Opposition Based Initialization in Particle Swarm Optimization (O-PSO)

Hajira Jabeen
National University of Computer and Emerging Sciences, Islamabad, 44000, Pakistan
hajira.jabeen@nu.edu.pk

Zunera Jalil
National University of Computer and Emerging Sciences, Islamabad, 44000, Pakistan
zunera.jalil@nu.edu.pk

Abdul Rauf Baig
National University of Computer and Emerging Sciences, Islamabad, 44000, Pakistan
rauf.baig@nu.edu.pk

ABSTRACT

Particle Swarm Optimization, a population based optimization technique has been used in wide number of application areas to solve optimization problems. This paper presents a new algorithm for initialization of population in standard PSO called Opposition based Particle Swarm Optimization (O-PSO). The performance of proposed initialization algorithm is compared with the existing PSO variants on several benchmark functions and the experimental results reveal that O-PSO outperforms existing approaches to a large extent.

Categories and Subject Descriptors

I.2.m Miscellaneous

General Terms

Algorithms, Performance, Experimentation.

Keywords

PSO, Opposition based learning, Swarm intelligence, Optimization.

1. INTRODUCTION

The Particle Swarm Optimization (PSO) is a population-based optimization method first proposed by Kennedy and Eberhart [1]. The algorithm simulates the behavior of bird flock flying together in n-dimensional space in search of some optimum place, adjusting their movements and distances in the constrained environment.

PSO can be viewed as a mid-level form of artificial and natural life which has proved to be successful on a wide range of real life problems like function optimization [2], pattern recognition [3], learning game environments [4] and others. PSO is easy to implement and does not require any problem specific information like gradient. It can be applied for the tasks where problem specific information is either unavailable or computationally expensive to obtain. In PSO, each particle represents a potential solution to an optimization problem. The concept of particle swarm optimization is optimizing these potential solutions by flying (moving) through the search hyperspace, accelerating towards "better" solutions.

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.
ACM 978-1-60558-505-5/09/07.

Initialization of population plays an important role in any optimization algorithm. It has been proven [5] that the random selection of solutions from a given solution space can result in exploiting the fruitless areas of the search space. Intelligent initialization methods based on realistic approaches are required for efficient results. It has been shown empirically [5] that random selection in case of using opposite population lowers the chances of exploring barren areas of search space. In this paper, the opposition based initialization technique has been proposed in which opposites or opponents of basic population are included in the initial population, and the potential ones survive.

This paper is organized as follows. We first introduce the Particle Swarm Optimization problem, the issues of initialization and performance. In the next section, the existing variants of PSO and opposition based learning (OBL) are briefly mentioned. In section 3, Opposition based PSO algorithm (O-PSO) has been proposed with description of initialization approach. Section 4 gives the simulation results and the final section concludes the paper highlighting the directions for future research.

2. LITERATURE REVIEW

A swarm in PSO contains multiple particles, where each particle maintains certain characteristics. One of the characteristic is a particle's current position which is represented by an n-dimensional vector corresponding to a potential solution of the function to be optimized. Each particle preserves an n-dimensional vector representing velocity which keeps track of the movement speed and direction. Another vector representing best position of particle is also reserved. Besides above mentioned vectors, each particle keeps its current fitness, which is obtained by evaluating the fitness function for each particle for its current position [6].

In past, considerable research has been done for optimization and efficient working of PSO. Several parameters have been introduced to improve the performance of PSO. Two important parameters are constriction coefficients and inertia weight. Constriction coefficients set the proportion to which we admire the previous best position of a particle and the global best particle of the swarm during the movement of one particular particle. The inertia weight determines the step size for movement.

Shi and Eberhart [7] introduced the concept of linearly decreasing inertia weight. A fuzzy method to change the inertia weight nonlinearly is proposed in [8]. In [9] the value of inertia weight is set at zero, except at the time of re-initialization. By

analyzing the convergence behavior of the PSO, a PSO variant with a constriction factor is introduced by Clerc and Kennedy [10]. Constriction factor guarantees the convergence and improves the exploration ability of swarm. Optimal values of constriction coefficients and inertia weights are proposed by Clerc [11] are 0.7298 and C_0=C_1=1.49618.

To improve the performance of PSO, another research focus has been variations in PSO topology. Keneddy [12] proposed that PSO with smaller neighborhood performs better on complex problems and larger neighborhood would perform better on simpler problems. Suganthan [13] suggests dynamic neighborhood that increases until it includes all the particles of the swarm.

Parsopoulos and Vrahatis used a combination of the global version and local version to make a unified particle swarm optimizer (UPSO) [14]. Mendes and Kennedy introduce a fully informed PSO [15], in which all the neighbors of the particle are used to update the velocity. The influence of each particle on its neighbors is weighted based on its fitness value and the neighborhood size. Some researchers have also used hybrid models of PSO [16].

The idea of opposition based learning is proposed by Tizoshi [17] which has been incorporated in several machine learning algorithms like opposition based reinforcement learning [18], opposition based differential evolution [5]and opposition based genetic algorithm [19]. Opposition based differential evolution [20] uses opposition based initialization scheme in which opposites of initial population are created and best from the combined population are chosen for evolution. The actual process of differential evolution is augmented by the opposition phase. It has been proved that opposition based learning process increases the convergence speed thus the evolution process accelerates.

Another method of opposition based differential evolution with jumping phenomena was proposed in [21] where a jumping rate was introduced and the individuals in a population were allowed to jump towards its opposite, once the jumping probability was met.

A method incorporating opposition based learning in PSO has been proposed by Wang [22]. The method uses opposition based learning and dynamic cauchy based mutation to avoid premature convergence in standard PSO.

The emphasis of our paper is to study the affect of opposition based initialization of swarm particles in standard PSO algorithm. The algorithm is elucidated in the next section.

3. PROPOSED ALGORITHM

Extensive and continuous effort has been done for optimization of PSO algorithm to make it efficient in solving various types of problems. It has been observed that the initialization of population plays an important role in the search process of evolutionary as well as swarm based algorithms. Better initialization tends to search efficiently through the hyperspace of desired solution and in case of bad initialization; algorithms may search in unwanted areas and may fail to converge. Initialization of population is very important for all optimization problems; however no significant research has been made in this area. We propose a new technique for initialization of population in Particle Swarm Optimization algorithm.

3.1 Opposition-based Initialization

In PSO, the swarm should truly represent the entire search space so that solution found from that swarm is optimal. Better and careful initialization based on priori information can lead to better results.

The social phenomenon of good and bad says, if one person is good then his/her opponent is bad. It rarely occurs that two opponents are totally good and totally bad at the same time. This is purely natural and in this paper we exploit this natural trait of human beings and propose a similar method for population initialization of PSO. The proposed approach to population initialization used the opposition based method in which the population and its opposite population is taken as input. The fitness of both populations are evaluated and only the fitter ones from both are selected as particles. The description of concepts used in the proposed algorithm is as under:

Particle: A swarm particle P_i in PSO can be defined as:

$P_i \in [a, b]$; where i=1,2,3,…..,D, and a,b $\in$ R

D represents dimensions and R represents real numbers

Opposite particle: Every particle P_i has a unique opposite P_{opi} in initially defined hyperspace which can be defined as:

$P_{opi} = a+ b- P_i$; where i=1,2,3,…,D and a,b $\in$ R (i)

D represents the number of dimensions and R represents real numbers. For a single dimensional particle, P_i =3 $\in$ [0, 10]; the opposite will be calculated as P_{opi} = (0+10)-3= 7.

Fitness Function: is the function which quantifies the optimality of a solution and is usually the objective function.

Each particle in the swarm maintains three attributes of D dimensions. These are i) current position ii) local best position and iii) velocity

Local best (l-best): is the best position that a particle has visited yielding the highest fitness value for that particle. This value can be smallest for a minimization task.

Global best (g-best): is the position where the best fitness is achieved by any particle of the swarm evolved so far.

Velocity Update: Velocity is a D-dimensional vector that determines the movement speed and direction of the particle. The velocity is updated by the following equation:

$$V_i = \omega V_i + C_0 rand(0,1)X_{lbest} - X_i + C_1 rand(0,1)X_{gbest} - X_i \quad \text{(ii)}$$

Where w is the inertia weight, C_0 and C_1 are the constriction coefficients, x_{lbest} and x_{gbest} are the local and global bests of the particle.

Position Update: Each particle (potential solution) updates its position to move in the solutions hyperspace in search of optimal solution. All the particles in a swarm move stochastically for optimal positions and update their positions using the following equation:

$$X_i = X_i + V_i \quad \text{(iii)}$$

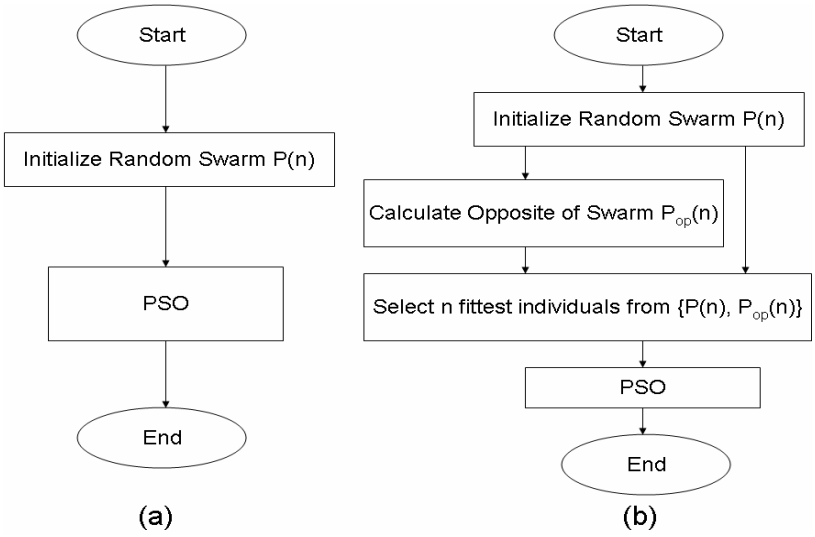

Figure 1. PSO with (a) Random population initialization and (b) Opposition-based population initialization

3.2 The O-PSO Algorithm

The following opposition-based population initialization algorithm can be used for population initialization instead of random initialization:

1. Initialize swarm of particles P(N) randomly; N is the population size.
2. Calculate opposites of swarm $P_{OP}(N)$ using (i)
3. Evaluate the fitness of both populations (P and P_{OP}) based on fitness function (F).
4. Choose N fittest individuals from set { P(N) U P_{OP} (N)} as initial population based on fitness value.
5. Repeat
    a. Calculate g-best
    b. Repeat for each particle pi ϵ P_i
        i. Calculate p-best
        ii. Update velocity and position components using (ii) and (iii) respectively.
6. Until <termination condition>

In the above algorithm, the initial basic population of swarms is initialized randomly. Then the basic swarm is used to create its opposite swarm. Fitness of each individual solution (particle) is evaluated and the fitter ones from both are selected to find the optimal solution using standard PSO algorithm. The algorithm is terminated when the desired solution is found or iterations have been completed. Figure 1 shows the proposed approach of initialization. The standard PSO gets improved by doing initialization of population based on opposite numbers. The position and velocity components are updated using standard PSO formulas [11].

4. EXPERIMENTAL RESULTS

4.1 Benchmark functions

In order to test the performance of opposition-based particle swarm optimization algorithm, a test set with four non-linear functions is used. The first three functions (*f1, f2, and f3)* are frequently used as benchmarks in swarm intelligence literature [2], while *f4* is commonly used as benchmarks for evolutionary algorithms. These functions are listed in Table 1.

In Table 1, Variable x is a real valued n-dimensional point and x_i is the i^{th} element of that point.*f1* is a simple unimodal function called generalized sphere function. *f2* is a unimodal function known as hard to optimize and is called Rosenbrock function. *f3* is a multimodal function with many local minima set around global minima and is named generalized Ackley function. *f4* is a multimodal function hard to optimize. All these functions can work with variable number of dimensions of the variable x. These functions present minimization problems with global minima set at 0 values for each dimension.

Two experiment setups were made to test the performance of O-PSO with the existing PSO variants and to evaluate the effect of incorporating opposition in PSO compared to standard PSO initialization method.

4.2 Experimental Setup I

To evaluate the performance of O-PSO, it has been compared with existing PSO variants PSO1, PSO2 and PPO (Predator-Prey Optimizer). The detailed descriptions of these algorithms can be found in [2]. We have used the results presented by Silva for comparison.

The algorithm was run with three different dimensions 10, 20 and 30 for each test function. The maximum iteration limit was set to 1000, 1500 and 2000 for 10, 20 and 30 dimensions respectively.

Table 1. List of benchmark functions used for experiments

f1	Sphere Function	$F_1(x) = \sum_{i=1}^{n} x_i^2$
f2	Rosenbrock Function	$F_2(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$
f3	Ackley Function	$F_3(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$
f4	A MultiModal Function	$F_4(x) = 418.9829n + \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$

Table 2. Average best solutions of each experimental setting over 200 runs

Function	Dim	Iterations	PSO1	PSO2	PPO	O-PSO
f1	10	1000	115E-20 ± 5831E-21	521E-3 ± 102090E-73	143E-34 ± 1227E-34	2.15E-85 ± 7.74296E-85
	20	1500	500E-12 ± 228482E-12	212E-41 ± 31323E-42	990E-26 ± 10389E-26	2.39E-131 ± 1.92E-131
	30	2000	413E-08 ± 130741E-08	622E-25 ± 10182E-25	532E-21 ± 37682E-21	5.2E-173 ± 103E-32
f2	10	1000	11883096 ± 3402262	5114766 ± 16443999	73008 ± 1355563	21.6 ± 9.8
	20	1500	18502141 ± 4349773	6889530 ± 1755632	6746585 ± 1497499	68.81189 ± 22.68682
	30	2000	24134265 ± 5265796	15513395 ± 3523946	16397916 ± 3766825	99.66251218 ± 26.5075264
f3	10	1000	2564E-11 ± 60335E-12	006941 ± 004307	7832E-08 ± 12394E-08	1.81541816E-05 ± 1.2E-05
	20	1500	000823 ± 001613	047375 ± 009981	184E-06 ± 268697E-07	1.81541816E-05 ± 3.2E-05
	30	2000	021048 ± 007028	108448 ± 014345	1252E-5 ± 171359E-06	1.81715406E-05 ± 3.4E-05
f4	10	1000	41113915 ± 2370806	68912034 ± 3168839	9373165 ± 1381862	370327.6618 ± 968197.2302
	20	1500	120235393 ± 4789437	199120137 ± 6202794	38048637 ± 2546739	40424720.7 ± 1554872
	30	2000	230519333 ± 6872059	342618449 ± 7736661	72470529 ± 3799708	6644981.6 ± 13531597

For all conducted experiments, the parameters inertia weight (w), constriction coefficients (C_0 and C_1) are set to 0.7298, 1.49618 and 1.49618 respectively. These values were derived from Clerc's analysis [9].

The population of size 20 has been used. Other experimental parameters were kept same as in [2] to facilitate the comparison. All results have been averaged over 200 runs to be consistent for comparison purpose.

Table 2 shows the experimental results of O-PSO in comparison with PSO1, PSO2 and PPO. The O-PSO algorithm provides better results and shows the effectiveness of the proposed initialization technique. It can be clearly observed that O-PSO performs better than PSO1 and PSO2 on all functions. Also, O-PSO gives better results than PPO on all benchmark functions except *f3*. PSO1 performed better than other variants for *f3*.

All the PSO variants failed to converge to the optimal values for *f4*. The optimal values achieved by O-PSO are much better than all three versions of PSO.

4.3 Experimental Setup II

Besides comparing the performance of O-PSO with other variations of PSO, we also tested the performance of O-PSO with standard PSO (using random initialization approach).

Instead of including N opposite particles, we included N random particles, resulting in a random population of size 2N. We selected best N particles from this population for optimization and refer this approach as 2-PSO in experiments. The experimental parameters are motioned in Table 3.

Table 3. Parameter values

Parameters	Values
Population size	20
Error Tolerance	0.0001
Iterations	10000
Search space	[-100,100]
Dimensions	1000

We performed the experiments on three functions (*f1, f2 and f3*) only since all PSO variants fail to achieve the error tolerance for *f4*. Table 4 shows the number of iterations taken by O-PSO and 2-PSO to achieve the error tolerance mentioned in Table 3. The results have been averaged over 200 runs.

Table 4. Number of iterations taken by O-PSO and 2-PSO to achieve error tolerance

Function	O-PSO	2-PSO
F1	75.35	78.73
F2	72.3	76.15
F3	129.2	132.4

The results show that O-PSO reached the error tolerance in lesser number of iterations on all the functions as compared to 2-PSO. This proves that the population and anti-population has a positive effect on the convergence of PSO.

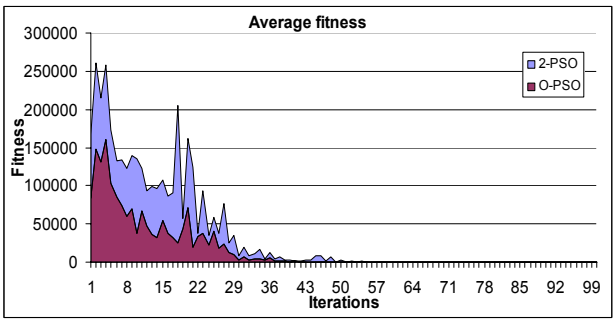

Figure 2. Average fitness of O-PSO and 2-PSO during convergence

Figure 2 shows the convergence properties of O-PSO as compared with 2-PSO. O-PSO not only starts from a better value but also converges faster to the optimal values as compared to 2-PSO.

5. CONCLUSIONS

This paper introduces a new initialization technique for initialization of population in particle swarm optimization algorithm. The initialized population considering opposites gives a better representation of search space and the solutions found are better as compared with standard PSO. Future work scenarios include considering PSO with different parameter variants with opposition based initialization and different fitness functions. Much more work need to be done to mimic the other behaviors of birds besides flocking e.g some birds may have more hunger rate and move faster than others. Another direction can be to incorporate opposite particles in dynamic environments.

6. REFERENCES

[1] J. Kennedy and R. C. Eberhart. Particle Swarm Optimization. *In Proceedings of IEEE International Conference on Neural Networks*, volume IV, pages 1942-1948, Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995.

[2] A, Silva, A. Neves and E.Costa. Chasing The Swarm: A Predator Pray Approach to Function Optimization. *In Proceedings of MENDEL 2002, 8th International Conference on Soft Computing* , Brno, Czech Republic. 2002.

[3] Mahamed G. H. Omran, *Particle Swarm Optimization Methods for Pattern Recognition and Image Processing,* PhD Thesis, University of Pretoria, Pretoria, November 2004.

[4] M,Leon and A.P.Engelbrecht, Learning To Play Games Using a PSO Based Competitive Learning Approach, *IEEE Transactions On Evolutionary Computation.* Vol 8, No 3, June 2004.

[5] Shahryar Rahnamayan, Hamid R. Tizhoosh, and Magdy M. A. Salama. Opposition-based Differential Evolution. *IEEE Transactions on Evolutionary Computation*, 2007

[6] F. van den Bergh, and A.P.Engelbrecht, A Cooperative Approach to Particle Swarm Optimization, *IEEE Transaction On Evolutionary Computation* ,Vol 8, No 3, June 2004.

[7] Y. Shi and R. C. Eberhart. A Modified Particle Swarm Optimizer. *In Proceedings of the IEEE Congress of Evolutionary Computation*, 1998, 69–73.

[8] Y. Shi and R. C. Eberhart. Particle Swarm Optimization with Fuzzy Adaptive Inertia Weight. *In Proceedings Workshop Particle Swarm Optimization*, Indianapolis, IN, 2001.

[9] A. Ratnaweera, S. Halgamuge, and H. Watson. Self-Organizing Hierarchical Particle Swarm Optimizer with Time Varying Accelerating Coefficients. *IEEE Transactions on Evolutionary Computation,* vol. 8, Jun. 2004, 240–255.

[10] Clerc, M., and Kennedy, J. The Particle Swarm Explosion, Stability, and Convergence In A Multidimensional Complex Space. *IEEE Transaction on Evolutionary Computation,* 6(1), 2002, 58–73.

[11] Riccardo Poli, James Kennedy and Tim Blackwell, Particle Swarm Optimization An Overview, *Swarm Intelligence*, 2007, 33-57.

[12] J. Kennedy. Effects of Neighborhood Topology on Particle Swarm Performance. *In Proceedings of Congress in Evolutionary Computation* 1999, 1931–1938.

[13] P. N. Suganthan. Particle Swarm Optimizer with Neighborhood Operator. *In Proceedings of Congress Evolutionary Computation*, Washington, DC, 1999, 1958-1962.

[14] K. E. Parsopoulos and M. N. Vrahatis,. UPSO: A Unified Particle Swarm Optimization Scheme. *In Lecture Series on Computational Sciences*, 2004,868–873.

[15] R. Mendes, J. Kennedy, and J. Neves .The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Transactions on Evolutionary Computation*, vol. 8, Jun. 2004, 204–210.

[16] Lovbjerg, M., Rasmussen, T. K. and Krink, T. Hybrid Particle Swarm Optimiser with Breeding and Subpopulations, *Proceedings of the third Genetic and Evolutionary Computation Conference (GECCO-2001).* 2001.

[17] H.R.Tizhoosh. Opposition-Based Learning: A New Scheme for Machine Intelligence. *Proceedings of International Conference on Computational Intelligence for Modeling Control and Automation -MCA'2005*, Vienna, Austria, vol. I, 2005, 695-701.

[18] Hamid R. Tizhoosh, Opposition-Based Reinforcement Learning, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol.10, No.4 pp. 578-585

[19] A.Iqbal, H.Jabeen, R.Baig, Opposition Based Genetic Algorithm with Jumping Phenomena, *The Second International Symposium on Intelligent Informatics, ISII 2009*,(submitted)

[20] S.Rahnamayan, H.R.Tizhoosh, M.M. Salama, "Opposition-Based Differential Evolution Algorithms", *IEEE Congress on Evolutionary Computation*, 2006

[21] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama,"Opposition-Based Differential Evolution (ODE) with Variable Jumping Rate," *Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence* (FOCI 2007)

[22] H. Wang and Y. Liu, Opposition-based Particle Swarm Algorithm with Cauchy Mutation, IEEE Congress on Evolutionary Computation, CEC, 2007.