International Journal of Computer Applications (0975 – 8887)
Volume 121 – No.11, July 2015

Dealing with Boundary Constraint Violations in Particle Swarm Optimization with Aging Leader and Challengers (ALC-PSO)

Avneet Kaur
Student, Computer Science Department,
Guru Nanak Dev University,
Regional Campus, Jalandhar

Mandeep Kaur
Assistant Professor, Electronics Department,
Guru Nanak Dev University,
Regional Campus, Jalandhar

ABSTRACT

Boundary violation is a common process in optimization problems. This problem can be seen in Particle Swarm Optimization(PSO) and its variants too. An aging based variant of PSO called, PSO with Aging Leader and Challengers(ALC-PSO) overcomes the stagnation problem that existed in PSO. To avoid the problem of random particles, some bound handling mechanisms need to be applied to Particle Swarm Optimization with Aging Leader and Challengers (ALC-PSO) to improve its performance. During the search process, some particles may leave the search boundaries within which the optimal solution is to be found. It becomes essential to handle such boundary constraint violations and some boundary handling strategies are required to be implemented. This paper presents some of these bound handling methods applied to ALC-PSO algorithm and comparisons are made with PSO. These methods include velocity initialization, velocity clamping and bound handling methods. The results are simulated on MATLAB R2011b for Ackley benchmark problem.

Keywords

Search Space, Particle, Velocity, Boundaries, Random Particles.

1. INTRODUCTION

The boundary constraint violation handling methods include 1.) Velocity initialization approaches that mean to re-initialize the velocity of particles whose velocity exceeds the pre-defined maximum velocity v_{max}. There are three velocity initialization strategies, that are initializing to zero, initializing to value within domain, initializing to small random value near zero 2.)velocity clamping, the velocity of particles in a swarm is clamped to a fixed value so that if a particle's velocity exceeds that value, it is set back to the clamped value, which can be maximum or minimum 3.) Bound Handling techniques include unmodified method i.e. do not alter velocity of a particle leaving the boundaries, deterministic back i.e. bringing a variation in velocity update rule, nearest value i.e. initializing the particle to peak values of velocity and position. To deal with the boundary constraint violations in ALC-PSO, these mechanisms are applied.

1.1 Basic Particle Swarm Optimization (PSO)

Particle swarm optimization is a heuristic global optimization method[1]. Particle Swarm Optimization (PSO) was originally proposed to simulate the swarming behavior of bird flocks as they wander from one place to other in search of food. Swarm intelligence is a kind of multi-agent system where agents follow some simple rules and interact with each other so that an interactive behaviour between them emerges. PSO is one of the types of swarm intelligence and every particle in the swarm flies within the search space by updating its individual velocity after every iteration of process towards: i.) *the personal best position ii.) global best position*

The fitness function of each particle is evaluated iteratively in order to determine the particle that offers the lowest function value for the function. The personal best, neighborhood best and global best are stored to a memory location that all particles can access and this location can be utilized to determine the particle's individual velocities [2].

2. PARTICLE SWARM OPTIMIZATION WITH AGING LEADER AND CHALLENGERS ALGORITHM

The aging mechanism is applied to the stochastic based Particle Swarm Optimization (PSO), so as to take away the limits that existed in PSO such as for example: it gets caught in local optima and the algorithm converges pre-maturely. When aging leader algorithm is applied to PSO, these limitations are removed in an efficient manner. Aging is an inevitable process[3] that involves all and spares none. This mechanism of 'aging' is used in the Particle Swarm Optimization Algorithm, to find an optimal solution for an optimization problem. An optimization problem can be extremely difficult[4] to solve manually, there is a need to solve such problem using an optimization algorithm. PSO is a very simple and efficient method for solving such problems. When Aging mechanism is applied to the PSO, the problem of premature convergence that existed in PSO is overcome and the efficiency of the algorithm is highly increased.The designing of ALC-PSO can be done in three steps:

1. Design lifespan controller- adjusting the lifespan of the leader according to its leading power is done. If the leader is efficient to lead the swarm, its lifespan is increased but if the leader is not efficient, its lifespan is decreased. The lifespan is adjusted adaptively on the basis of leading power.

2. Generation of Challenger- A challenger is generated to challenge the leader of swarm who becomes inefficient.

A challenger is generated using the following pseudocode:

```
1. count= 0
2. For j=1 to n
3. If random(0,1)< pro
```

```
4.  Challenger$^j$ = random($L^j$, $U^j$)
5.  Count=count+1
6.  Else
7.  Challenger $^j$=Leader$^j$
8.  End if-else
9.  End for
10. If count=0// make sure that challenger is different
    from the leader
    randomly select a dimension ran
11. challenger$^{ran}$=random($L^{ran}$, $U^{ran}$)
12. end if
13. return
14. end procedure [5]
```

For every dimension j(j=1,2,… n) , a random number rnd_j is uniformly distributed within (0,1) is generated and compared with a parameter pro € (0,1). If rnd_j < pro, challenger j is set to a randomly generated number that is uniformally distributed in range [L^j, U^j], where L^j and U^j are lower and upper bounds of dimension j, else Challenger ^{j} is inherited from the previous leader.

3. **Accepting a challenger**- The acceptance or rejection of challenger is done by comparing the leading power of challenger with leader.

The challenger can be accepted or rejected using the following pseudocode:

```
1.  For i=1 to m                /*record current status*/
2.  Old $X_i$= $X_i$ ; Old $V_i$=$V_i$
3.  End for
4.  For iteration= 1 to T    /*test leading power of
    leader for T iterations*/
5.  For i=1 to M
6.  For i=1 to n             /* update velocity and
    position */
7.  $v_i^j = w.* v_i^j + c_1.r_1^j. (pBest_i^j - x_i^j) + c_2. r_2^j. (Leader^j - x_i^j)$
8.  $x_i^j = x_i^j + v_i^j$
9.  end for
10. Evaluate f($X_i$)
11. Update $pbest_i$ and challenger
12. End for
13. If atleast one pbest position is improved
14. Leader=challenger   /* use the challenger as the new
    leader */
15. b=0                         /* age is initialized to 0 */
16. t=$t_0$  lifespan is updated to the initial value
17. return
18. end if
19. end for
/*challenger is unacceptable and old positions are resumed */
20. For i=1 to M
21. $X_i$= old$X_i$ ; $V_i$=old$V_i$
22. End for
23. b=t-1;
24. return
25. end procedure
```

The leading power of newly generated challenger is evaluated, if this challenger has enough leading power, it replaces the old leader and itself becomes the new leader.

Age is initialized to 0 i.e. b=0

Lifespan t is initialized to t_0. else the old leader remains unchanged and will continue leading the swarm[5].

When this aging mechanism is applied to the PSO algorithm, the algorithm restricts some particles from becoming the leader of the swarm and thus, does not let those weaker particles to enter the process of finding the best solution. The challengers are generated by comparing their leading power[9]. If the leading power of the challenger is better, its lifespan is increased and it becomes the leader of the swarm. Thus, the stagnation problem is removed.

3. DEALING WITH BOUNDARY CONSTRAINT VIOLATIONS

A problem that arises in PSO and ALC-PSO as well, is the random behavior of particles[6], due to which the particles leave the search boundaries within which the solution is usually to be found. If the particles go out of the desired boundaries, there is a probability of best particle gone out of the search boundaries, so it becomes necessary to bring the particles back within the boundaries, so that no effort is wasted for finding the solution which is no longer available in the set of feasible solutions.

To overcome the situation of randomness of particles, boundary constraints are applied to the velocity of the particles of swarm. Every time a particle goes out from the defined boundaries, it is brought back within the search space, so that no effort is wasted in finding that particle, which is no longer available in the search space. When the particle is initialized, it is initialized to zero velocity, i.e. particles are stationary. When randomness within swarm increases, particles fly with different velocities and hence change their positions. Thus, particles move within swarm to find the best position. But as particles go out of the search space, they are required to be brought back into the search space, which can be done by re-initializing the velocities of such particles.

There three ways of dealing with boundary constraint violations-

1. Velocity re-initialization
2. Velocity clamping
3. Bound handling methods

3.1 Velocity re-initialization Strategies

The velocity of every particle in the swarm is updated using the following equation:

$$v_i(t+1) = w*v_i(t)+c_1r_1(t)(p(t)- x_i(t))+c_2r_2(t)(g(t) -x_i(t))$$

where w is the inertia weight [7],

c_1 and c_2 are the acceleration coefficients,
$r_1(t)$, $r_2(t)$ € $U(0, 1)^{nx}$, nx is the dimension of the search space,
$x_i(t)$ is the current position of the ith particle,
p (t) is the particle's pbest position,
g (t) is the gbest position.

Particle positions are updated using the following equation:

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

During each iteration of the algorithm, the velocity and position of the particles get updated by following the velocity and position update rules. During this process, a particle may leave the boundary positions, they need to be re-initialized. This re-initialization of particles can be done in three ways, in order to save the searching effort and searching time that has to be made in finding the solutions for the algorithm.

3.1.1 Initialize velocities to zero

For all the particles in the swarm(i=1,2,... nx) where nx is the total number of particles. This initialization limits the initial exploration ability of the swarm and the surface occupied initially by the particles of the swarm. The momentum of every particle is zero initially i.e. they do not move in starting, they are initialized to zero velocity. If the momentum value is greater than zero, it will lead to larger step size , which will further increase the randomness of the particles. The positions of the particles also are uniformly distributed throughout the swarm.

3.1.2 Initialize of velocities to the random values within the domain

The domain is set for the optimization problem i.e. Vi(0) € U(-x min, x max)^{nx} , nx being the dimension of the search space,. Because of the larger step sizes, the random initialization of the velocities help in the improvement of exploration ability of the swarm. These larger step sizes also increase the initial diversity of swarm and it may result in particles violating the boundary constraints on the search space. As the initial positions of the particles are initialized in the domain [x min, x max], similarly, the initial velocities of the particles are initialized in a fixed domain. In this way, these roaming particle's best positions may also violate the boundary constraint.

3.1.3 Initialize the velocities to small random values

In response to the problem of particles leaving the boundary values due to large step sizes, the velocities of particles were initialized to small values. It was supposed that this kind of initialization will not suffer from the problem of boundary violation and will contribute to the diversity of the swarm. But it was noticed that particles still leave the boundaries. Another problem that arose is how the small values will be found, which depends on the characterization of the optimization problems.

3.2 Velocity Restriction/ Velocity Clamping

To limit the range of velocity for every particle, the swarm is clamped with velocity i.e. a restriction is imposed on the velocity of particles. The concept of velocity clamping is simple and it follows the rule that boundary values of velocity are assigned to the particles of a swarm, such that if any particle goes out of the desired range, it is set to the clamped value [9].

1. **Maximum velocity**- if particle's velocity increases the maximum allowed velocity, it is set to this maximum value.
2. **Minimum velocity**- if particle's velocity decreases the minimum allowed velocity, it is set to this minimum value. Example- Vmax=10 , Vmin= -10. *[10]*

After every iteration, velocity is updated, if it increases vmax or decreases vmin, it is set to boundary values. Usually, $v_{max,\ j}$ along jth dimension is taken to be 0.1 to 1.0 times of maximum value of x along that dimension.

If a condition arises, where the current position of the particle becomes same as the personal best position and global best position, the particle may not change its position and with this, all other particles will follow this position. As a result, all the particles of the swarm will get accumulated at that point , so the swarm converges at that point only , leading to premature convergence.

During the process of finding solutions, if the particle's current position is far away from the personal best (pbest) and global best positions(gbest), particle's velocity may exceed its limits. So, it becomes essential to restrict the values of velocity for every particle of the swarm, such that the particles don't get out of the boundaries of search space If particles go out of the search space, it will affect the position of particles; as it will result in larger position updates due to large velocity values. This will further lead the particles leaving its boundaries. So, it is essential for the particle's velocity to be clamped to certain limits, such that if these particles increase their velocities, these are put back to the limits.

The exploration-exploitation balance is done to measure the accuracy and efficiency of optimization algorithms. Exploration capability means the ability of algorithm to explore into various directions of the search space to find the global optima whereas exploitation capability means the algorithm focuses on a specific region in order to find out the candidate solution. The balanced exploration-exploitation ratio accounts for better optimization algorithms. In Particle Swarm Optimization algorithms, the particles that are far away from pbest, its velocities exceeds to larger values, so they lead to larger position update rules, further leading to particles leaving the boundaries of search space. In order to overcome the problem of velocity exceeding the intended values, the velocity of particles of swarm need to be clamped, so that globl exploration be controlled. Velocity Clamping is used to accelerate the algorithm's convergence speed and to avoid premature convergence of algorithm. If the velocity *v* of some particle *i* exceeds a maximum allowed velocity limit, it is set to the maximum value of velocity (*vmax,j*), which is the maximum allowed value of velocity. Velocity is updated as:

$vij\ (t+1) = v_{ij}(t+1),\ if\ v_{ij}(t+1)<v_{max,\ j}\ else\ v_{ij}(t+1)= v_{\max,j}$

Larger values of *vmax,j* cause global exploration and smaller values encourage local exploitation. It is not easy to choose the appropriate value of Vmax for solving a particular optimization problem being in consideration. Poor vmax chosen can lead to extremely poor performance. Moreover there is no simple and reliable method to choose the vmax. Mostly trial and error method is used[8].

Applying velocity clamping to particles of swarm is a must-do for having efficient results.

1. It does not affect the position of particles.
2. It decreases the step sizes.
3. Changes the particle's search direction to have better exploration

3.3 Bound Handling

Many bound handling approaches have been proposed[11] [12]. The particles in the swarm are restricted to leave the bounds of the search space. If particles leave the boundaries, some techniques are followed to bring them back into search area. Bound Handling mechanisms include:

1. Creation of feasible-only solutions during evolutionary search.
2. Explicit mechanism to repair an infeasible solution i.e. bringing the infeasible solution back into the feasible search space.

The bound handling mechanisms are divided into two groups-

1. Group A- These mechanisms carry out feasibility search variable wise.
2. Group B- These mechanisms carry out feasibility search vertically.

The boundary constraints can be applied by following –

A. **Change pbest/gbest update**- such a particle is not selected.

B. **Update velocity**- Velocity can be updated in following three ways:

1. Unmodified- The velocity of such a particle is not altered i.e. the particle's velocity remains the same.
2. Deterministic back- The velocity is set by multiplying the negate of a predefined constant as- $v_{i,t+1=}$ -k $v_{i,t+1}$
3. Nearest value- The velocity of the particle is set to the peak value assigned for the process[13].

There are position bound handling methods [14] but these will not be combined as it may make the process complex.

4. SIMULATION RESULTS

The PSO and ALC-PSO algorithms are implemented and results are simulated on MATLAB R2011b for the Ackley benchmark problem and error i.e. difference between results fund by actual optimal problem and those found by PSO and ALC-PSO algorithms are calculated with respect to every iteration of the algorithm. Every strategy is applied for 50 particles and 20 iterations of process.

4.1. Velocity Initialization

If any particle leaves the boundaries of the search space or its velocity increases the pre-defined value of velocity, it needs to be re-initialized so that it is brought back into the search space, as the initial condition of algorithm and fresh procedure be started for that particle, so that no effort is wasted in finding the particle that has left the boundaries of search space

4.1.1 Velocity initialization to 0

This initialization can be done by bringing the particle back to initial momentum, as when it was initialized at the starting of algorithm. The velocity of the particle is set to zero, if its velocity increases the maximum velocity (peak velocity), clamped for a process, or its velocity decreases the minimum velocity.

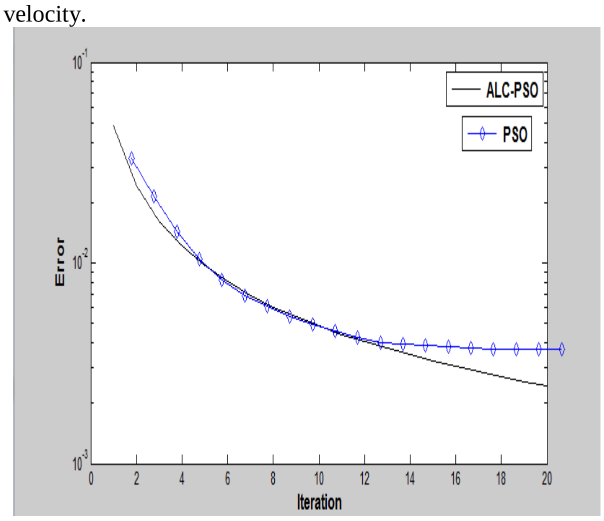

Figure 1: Graph Plots between error and iteration for PSO and ALC-PSO for Ackley Function using Velocity initialization to zero strategy

Discussion of Results- The error value for PSO at iteration 2 is located at $10^{-1.5}$, whereas for ALC-PSO it is closer to $10^{-1.8}$ that means the error is more at starting for ALC-PSO but as iterations are run, there is a constantly decreasing error for both curves. When iteration 12 is reached, the curve for ALC-PSO starts falling compared to PSO, resulting in lesser error value than PSO. Finally, at iteration 20, the ALC-PSO curve declines as compared to PSO at value near to $10^{-2.5}$ whereas it is near to 10^{-2} for PSO.Better performance of ALC-PSO for Ackley function giving gbest value 0.04880 compared to 0.0880 given by PSO.

4.1.2 Velocity Initialization to value within domain

The velocity initialization to a value within domain strategy initializes the velocity of the particle that has gone out of the search space to some random value within a pre-defined range. The range taken here is [0,1] within which the velocities are initialized if any particle violates the boundary constraints for the velocity and position of the search space..

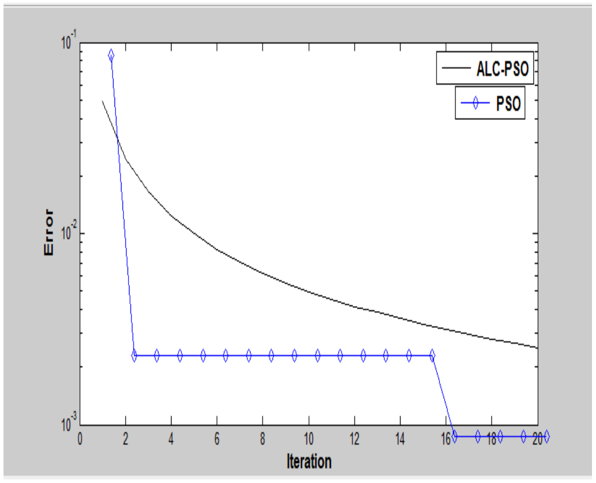

Figure 2: Graph Plots between error and iteration for PSO and ALC-PSO for Ackley Function using Velocity initialization to value within domain

Discussion of Results- The error plots for PSO at iteration 1 nearly touches the value 10^{-1} whereas the value for ALC-PSO at iteration 1 is lesser than 10^{-1} i.e. for PSO . The PSO error curve shows a sudden fall at iteration 2 to value nearly $10^{-2.5}$ which then continues to be stable till iteration 15 which then falls to value 10^{-3} after iteration 15 to iteration 20. Whereas for ALC-PSO, the error curve shows a constant decrease in error rate till last iteration 2 . The error rate for PSO is smaller than for ALC-PSO using Ackley function for the velocity initialization to values between domain. The gbest value for PSO is found to be 0.0196 and for ALC-PSO , it came out to be 0.0505 that means the PSO performs better than ALC-PSO for Ackley function while using velocity initialization to value within domain.

4.1.3 Velocity initialization to random value near zero

The velocity of the particle is initialized to some random value near to 0, so that the particles can be brought back into the search space for finding the solution of the problem. Here, the values take are: 0.004 and 0.002.

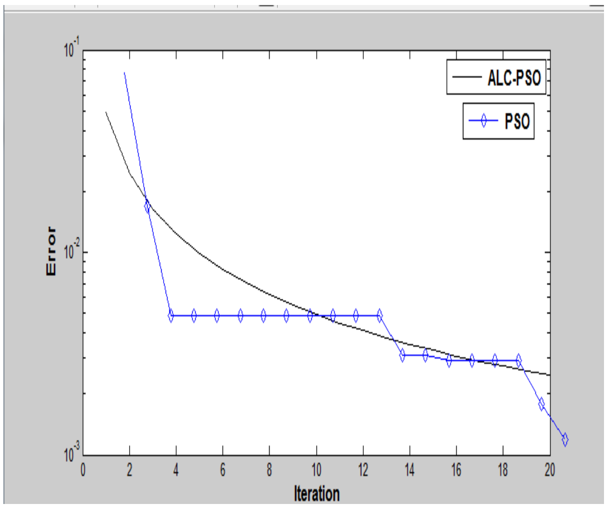

Figure 3: Graph Plots between error and iteration for PSO and ALC-PSO for Ackley Function using Velocity initialization to small random value near zero

Discussion of Results- The error curves for PSO shows sudden steep at iteration 2 to iteration 4 from error value near to 10^{-1} to error value $10^{-2.5}$. The ALC-PSO shows a constant decrease of error value from nearly $10^{-0.8}$ to $10^{-0.5}$ at iteration 11. The error rate for ALC-PSO comes to a value of $10^{-0.7}$ during last 20th iteration while the error rate for PSO comes to value nearly 10^{-3} at the last 20th iteration. The error values for the ALC-PSO are more in the starting iterations but as iterations are run, the values decrease while the error rate for PSO which was initially higher than ALC-PSO decreases than ALC-PSO at iteration 3 and continues to be lower than ALC-PSO till iteration 11 which then rises slightly and again falls close to ALC-PSO curve. The gbest values for ALC-PSO and PSO are closer to each other as for ALC-PSO, the value is 0.0493 while for PSO the value was little higher 0.642.

Table 1. gbest values found by PSO and ALC-PSO for velocity initialization strategies

Strategy	Gbset for PSO	Gbest for ALC-PSO
Velocity initialization to zero	0.0880	0.0488
Velocity initialization to value within domain	0.0196	0.0505
Velocity initialization to small random value near zero	0.642	0.0493

4.2 Velocity clamping in PSO

Velocity clamping is essential to restrict the particles from going out of boundaries of search space.

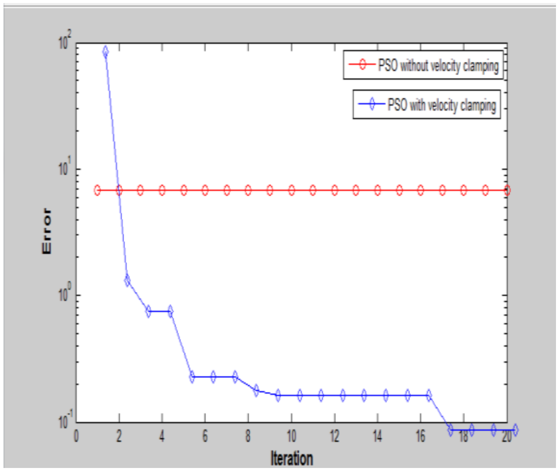

Figure 4: Graph Plots between error and iteration for PSO with and without velocity clamping for Ackley Function with 50 particles and 20 iterations

Discussion of Results- The error plots for the PSO with velocity clamping and PSO without velocity clamping are shown. The error value for PSO without velocity clamping remains constant at $10^{0.8}$ while error values for PSO with velocity clamping varies from 10^{2} at iteration 2 to value 10^{-1} at 20th iteration. The results given by PSO with velocity clamping are better than PSO without velocity clamping having gbest values 0.0550 and 0.9273 respectively.

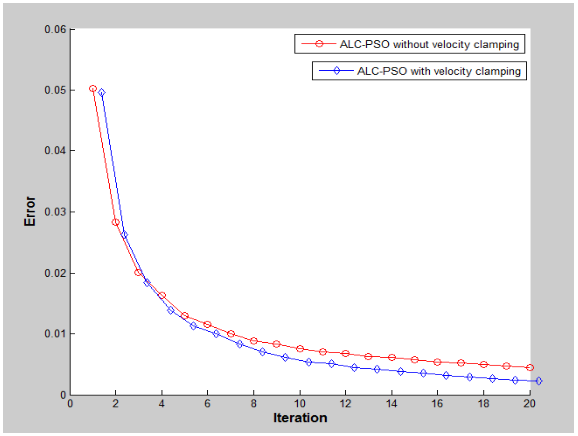

Figure 5: Graph Plots between error and iteration for ALC-PSO with and without velocity clamping for Ackley Function with 50 particles and 20 iterations

Discussion of Results- The ALC-PSO with velocity clamping shows better performance than ALC-PSO without velocity clamping. The error curve for ALC-PSO with velocity clamping remains lower than the ALC-PSO without velocity clamping. The error values range from 0.005 to nearly 0 for Ackley function. The gbest value given by ALC-PSO with velocity clamping is better i.e. lesser than ALC-PSO without velocity clamping. Gbest values are 0.0850 and 0.0894 respectively for the Ackley function.

Table 2. gbest values found by PSO and ALC-PSO with and without velocity clamping

PSO		ALC-PSO	
Without velocity clamping	with velocity clamping	Without velocity clamping	With velocity clamping
0.9273	0.0550	0.0894	0.0850

4.3 Bound Handling

Three bound handling techniques are implemented.

1. **Unmodified** – In the unmodified bound handling approach, the velocity of the particle leaving boundaries of search space is unaltered.

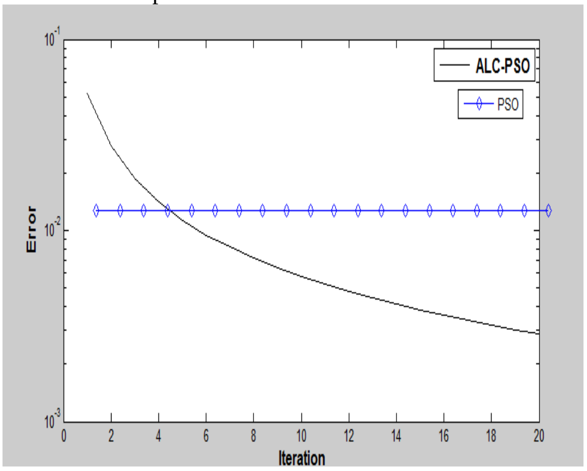

Figure 7: Graph Plots between error and iteration for PSO and ALC-PSO for Ackley Function with 50 particles and 20 iterations using Unmodified bound handling

Discussion of Results- The PSO error curve shows a constant error reading of $10^{-1.9}$ for every iteration of the process while ALC-PSO shows an exponential decrease in the value of error form $10^{-1.3}$ at iteration 1 to $10^{-2.5}$ at iteration 20, remaining lower than PSO curve for maximum number of iterations. The gbest value given by the ALC-PSO is better than PSO for the unmodified bound handling mechanism applied to the Ackley benchmark problem. The gbest value given by PSO is 0.4143 which is higher than 0.0574 gbest value for ALC-PSO.

2. **Deterministic back**

A constant negative term k is multiplied with the velocity of the particle so as to bring the particle back into the bounds of the search space.

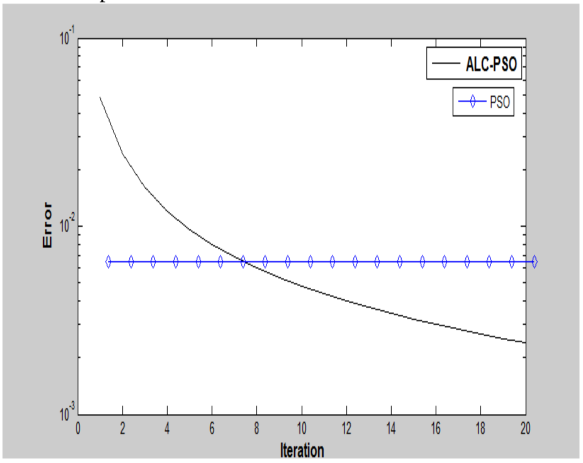

Figure 8: Graph Plots between error and iteration for PSO and ALC-PSO for Ackley Function with 50 particles and 20 iterations using Deterministic back bound handling method

Discussion of Results- The PSO error curve shows a constant error value at $10^{-2.3}$ nearly for all the iterations of the process while ALC-PSO error curve shows a constant exponential decrease in the value of error from value $10^{-1.4}$ at iteration 1 to value of $10^{-2.8}$ nearly at the last 20th iteration.The gbest value for the ALC-PSO is 0.0481 while for PSO is 0.1601 for the Ackley function using deterministic back bound handling strategy.

3. **Nearest value-** If any particle crosses the search boundaries, it is brought back into the search bounds. If the position value at any interval of time, exceeds the peak value for position, it is set back to the boundary value i.e. peak value so that it reaches back into the intended search space. In simple word, Nearest value means Initialize the velocity to peak value for increase in position peak value defined for every particle.

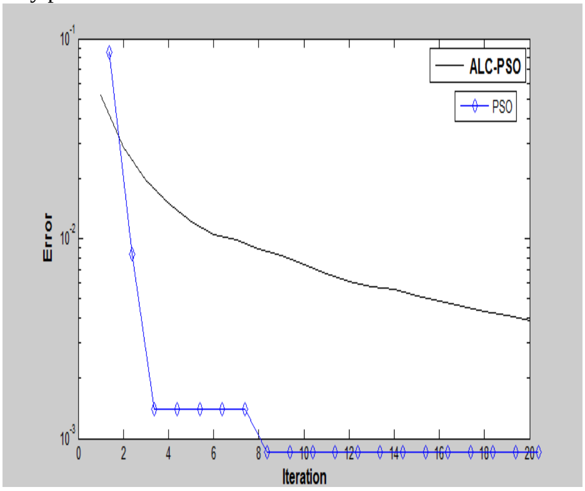

Figure 9: Graph Plots between error and iteration for PSO and ALC-PSO for Ackley Function with 50 particles and 20 iterations using Nearest Value bound handling method

Discussion of Results- The graph plots between error and iteration for Ackley function show a clear distinction between the PSO and ALC-PSO error curves. The nearest value method does not perform well for the Ackley function. The ALC-PSO shows more error than PSO, hence less efficient for this method of bound handling. The error value for PSO at iteration 3 is 10^{-1}, that falls to nearly 10^{-3} immediately and remains constant till 7th iteration, then becomes constant at $^{10-3}$ for all the remaining iterations. The Ackley function gives gbest value 0.0263 for the PSO algorithm and 0.0779 for the ALC-PSO algorithm when the nearest value strategy is used for handling the boundaries of search space.

Table 3. gbest value found by PSO and ALC-PSO for bound handling techniques

Strategy	PSO	ALC
Unmodified	0.4143	0.0574
Deterministic back	0.1601	0.0481
Nearest Value	0.0263	0.0779

5. CONCLUSION

Dealing with boundary constraint violations has been done successfully, which can be seen through the improved performance of ALC-PSO algorithm. Velocity clamping is an essential concept in the PSO algorithm, and so in ALC-PSO algorithm to restrict the particles from exceeding the maximum pre-defined velocity. Out of the three velocity initialization strategies, initializing to zero and initializing to small random value within domain works well while initializing to value within domain does not work well for ALC-PSO algorithm. All the bound handling techniques prove to be successful when implemented on ALC-PSO algorithm.

6. REFERENCES

[1] Qinghai Bai, "Analysis of Particle Swarm Optimization Algorithm" Volume 3, no.1, February 2010.

[2] A. E. Smith, "Swarm intelligence: from natural to artificial systems [book reviews]," IEEE Transactions on Evolutionary Computation, vol. 4, no. 2, pp. 192–193, 2000.

[3] Avneet Kaur "Particle Swarm Optimization with Aging Leader Algorithm : A Review ", International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 4 Issue 02, February-2015.

[4] Woo Nam Lee and Jong Bae Park, "Educational Simulator for Particle Swarm Optimization and Economic Dispatch Applications ", IEEE Transactions on Power Systems, 03/2005.

[5] Wei-Neng Chen , Jun Zhang, Ni Chen, Zhi-Hui Zhan , Henry Shu-Hung Chung , Yun Li, Yu-Hui Shi "Particle Swarm Optimization with an Aging Leader and Challengers" , IEEE, 2013.

[6] Andries Engelbrecht, "Particle Swarm Optimization: Velocity Initialization", WCCI 2012 IEEE World Congress on Computational Intelligence, June 2012.

[7] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," Proc. IEEE Congr. Evol. Comput, pp. 1945–1950, Jul. 1999.

[8] Daniel Bratton, James Kennedy, "Defining a Standard for Particle Swarm Optimization", Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS), 2007.

[9] Avneet Kaur, "Particle Swarm Optimization with Aging Leader Algorithm : A Review", International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 4 Issue 02, February-2015.

[10] Farrukh Shahzad, A. Rauf Baig, Sohail Masood, Muhammad Kamran,Nawazish Naveed, "Opposition-Based Particle Swarm Optimization with Velocity Clamping (OVCPSO)", Advances in Computational Sciences, Advances in Computational Intelligence , Advances in Intelligent and Soft Computing Volume 116, pp 339-348

[11] Juan C. Fuentes Cabrera and Carlos A. Coello Coello, "Handling Constraints in Particle Swarm Optimization using a Small Population Size", Advances in Artificial Intelligence , Lecture Notes in Computer Science, Volume 4827 , pp 41-51

[12] Jian Li, Bo Ren, and Cheng Wang, "A Random Velocity Boundary Condition for Robust Particle Swarm Optimization" Bio-Inspired Computational Intelligence and Applications, , Lecture Notes in Computer Science, Volume 4688, pp 92-99

[13] Analyzing the Effects of Bound Handling in Particle Swarm Optimization.

[14] Wei Chu, Xiaogang Gao, Soroosh Sorooshian, Handling boundary constraints for particle swarm optimization in high-dimensional search space, Springer, Information Sciences 181 (2011) 4569–4581, October 2010.