International Journal of Computing and Network Technology

http://dx.doi.org/10.12785/ijcnt/010201

B-Positive Particle Swarm Optimization (B.P.S.O)

Muhammad Naveed Tabassum*, Rashed Meer, Zeshan Ali

Electrical Engineering Department, King Saud University, Riyadh, Saudi Arabia
Email Address: mtabassum@ksu.edu.sa

Received: 12 Dec. 2012, Revised: 8 Jan. 2013; Accepted: 15 Jan. 2013
Published online: 1 May 2013

Abstract: This paper introduces a comparatively new technique for Particle Swarm Optimization (P.S.O). The standard P.S.O technique is modified in a unique way to come up with B-positive Particle Swarm Optimization. The B.P.S.O which is simulated in Matlab lets the particles in a multi-dimensional space to move in an overall positive direction. In other words the particles are made to move from one side of the space to the other without negative (backward) displacement in search of the global best position. At the same time the displacement magnitude is slightly reduced randomly to discourage the particles from jumping out of the space boundary. The lost particles are randomly thrown around the then known best position, this in return saves a lot of time and effort resulting in improved overall simulation results. Five popular benchmark functions are used to evaluate the performance of B.P.S.O and the result in terms of mean and standard deviation values for global minimum and mean time per replica are compared with previous Standard P.S.O results. The B.P.S.O turns out to be more efficient in terms of optimum convergence and simulation speed.

Keywords: Particle swarm optimazation; swarm intelligence; artificial intelligence;

I. Introduction

Particle Swarm Optimization frequently abbreviated as PSO is a technique which was first introduced by Dr. Eberhart and Dr. Kennedy in 1995 [1]. It is basically a population based stochastic technique which is since being used to solve many optimization problems The PSO was originally inspired by behavioral models of fish schooling and bird flocking also known as swarm intelligence. This type of intelligence is based on socio-psychological principles which provide an insight of social behavior as well as contribute in engineering applications [1].

The computer algorithm modeled on this type of social behavior is initialized with simple software agents called particles. These particles which can also be considered as potential solutions scatter in a given space and change their positions and velocities by following the current optimum particles according to the same rules inspired by the behavior models of bird flocking.

Since its creation, PSO has been applied in many research and engineering application areas. One of the reasons that it is widely used in many areas is that it has been demonstrated and proved from time to time that PSO gets better results in a faster converging as well as cheaper way as compared to other available techniques. Another important reason is that it has fewer parameters as compared to other algorithms. One version, with some variations can be used to work for wide variety of applications. This makes PSO an attractive optimization approach as a whole.

II. CLASSICAL PARTICLE SWARM OPTIMAZATION

As the name suggests, the Classical Particle Swarm Optimization [1] means to optimize a problem by having a swarm (which is also called a population) of deserving solutions (which are also called the particles). The optimization is conducted by moving these particles in a bounded space using a developed mathematical formula. The two main parameters in the classical PSO are the particle's position and velocity.

The mathematical formula suggests that the particle's upcoming movement is influenced by its personal experience and the social collaborative experience. The personal experience is simply the particles best known position giving the best known solution whereas the social experience is the information that the particle has about the best known position in the whole swarm. The position of the particle is updated as new best positions are found. In other words the whole swarm gradually moves toward the best position of the space having the best possible solution.

The classical PSO algorithm [2] is as follows;

- Initialize the particles with random velocities and positions in a given Dimension.
- Compute the fitness of all particles using the desired benchmark function, choose the lowest one as global best and assign the current positions of all particles as their best.
- Calculate the next velocities and positions using the equations 1 and 2.
- Calculate the fitness of all particles using the updated velocities and positions. If the new fitness is less than the particles best fitness, the new fitness is considered the best one. In the same way the new position is considered the particle best position.
- In the same way if the new particle's best fitness is less than the overall global best fitness, it is considered the new best global fitness and its corresponding position is considered new global best position.
- Repeat third step for desired number of function evaluations.

The mathematical equations [2], [3] used to calculate the new velocity and position are as follows.

$$\vec{v}_i(j+1) = \vec{v}_i(j) + \vec{r}_i(j)(\vec{p}_i(j) - \vec{x}_i(j)) + \vec{r}_i(j))(\vec{g}(j) - \vec{x}_i(j)) \quad (1)$$

$$\vec{x}_i(j+1) = \vec{x}_i(j) + \vec{v}_i(j+1) \quad (2)$$

Where, $\vec{v}_i(j)$ is the velocity vector of particle i at iteration j

$\vec{x}_i(j)$ is the position vector of particle i at iteration j ,

$\vec{p}_i(j)$ is the D-dimensional personal best of particle i ,

$\vec{g}_i(j)$ is the D-dimensional global best of the whole swarm.

III. STANDART PARTICLE SWARM OPTIMAZATION

The algorithm of Standard PSO [4] is almost the same as classical PSO. The new positive constants that are added in the velocity update equation are c_1, c_2, and w. The new velocity update equation can be written as follows.

$$\vec{v}_i(j+1) = \underbrace{w\,\vec{v}_i(j)}_{\text{inertia part}} + \underbrace{c_1\,\vec{r}_i(j)(\vec{p}_i(j) - \vec{x}_i(j))}_{\text{cognition part}} + \underbrace{c_2\,\vec{r}_i(j)(\vec{g}(j) - \vec{x}_i(j))}_{\text{social part}} \quad (3)$$

Another new feature introduced in the Standard PSO is the velocity clamping. The purpose was to make particles to take reasonably sized steps to cover the space instead of bouncing about excessively. It has been proved that velocity clamping offers considerable improvements to the performance.

Velocity clamping is done by first calculating the range of the search space on each dimension, which is done by subtracting the lower bound from the upper bound. Secondly the velocities are then clamped to a percentage of that range.

IV. B-POSITIVE P.S.O (B.P.S.O)

The modifications made in the PSO in order to improve the output values of the benchmark functions mentioned in Table 1 are as follows;

- While initializing the B.P.S.O the velocity (which is actually the displacement of the particles) is clamped to be from zero to a specific percentage (30%) of the total search space. The clamping effect is already a part of Standard PSO but in this case the whole negative part of the velocity is removed which makes negative displacement impossible. Figure 1 shows the overall displacement of particles in a two dimensional search space. It can be noticed that the particles only move in positive direction as the lower bound of velocity (displacement) vector is kept 0.

- In Standard PSO, every time a new velocity is calculated using (3), it is simply added to the position of a particle using (2) to get the new position. In the B.P.S.O the velocity is multiplied with R, where R is a uniform random number generator which generates random numbers between (0, 1).

$$\vec{x}_i(j+1) = \vec{x}_i(j) + R(0,1)\vec{v}_i(j+1) \quad (4)$$

- In Standard PSO, if the new position of a particle is less than the particles optimum position, the new position becomes the optimum one otherwise nothing is done. In the B.P.S.O, instead of doing nothing in the second case, the particle is thrown randomly around its best available position for future evaluations.

$$\text{if} \quad f(\vec{x}_i(j)) > f(\vec{p}(j)) \quad \text{then} \quad \vec{x}_i(j) = R(0,1)\,\vec{p}_i(j) \quad (5)$$

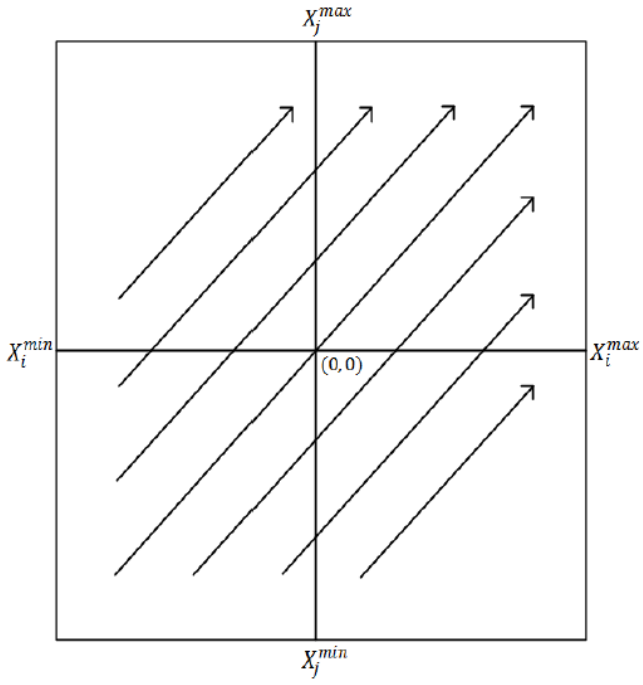

Figure 1. Overall movement of particles in two dimensional search space

V. BENCH MARK FUNCTIONS

There are different benchmark-functions used to evaluate the performance of PSO [5]. Table 1 mentions the five benchmark functions, their equations, bounds and global optimum values. In this paper these five benchmark functions are used to evaluate the performance of Standard PSO as well as the B.P.S.O.

Table 1. Benchmark functions

	Function Name	Equation	Bounds	Global Optimum
BF1	Sphere	$f_1 = \sum_{i=1}^{D} x_i^2$	$(-100,100)^D$	$f_1(\vec{x}) = 0$ $for\ \vec{x} = (0,0,\dots,0)$
BF2	Ackley	$f_2 = -20\exp\left\{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right\} - \exp\left\{\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right\} + 20 + e$	$(-32,32)^D$	$f_2(\vec{x}) = 0$ $for\ \vec{x} = (0,0,\dots,0)$
BF3	Rosenbrock	$f_3 = \sum_{i=1}^{D-1}\{100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\}$	$(-30,30)^D$	$f_3(\vec{x}) = 0$ $for\ \vec{x} = (1,1,\dots,1)$
BF4	Rastrigin	$f_4 = \sum_{i=1}^{D}\{x_i^2 - 10\cos(2\pi x_i) + 10\}$	$(-5.12,5.12)^D$	$f_4(\vec{x}) = 0$ $for\ \vec{x} = (0,0,\dots,0)$
BF5	Schwefel's Problem 2.26	$f_5 = -\sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	$(-500,500)^D$	$f_5^{2D}(\vec{x}) = -837.97$ $f_5^{30D}(\vec{x}) = -12569.5$ $for\ \vec{x} = (420.98,420.98,\dots,420.98)$

VI. SIMULATION RESULTS

The simulation for standard PSO as well as B.P.S.O was conducted using MATLAB on an Intel core i5 (2nd Generation) processor.

A. Selection of Inertia Weight, Acceleration Constants and Maximum Velocity

Different values of inertia weight, accelerations constants and maximum velocity were explored. The output results for all five benchmark functions mentioned in table 1 are listed below in table 2, 3, and 4 by using different values of inertia weight, accelerations constants and maximum velocity.

Table 2 shows the mean and standard deviation of global best values for *c1=c2=1.1931471*, *w=0.721347* [6] and Vmax being 15%, 20% and 50% of total range of search space.

Table 3 shows the mean and standard deviation of global best values for *c1=c2=1.49445*, *w=0.72984* [7] and Vmax being 15%, 20% and 50% of total range of search space.

Table 4 shows the mean and standard deviation of global best values for *c1=c2=2*, w is linearly decreasing from 0.9 to 0.4 [8] and Vmax being 15%, 20% and 50% of total range of search space.

Table 2. Mean global best values and std. dev. for c1=c2=1.1931471 and W=0.721347 for D=30.

Benchmark Functions	Vmax = 15% of range		Vmax = 20% of range		Vmax = 50% of range = UB	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
BF1: Sphere	1.34e-102	4.24e-102	1.35e-99	3.65e-99	1.93e-108	5.86e-108
BF2: Ackley	4.06	1.49	4.41	1.63	5.21	2.21
BF3: Rastrigin	61.58	14.93	75.31	18.9	93.43	35.90
BF4: Rosenbrock	3.67	2.81	2.35	3.78	14.85	27.99
BF5: Schwefel's 2.26	-7343.93	597.41	-8095.88	1193.19	-17269.14	368

Table 3. Mean global best values and std. dev. for c1=c2=1.49445 and W=0.72984 for D=30.

Benchmark Functions	Vmax = 15% of range		Vmax = 20% of range		Vmax = 50% of range = UB	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
BF1: Sphere	5.01e-87	1.51e-86	8.46e-89	1.14e-88	9.59e-088	1.83e-087
BF2: Ackley	0.81	0.92	0.81	0.88	2.38	0.95
BF3: Rastrigin	59.10	15.36	72.63	15.29	61.88	11.45
BF4: Rosenbrock	6.79	6.55	5.31	6.95	14.96	20.19
BF5: Schwefel's 2.26	-7314.43	675.05	-8228.43	833.86	-21423.16	3407.75

Table 4. Mean global best values and std. dev. for c1=c2=2 and W is linearly decreasing from 0.9 to 0.4 for D=30.

Benchmark Functions	Vmax = 15% of range		Vmax = 20% of range		Vmax = 50% of range = UB	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
BF1: Sphere	4.43e-28	1.28e-27	1.08e-29	2.97e-29	5.44e-28	1.26e-27
BF2: Ackley	1.5e-14	5.86e-15	1.47e-14	4.71e-15	1.75e-14	6.65e15
BF3: Rastrigin	31.74	10.14	29.35	7.31	31.93	9.33
BF4: Rosenbrock	31.65	23.55	26.12	18.21	57.08	40.57
BF5: Schwefel's 2.26	-9638.07	828.04	-12137.14	888.77	-35174.96	3041.82

It can clearly be noticed that by comparing Table 2, Table 3 and Table 4, the best results are for Table 4 for *Vmax = 20% of range*. So, after exploring different values for inertia weight, acceleration constants, and maximum velocity, the best ones were chosen as c1 =c2 =2, w linearly decreasing from 0.9 to 0.4 and *Vmax* being 20% of total range of space. Table 5 shows the expected minimum, mean, standard deviation and mean time per replica (in seconds) for these chosen values.

B. The Effects of Acceleration Constants and Inertia Weight

The individual effects of acceleration constants c1, c2 and inertia weight w are as follows:

- The acceleration constant c1 present in the cognition part as shown in (3) affects the influence of personal experience of the particle. Increasing c1 encourages a particle to follow its own instincts [9]. In other words, over iterations the particles is drawn towards its personal best position.

- The acceleration constant c2 present in the social part of (3) affects the influence of social experience of the particle. In other words for a greater value of c2, the particle is drawn towards the swarm's best position [9]. In practice, the weight of cognition and social part is kept at balance by keeping both the accelerations constant equal to each other (c1=c2=2)[1][9].

- The value of inertia weight w is kept less than one to keep the effect of past personal best lesser than the recent personal best on the particle's velocity [10]. In other words, inertia weight makes a particle to forget its bad past experiences. A time varying inertia weight speeds up this process which helps the algorithm to delay the premature convergence to later iterations for better results [9].

C. B-positive Particel Swarm Optimization (B.P.S.O)

The mean values of function evaluations and their respective standard deviation (under the PSO-DD column) in Table 5 are taken from an article on PSO-DD [11]. The results are for *100,000* function evaluations. Table 5 shows a comparison of the expected minimum, mean and standard deviation for global values of Standard PSO, PSO-DD [11], and B.P.S.O. The mean time per replica (in seconds) of Standard PSO and B.P.S.O are also compared.

Table 5. Comparison of standard PSO, PSO-DD [11], and B.P.S.O. the results below 10^{-16} are considered as 0.

			Standard PSO			PSO-DD [11]		B.P.S.O		
Benchmark Functions	Dimensions	Expected Minimum	Mean	Std. Dev.	Time /Replica	Mean	Std. Dev.	Mean	Std. Dev.	Time /Replica
BF1: Sphere	D=2	0	5.77e-28	9.65e-28	0.225	-	-	0	0	0.15
	D=30	0	1.08e-29	2.97e-29	3.94	3.11e-58	1.33e-57	0	0	3.65
BF2: Ackley	D=2	0	2.43e-14	3.23e-14	0.227	-	-	0	0	0.12
	D=30	0	1.47e-14	4.71e-15	4.3	1.02e-01	0.3544	0	0	2.7
BF3: Rastrigin	D=2	0	0	0	0.22	-	-	0	0	0.12
	D=30	0	29.35	7.31	4.2	40.6283	12.7305	0	0	2.6
BF4: Rosenbrock	D=2	0	2.21e-8	4.72e-8	0.22	-	-	2.04e-4	8.12e-5	0.11
	D=30	0	26.12	18.21	4.05	29.8205	27.3051	25.7	0.26	2.5
BF5: Schwefel	D=2	-837.97	-1189 .33	207.40	0.024	-	-	-1281.58	251.9	0.12
	D=30	-12569.5	-12137.14	888.77	4.65	-10647.5	349.56	-12568.5	0.87	3.03

VI. CONCLUSION

This paper shows the implementation of Standard Particle Swarm Optimization and a new technique for Particle Swarm Optimization (B.P.S.O). After justifying the values of inertia weight *(w)* and acceleration constants *(c1, c2)*, Standard P.S.O and B.P.S.O are tested for five well known Benchmark Functions available in Table 1. A comparison is done between Standard PSO, PSO-DD and B.P.S.O results. It is observed that the convergence to global optimum of B.P.S.O is comparatively efficient. The Global best position results also show that compared to standard PSO, the B.P.S.O approach closer to global optimum position. As a whole, the B.P.S.O seems to give good results for all five benchmark functions.

ACKNOWLEDGEMENT

This research is supported by the Electrical Engineering Department of King Saud University and is a result of the motivation and encouragement of Dr. Adel Abdennour, who is a professor in King Saud University.

References

[1] J. Kennedy and R. C. Eberhart, Particle swarm optimization, in Proceedings of the *IEEE International Conference on Neural Networks, Perth, Australia*, (1995), 1942-1948.

[2] Y. Shi, Particle swarm optimization, *IEEE Neural Network Society*, Feature Article, 8–13 (2004).

[3] J. Kennedy, The particle swarm: Social adaptation of knowledge, *Proc. IEEE Int. Conf. Evolutionary Computation*, 303-308 (1997).

[4] P. Engelbrecht, *Computational Intelligence: An Introduction*, 2 ed., (2007): John Wiley and Sons.

[5] X. Yao, Y. Liu, and G. Lin. , Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, **3(**2), 82–102(1999).

[6] M. Clerc, *Standard Particle Optimisation From 2006 to 2011 version 13-July-2011*.

[7] R. Eberhart and Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in *Proc. Congress on Evolutionary Computation, La Jolla, CA*, 2000, 84-88.

[8] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, *Proc. 1999. Congress on Evolutionary Computation*, (1999), 1942-1948.

[9] Y. Shi and R. Eberhart, A modified particle swarm optimizer, in Evolutionary Computation Proceedings, *IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, Anchorage, AK*, (1998), 69-73.

[10] Y. Shi, and R. Eberhart, Parameter selection in particle swarm optimization, Proceedings of *7th Annual Conference on Evolution Computation*, (1998), 591-601.

[11] C. Worasucheep, A Particle Swarm Optimization with stagnation detection and dispersion. *Evolutionary Computation*. CEC 2008. *(IEEE World Congress on Computational Intelligence)*. IEEE Congress on Digital Object Identifier, 424 – 429.