$	30/50	[-100,100]	0
$f_9(x)=\sum_{i=0}^{n-1}\lfloor x_i+1/2\rfloor^2$	30/50	[-100,100]	0
$f_{10}(x)=(\sum_{i=0}^{n-1}(i+1)x_i^4)+rand[0,1]$	30/50	[-1.28,1.28]	0
$f_{11}(x)=\sum_{i=0}^{n-1}(\sum_{j=0}^{i}x_i)^2$	30/50	[-100,100]	0
$f_{12}(x)=\sum_{i=0}^{n-1}100(x_{i+1}-x_i^2)^2+(x_i-1)^2$	30/50	[-30,30]	0
$f_{13}(x)=\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}((x_i-1)^2(1+\sin^2(3\pi x_{i+1})))+(x_n-1)(1+\sin^2(2\pi x_n))$	30/50	[-10,10]	-21.5024
$f_{14}(x)=(\sum_{i=1}^{n}x_i^2)^{1/4}[\sin^2(50(\sum_{i=1}^{n}x_i^2)^{1/10})+1.0]$	30/50	[-32.767,32.767]	0
$f_{15}(x)=\frac{1}{n}\sum_{i=1}^{n}(x_i^4-16x_i^2+5x_i)$	30/50	[-5,5]	-78.3323
$f_{16}(x)=-\sum_{i=1}^{n}\sum_{j=1}^{5}j\sin((j+1)x_i+j)$	30/50	[-10,10]	-

Remarks 1. Functions sine and cosine take arguments in radians.

*Remark*s 2. The function u used in f_5 and f_6 and the values y_i used in f_5 are all defined below.

*f_5 : $y_i=1+\frac{1}{4}(x_i+1)$

*f_5, f_6 : $u(x,a,b,c)=b(x-a)^c$ if $x>a$, $u(x,a,b,c)=b(-x-a)^c$ if $x<-a$, else $u(x,a,b,c)=0$

Table 2. Numerical results (dimension: 30)

Fun-ction	BPSO			QI-PSO		
	Mean Best Fitness	Diversity	Stddev	Mean Best Fitness	Diversity	Stddev
*f*1	81.58668	1.075614e-08	35.570923	0.994954	5.948432e-11	2.354492
*f*2	2.62144	1.696744e-96	7.86432	2.523604e-45	3.203878e-20	6.505185e-45
*f*3	0.035265	8.958573e-08	0.029109	0.015979	5.454222e-08	0.013563
*f*4	-8406.742218	4.550712e-06	595.779702	-9185.074692	4.066819e-06	760.633113
*f*5	5.505851e-13	1.471375e-13	2.757006e-25	5.505851e-13	7.84124e-13	1.913863e-25
*f*6	-1.147328	4.805696e-08	0.003296	-1.149339	3.503531e-08	0.003296
*f*7	4.000000	7.361003e-124	4.89897	5.020939e-30	9.011109e-30	1.271869e-29
*f*8	0.000244	0.003937	0.000187	0.000148	7.496987e-05	8.128944e-05
*f*9	0.000000	1.678432	0.000000	0.000000	0.6674	0.000000
*f*10	24.532977	0.200206	14.640568	0.454374	0.192434	0.353778
*f*11	8.103896e-06	1.000654e-05	3.623224e-06	2.614209e-40	1.357432e-20	5.848717e-40
*f*12	99.795759	3.655268	438.491315	77.916591	0.06907	166.009829
*f*13	-15.301387	5.412045e-08	9.36361	-21.502311	6.948980e-08	3.552714e-15
*f*14	3.531709	0.730954	2.484447	0.974427	0.000265	0.283325
*f*15	-77.012904	1.799400e-07	1.049466	-77.201394	1.400101e-07	0.565469
*f*16	-155.613795	1.508819e-08	17.664877	-179.040627	1.179486e-08	30.28809

Table 3. Numerical results (dimension: 50)

Function	BPSO		QI-PSO	
	Best Fitness	Diversity	Best Fitness	Diversity
*f*1	192.383628	2.412312	4.77049e-17	2.596399e-09
*f*2	5.768702e-08	1.492921e-07	2.161201e-73	1.157111e-36
*f*3	0.019661	8.376331e-08	0.009850	8.484164e-08
*f*4	-13790.157053	5.204209e-06	-14268.46157	6.065605e-06
*f*5	2.535693	4.611654e-06	3.303511e-13	1.506118e-13
*f*6	-1.150438	6.142788e-08	-1.150438	7.384989e-08
*f*7	9.782400	3.776982e-15	6.552311e-51	2.937678e-50
*f*8	1.514302	27.53219	0.004083	0.017026
*f*9	0.000000	1.477857	0.000000	0.659437
*f*10	110.563669	0.245941	0.672779	0.266817
f_{11}	7.462966e-01	4.993772e-05	9.431749e-71	2.867124e-36
f_{12}	22.191719	2.324115	10.954427	0.036353
f_{13}	-11.000000	5.633709e-08	-21.502311	8.306594e-08
f_{14}	10.012826	24.758948	3.091645	0.002222
f_{15}	-77.234331	2.398338e-07	-77.766863	3.100701e-07
f_{16}	-204.194616	1.54925e-08	-239.204493	1.853805e-08

4. Experimental Settings

In order to make a fair comparison of BPSO and QI-PSO, we fixed the same seed for random number generation so that the initial swarm population is same for both the algorithms. The number of particles in the swarm (swarm size) is taken as 30. A linearly decreasing inertia weight is used which starts at 0.9 and ends at 0.4, with the user defined parameters c_1=2.0 and c_2=2.0.

We have also evaluated diversity, an important aspect for checking the efficiency of the swarm, for

both the algorithms. The diversity measure [14] of the swarm is taken as:

$$Diversity(S(t)) = \frac{1}{n_s}\sum_{i=1}^{n_s}\sqrt{\sum_{j=1}^{n_x}(x_{ij}(t) - \overline{x_j(t)})^2} \qquad (4)$$

where S is the swarm, $n_s = |S|$ is the swarm size, n_x is the dimensionality of the problem, x_{ij} is the j^{th} value of the i'th particle and $\overline{x_j(t)}$ is the average of the j^{th} dimension over all particles, i.e.

$$\overline{x_j(t)} = \frac{\sum_{i=1}^{n_s} x_{ij}(t)}{n_s}.$$

For each algorithm, the maximum number of iterations allowed was set to 30,000 for 30 dimensions and 100,000 for 50 dimensions. For 30 dimensions, a total of 10 runs for each experimental setting were conducted and the average fitness of the best solutions throughout the run was recorded. For 50 dimensions only a single run was performed and the best fitness was recorded. Each algorithm was tested with all of the numerical benchmark problems reported in Table 1.

5. Experimental Results

In order to check the compatibility of the proposed QI-PSO algorithm we have tested it on a suite of 16 benchmark problems given in Table 1. The test bed comprises of a variety of problems ranging from a simple spherical function to highly multimodal functions and also a noisy function (with changing optima). In Table 2, we have shown the results of problems with dimension 30 and in Table 3, are given the results of all problems with increased dimension 50 in terms of best fitness value and diversity. In Table 4 and 5, we have shown the performance improvement of QI-PSO with BPSO. Figures 1 - 4 show the mean best fitness curves for the first four benchmark problems.

From the numerical results presented in Tables 2, 3, 4 and 5, it is evident that QI-PSO is a clear winner. It gave a better performance in comparison to the BPSO algorithm for problems with dimension 30 in all the cases except for function f_5 and f_9. In this case both BPSO and QI-PSO performs the same. However it is quite interesting to note that QI-PSO gave a much better performance for the same function (f_5) when the dimension is increased to 50. In terms of percentage of improvement (of average fitness function value), the results favor QI-PSO particularly for higher dimension (50).

6. Conclusions

In this we proposed a simple and modified version of BPSO by incorporating a newly defined crossover operator in it. The empirical results show that the proposed algorithm is quite competent for solving problems of dimensions up to 50.

Metaphorically, this article uses the concept of polygamy among various Indian tribes, where the leader of the tribe keeps two partners to produce an offspring.

However, we would also like to say that since the results quoted here are based on the empirical study only and we are yet to explore theoretical relevance of the proposed QI-PSO, making any concrete judgment does not sound fair. Moreover the platform on which we have conducted the experiments is quite narrow. We are working on the problems with higher dimensions (100 and above) and in future we would try more complex optimization problems particularly problems in dynamic environment.

Table 4. Improvement (%) of QI-PSO in comparison with BPSO (dimension: 30)

Function	Improvement (%)	Function	Improvement (%)
$f1$	98.780495	$f9$	0.000000
$f2$	100	$f10$	98.147905
$f3$	54.688785	f_{11}	100
$f4$	9.258432	f_{12}	21.923946
$f5$	0.000000	f_{13}	40.525241
$f6$	0.175277	f_{14}	72.409193
$f7$	100	f_{15}	0.244751
$f8$	39.344262	f_{16}	15.054469

Table 5. Improvement (%) of QI-PSO in comparison with BPSO (dimension: 50)

Function	Improvement (%)	Function	Improvement (%)
$f1$	100	$f9$	0.000000
$f2$	100	$f10$	99.391501
$f3$	49.900819	f_{11}	100
$f4$	3.468449	f_{12}	50.637321
$f5$	100	f_{13}	95.475555
$f6$	0.000000	f_{14}	69.123153
$f7$	100	f_{15}	0.689502
$f8$	110.877709	f_{16}	17.145348

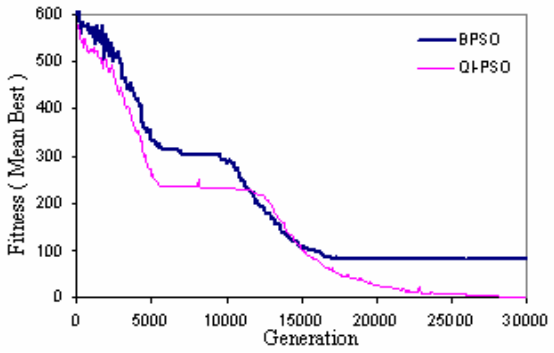

Figure 1. Performance comparison for f_1

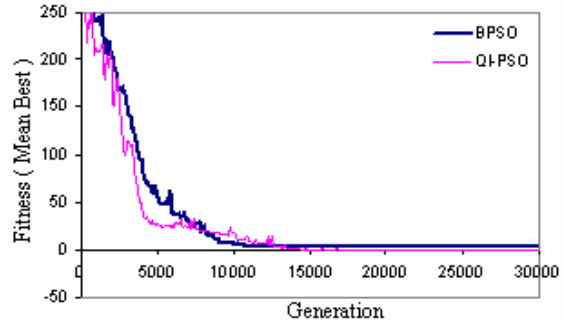

Figure 2. Performance comparison for f_2

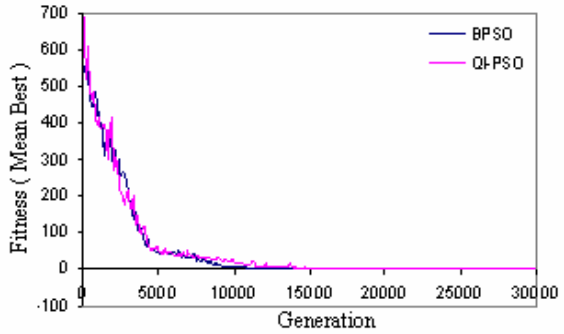

Figure 3. Performance comparison for f_3

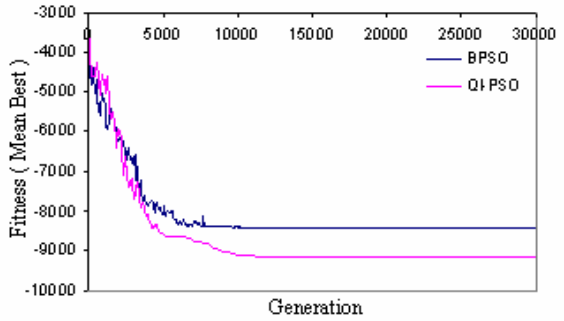

Figure 4. Performance comparison for f_4

7. References

[1] Angeline P. J., "Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference," The 7th Annual Conference on Evolutionary Programming, San Diego,USA,(1998).

[2] X. Hu, R. C. Eberhart, and Y. Shi, "Swarm Intelligence for Permutation Optimization: A Case Study on n-Queens problem", In Proc. of IEEE Swarm Intelligence Symposium (2003), pp. 243 – 246.

[3] V. Miranda and N. Fonseca, "EPSO – Best-of-two-worlds Meta-heuristic Applied to Power System problems", In Proc. of the IEEE Congress on Evolutionary Computation, Vol. 2, (2002), pp. 1080 – 1085.

[4] V. Miranda and N. Fonseca, "EPSO – Evolutionary Particle Swarm Optimization, a New Algorithm with Applications in Power Systems", In Proc. of the Asia Pacific IEEE/PES Transmission and Distribution Conference and Exhibition, Vol. 2, (2002), pp. 745 – 750.

[5] T-O. Ting, M. V. C. Rao, C. K. Loo, and S-S. Ngu, "A New Class of Operators to Accelerate Particle Swarm Optimization ", In Proc. of the IEEE Congresson Evolutionary Computation, Vol. 4, (2003), pp. 2406 – 2410.

[6] X. Yao and Y. Liu, "Fast Evolutionary Programming", In L. J. Fogel, P. J. Angeline, and T. B. Back, editors, Proceedings of the Fifth Annual Conference on Evolutionary Programming, MIT Press, (1996), pp. 451 – 460.

[7] X. Yao, Y. Liu and G.Lin, "Evolutionary Programming made Faster", IEEE Transactions on Evolutionary Computation, Vol. 3(2), (1999), pp. 82 – 102.

[8] P. J. Angeline, "Using Selection to Improve Particle Swarm Optimization", In Proc. of the IEEE Congress on Evolutionary Computation, IEEE Press, (1998), pp. 84 – 89.

[9] M. Clerc, "Think Locally, Act Locally: The Way of Life of Cheap-PSO, an Adaptive PSO", Technical Report, http: // clerc.maurice.free.fr/pso/, (2001).

[10] Kennedy, J., "The Particle Swarm: Social Adaptation of Knowledge," IEEE International Conference on Evolutionary Computation (Indianapolis, Indiana), IEEE Service Center, Piscataway, NJ,(1997), pg.303-308.

[11] R.C. Eberhart, Y. Shi, "Particle Swarm Optimization: developments, Applications and Resources," IEEE International Conference on Evolutionary Computation, (2001), pg. 81 -86.

[12] Shi, Y. H., Eberhart, R. C., "A Modified Particle Swarm Optimizer," IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, (1998), pg. 69 – 73.

[13] M. M. Ali and A. Torn, "Population Set Based Global Optimization Algorithms: Some Modifications and Numerical Studies", www.ima.umn.edu/preprints/, (2003).

[14] Engelbrecht, A. P., "Fundamentals of Computational Swarm Intelligence", John Wiley & Sons Ltd., (2005).