A New PSO Algorithm with Crossover Operator for Global Optimization Problems

Millie Pant[1], Radha Thangaraj[1], and Ajith Abraham[2]

[1] Department of Paper Technology, IIT Roorkee, India
millifpt@iitr.ernet.in, t.radha@ieee.org
[2] Center of Excellency for Quantifiable Quality of Service, Norwegian University of Science and Technology, Norway
ajith.abraham@ieee.org

Abstract. This paper presents a new variant of Particle Swarm Optimization algorithm named QPSO for solving global optimization problems. QPSO is an integrated algorithm making use of a newly defined, multiparent, quadratic crossover operator in the Basic Particle Swarm Optimization (BPSO) algorithm. The comparisons of numerical results show that QPSO outperforms BPSO algorithm in all the twelve cases taken in this study.

Keywords: Particle Swarm Optimization, Crossover, Global Optimization, Quadratic Interpolation.

1 Introduction

Evolutionary Algorithms (EA) and Particle Swarm Optimization (PSO) are perhaps the two most common stochastic techniques for solving global optimization problems. Both the techniques have proved their mettle in solving the complex optimization problems (test as well as real life problems). 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.
- The quality of the solutions are improved using techniques inspired from real world phenomenon 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.

Both the algorithms have their share of weaknesses and strengths and it is quite natural to think that the fusion of the two will result in an algorithm which has properties of both the algorithms. 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]). The objective of this paper is to see the performance of a BPSO after including a crossover operator in it. For this purpose we developed a new crossover operator called quadratic crossover operator. It makes use of three swarm particles to produce a new particle that lies on the point of minima of the quadratic curve (hence the name quadratic crossover) passing through the three chosen particles.

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

2 Basic Particle Swarm Optimization

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:

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

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

The first part of equation (1) represents the inertia of the previous velocity, the second part is the cognition part and it tells us about the personal thinking 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].

3 Proposed QPSO Algorithm

The proposed QPSO algorithm is a simple and modified integrated version of BPSO and EA. The quadratic crossover operator suggested in this paper is a nonlinear multi parent crossover operator which makes use of three particles (parents) of the swarm to produce a particle (offspring) which lies at the point of minima of the quadratic curve passing through the three selected particles. The new particle is accepted in the swarm irrespective of the fact whether it better or worse than the worst particle present in the swarm. In this way the search is not limited to the region around the current best location but is in fact more diversified in nature.

The quadratic interpolation [13] uses a = X_{min}, the particle having minimum function value and two other randomly selected particles {b, c} from the swarm to determine the coordinates of the new particle $\tilde{x}^i = (\tilde{x}^1, \tilde{x}^2, \ldots\ldots, \tilde{x}^n)$, where

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

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 proposed QPSO are given in Fig. 1.

```
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)
    Replace the Worst Particle by the New Particle
While (stopping condition is reached)
```

Fig. 1. Flow QPSO Algorithm

4 Experimental Settings

In order to make a fair comparison of BPSO and QIPSO, 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 considered the measurement of diversity, which is an important aspect for checking the efficiency of the swarm. High diversity generally implies a better balance between exploration and exploitation. The diversity measure [14] of the swarm is taken as:

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

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.

$$\overline{x_j(t)} = \frac{\sum_{i=1}^{n_s} x_{ij}(t)}{n_s}. \tag{5}$$

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 value was recorded.

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
$f_4(x)=-\sum_{i=1}^{n}x_i\sin(\sqrt{\mid x_i \mid})$	30/50	[-500,500]	-418.9829*n
$f_5(x)=\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}((x_i-1)^2(1+\sin^2(3\pi x_{i+1})))+(x_n-1)(1+\sin^2(2\pi x_n))$	30/50	[-10,10]	-21.5024
$f_6(x)=-\sum_{i=1}^{n}\sum_{j=1}^{5}j\sin((j+1)x_i+j)$	30/50	[-10,10]	-
$f_7(x)=\sum_{i=0}^{n-1}\mid x_i\mid+\prod_{i=0}^{n-1}\mid x_i\mid$	30/50	[-10,10]	0
$f_8(x)=\max\mid x_i\mid,\quad 0\le i<n$	30/50	[-100,100]	0
$f_9(x)=(\sum_{i=1}^{n}x_i^2)^{1/4}[\sin^2(50(\sum_{i=1}^{n}x_i^2)^{1/10})+1.0]$	30/50	[-32.767,32.767]	0
$f_{10}(x)=(\sum_{i=0}^{n-1}(i+1)x_i^4)+rand[0,1]$	30/50	[-1.28,1.28]	0
$f_{11}(x)=\frac{1}{n}\sum_{i=1}^{n}(x_i^4-16x_i^2+5x_i)$	30/50	[-5,5]	-78.3323
$f_{12}(x)=\frac{\pi}{n}\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(y_{i+1}\pi)]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ *	30/50	[-50,50]	0

Remark1: Functions sine and cosine take arguments in radians.

$*f_{12}$: $y_i = 1+\frac{1}{4}(x_i+1)$

$u(x,a,b,c)=b(x-a)^c$, if $x>a$

$u(x,a,b,c)=b(-x-a)^c$, if $x<-a$

$u(x,a,b,c)=0$, if $-a\le x\le a$

5 Experimental Results

In order to check the compatibility of the proposed QPSO algorithm we have tested it on a suite of 12 standard bench mark problems, generally used for testing the efficiency of global optimization algorithms, given in Table 1. The test bed comprises 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 problems with increased dimension 50 in terms of best fitness

Table 2. Numerical results (dimension: 30)

Function	BPSO		QPSO	
	Mean Best Fitness	Diversity	Mean Best Fitness	Diversity
*f*1	8.158668e+01	1.075614e-08	5.97167e-01	1.639511e-05
*f*2	2.621440e+00	1.696744e-96	8.517991e-43	2.92223e-20
*f*3	3.526500e-02	8.958573e-08	2.940000e-02	9.426306e-05
*f*4	-8.406742e+03	4.550712e-06	-9.185054e+03	1.078529e+02
*f*5	-1..301387e+01	5.412045e-08	-2.150231e+01	1.082100e-01
*f*6	-1.556138e+02	1.508819e-08	-3.258289e+02	5.055760e-01
*f*7	4.000000e+00	7.361003e-124	1.154568e-20	8.341826e-05
*f*8	2.440000e-04	3.937000e-03	3.510000e-04	5.535000e-03
*f*9	3.531709e+00	7.309540e-01	8.585330e-01	2.600000e-04
*f*10	2.453298e+01	2.002060e-01	4.540630e-01	3.234900e-01
*f*11	-7.701290e+01	1.799400e-07	-7.795535e+01	5.165870e-01
*f*12	5.505851e-13	1.471375e-13	5.505851e-13	9.682600e-02

Table 3. Numerical results (dimension: 50)

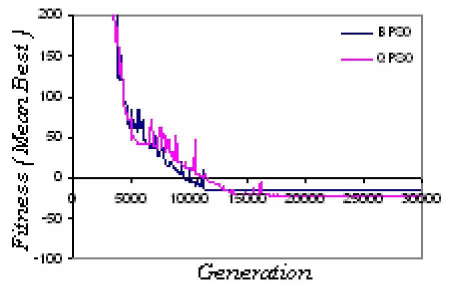
Function	BPSO		QPSO	
	Best Fitness	Diversity	Best Fitness	Diversity
*f*1	1.923836e+02	2.412312e+00	0.000000e+00	8.158896e-11
*f*2	5.768702e-08	1.492921e-07	9.714999e-74	9.735866e-37
*f*3	1.966100e-02	8.376331e-08	2.466000e-03	3.654810e-01
*f*4	-1.379016e+04	5.204209e-06	-1.407498e+04	1.464926e+02
*f*5	-1.100000e+01	5.633709e-08	-2.150231e+01	1.596100e-02
*f*6	-2.041946e+02	1.54925e-08	-5.243369e+02	5.555140e-01
*f*7	9.782400e+00	3.776982e-15	7.651628e-37	1.238780e-01
*f*8	1.514302e+00	2.753219e+01	4.875000e-03	1.240090e-01
*f*9	1.001283e+01	2.475895e+01	2.509801e+00	2.387241e+00
*f*10	1.105637e+02	2.45941oe-01	6.736030e-01	3.990910e-01
*f*11	-7.723433e+01	2.398338e-07	-7.776686e+01	1.11159e+00
*f*12	2.535693e+00	4.611654e-06	3.303511e-13	3.386880e-01

value. In Table 4, we have shown the performance improvement of QPSO with BPSO. Fig, 2 show the mean best fitness curves for selected benchmark problems.

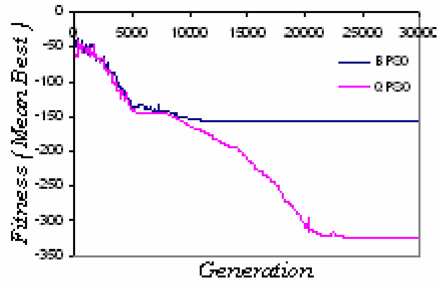
From the numerical results presented in Tables 2, 3, 4 and 5, it is evident that QPSO is a clear winner. It gave a better performance for all the problems with dimension 30 except for f_8. However it is quite interesting to note that QPSO gave a much better performance for the same function (f_8) when the dimension is increased to 50. In terms of percentage of improvement (of average fitness function value), the

Table 4. Improvement (%) of QPSO in comparison with BPSO

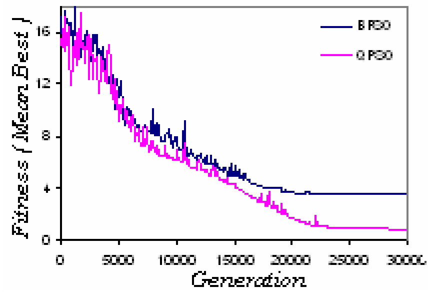
Function	Percentage Dim: 30	Percentage Dim: 50	Function	Percentage Dim: 30	Percentage Dim: 50
f_1	99.268058	100	f_7	100	100
f_2	100	100	f_8	-	99.678069
f_3	16.631221	87.457403	f_9	70.635432	74.934139
f_4	9.258188	2.065378	f_{10}	98.149173	99.390356
f_5	40.525241	95.475555	f_{11}	1.223753	0.689502
f_6	109.383101	156.782964	f_{12}	0.000000	100



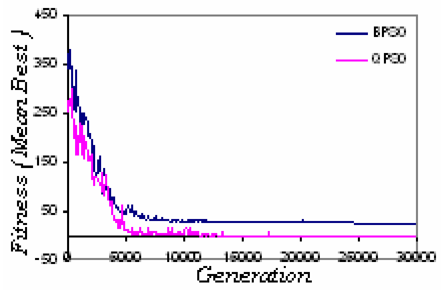
(a) f_5



(b) f_6



(c) f_9



(d) f_{10}

Fig. 2. Convergence graph for selected benchmark problems

results favor QPSO particularly for higher dimension (50). One interesting and noticeable fact about QPSO is that despite low diversity (although higher than BPSO in most of the cases) it gave good results.

6 Conclusion

In this research article, we proposed a simple and modified version of an integrated algorithm exploiting the features of PSO and EA. The novelty of the present work is the use of a nonlinear quadratic crossover operator and the manner in which it is applied. The new solution is accepted even if it is worse than the worst solution; this prevents the search space from contracting and thereby maintains diversity. The empirical results show that the proposed algorithm is quite competent for solving problems of dimensions up to 50. In future we shall also incorporate the phenomenon of mutation in the algorithm and will use it to solve the problems of higher dimensions.

Acknowledgments. The authors would like to thank the unknown referees for their comments, which helped in bringing the paper in more presentable format.

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. Hu, X., Eberhart, R. C., and Shi, Y.: Swarm Intelligence for Permutation Optimization: A Case Study on n-Queens problem. In Proc. of IEEE Swarm Intelligence Symposium, pp. 243 – 246 (2003).
3. Miranda, V., and Fonseca, N.: EPSO – Best-of-two-worlds Meta-heuristic Applied to Power System problems. In Proc. of the IEEE Congress on Evolutionary Computation, Vol. 2, pp. 1080 – 1085 (2002).
4. Miranda, V., and Fonseca, N.: 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, pp. 745 – 750 (2002).
5. Ting, T-O., Rao, M. V. C., Loo, C. K., and Ngu, S-S.: A New Class of Operators to Accelerate Particle Swarm Optimization. In Proc. of the IEEE Congresson Evolutionary Computation, Vol. 4, pp. 2406 – 2410 (2003).
6. Yao, X., and Liu, Y.: 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, pp. 451 – 460 (1996).
7. Yao, X., Liu, Y., and Lin, G.: Evolutionary Programming made Faster. IEEE Transactions on Evolutionary Computation, Vol. 3(2), pp. 82 – 102 (1999).
8. Angeline, P. J.: Using Selection to Improve Particle Swarm Optimization. In Proc. of the IEEE Congress on Evolutionary Computation, IEEE Press, pp. 84 – 89 (1998).
9. Clerc, M.: 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, pg.303-308 (1997).
11. Eberhart, R.C., and Shi, Y.: Particle Swarm Optimization: developments, Applications and Resources. IEEE International Conference on Evolutionary Computation, pg. 81 -86 (2001).
12. Shi, Y. H., and Eberhart, R. C.: A Modified Particle Swarm Optimizer. IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, pg. 69 – 73 (1998).
13. Ali, M. M., and Torn, A.: 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).