Particle Swarm Optimizations: A Critical Review

Atyabi, Adham
Center of Artificial Intelligence and Intelligent Computing, Faculty of IT, Multimedia University, Cyberjaya, Malaysia
adham.atyabi07@mmu.edu.my

PhonAmnuaisuk, Somnu
Center of Artificial Intelligence and Intelligent Computing, Faculty of IT, Multimedia University, Cyberjaya, Malaysia
somnuk.amnuaisuk@mmu.edu.my

Ho Chin, Kuan
Center of Artificial Intelligence and Intelligent Computing, Faculty of IT, Multimedia University, Cyberjaya, Malaysia
ckho@mmu.edu.my

Samadzadegan, Sepide
Computer Science, Faculty of Computer, Islamic Azad University, Tehran, Iran
sepideh.samadzadegan@hotmail.com

Abstract—Particle Swarm Optimization (PSO) method is an Evolutionary algorithm which outperformed other evolutionary algorithms such as GA and inspired from animal's group works and behaviors. Particle Swarm Optimization (PSO) was firstly built to be used in the distributed robotic systems. Although PSO offers many promising prospects, the approach also has some weaknesses and therefore other researchers have tried to introduce optimized versions to improve it. In this paper, we describe Particle Swarm Optimization and discuss eight various optimizations which were published by other researchers to solve its weaknesses. We also compare these modified versions with each other and investigate their strengths and weaknesses.

I. INTRODUCTION

The Particle Swarm Optimization (PSO) is a population–base, self-adaptive search optimization technique that originally introduced by Kennedy and Eberhart in 1995 and inspired from bird flocking, fish schooling and even human social behaviors [1]-[25].

Later PSO has turned out to be a worthy alternative to the standard Genetic Algorithm (GA) and other iterative optimization and evolutionary algorithms. As PSO is a population-based technique, it uses a population of individuals (particles) with a random initialization over the search space and an ability to search for optimal values of a function by updating the population through a number of generations and generating a new population from the old one.

Although PSO has outperformed GA in various problem solving tasks, it also has some weaknesses which have caused others to implement modified versions of it. Angelina in [1] pointed out that the PSO often converges significantly faster to the global optimum than the EA but has difficulties in fine-tuning solutions. Hence, the performance of the PSO flattens out with a loss of diversity in the search space as the overall result [1].

Weaknesses of Basic PSO are as follow [2]-[9]:

1. Parameter control and fitness evaluator function. For examples, various methods for controlling parameter's values were used in PSO such as: Time Varying Acceleration Coefficients (TVAC) [2],[4]-[8], Time Varying Inertia Weigh (TVIW) [6], linear decreasing Weight (LDW) [24], Threshold Model [2], [9] and Random or even Constant/Fix values (RANDIW, FAC) [6], etc. Furthermore different fitness evaluator functions were used such as: standard benchmarking functions (Sphere, Rosenbrock, Rastrigrin, Griewank, Schaffer's f6, etc [3]-[11]) or self structural fitness functions (mostly used in multi objectives and multi-dimensional domains were standard functions were useless).

2. Lack of Dynamic velocity adjustment results the inability to hill-climb solution (e.g. premature convergence and lack of diversity).

3. Inability to work on dynamic search spaces and real-time problems (e.g. PSO were not worthy in domains that certain solutions should be found during a specific time or in dynamic problems were the situation and states would be changed in each period of time).

Since 1995 that PSO method was introduced, many researchers presented optimizations to solve its weaknesses and these methods were used for various scenarios. We categorize them as follows:

Low dimensional scenarios:

1. Particle Swarm Optimization with Spatial Extension (SEPSO) [2], [3].
2. Attraction and Repulsion in Particle Swarm Optimization (ARPSO) [2], [4].
3. Hierarchical particle Swarm Optimizer with time varying acceleration coefficients (HPSO-TVAC) [2], [5], [6], [7].
4. Particle Swarm Optimizer with Mutation and Time Varying Acceleration Coefficients (MPSO-TVAC) [6].
5. Constriction factor Particle Swarm optimization (CPSO) [8].
6. Division of Labor in Particle Swarm optimization (DOLPSO) [2], [9].
7. Gregarious Particle Swarm Optimizer (G-PSO) [7].
8. Fitness Distance Ratio Based Particle Swarm Optimization (FDR-PSO) [10].
Multi objective optimization problems:
2. A Particle swarm Optimization for Reactive power and Voltage Control Considering Voltage Security Assessment (VEPSO) [12].
3. Particle Swarm Optimization with Area Extension [20].
4. Effects of Communication Range, Noise and Help Request Signal on Particle Swarm Optimization with Area Extension (AEPSO) [21].
5. Effectiveness of Cooperative Learning on Particle Swarm Optimization with Area Extension (AEPSO) [22].
Hybrid PSOs:
1. Fuzzy scenarios:
   • An Integrated Fuzzy Control system for Structural Vibration [13].
   • A Particle swarm optimized Fuzzy Neural Network for Voice Controlled Robot Systems [14].
2. Niching techniques [15]:
   • Stretched PSO.
   • Nbest PSO.
   • Niche PSO.
3. others
   • Particle Swarm Optimization algorithm in Signal Detection and Blind Extraction (PSO-MUD, PSO-BES) [16].
   • DEPSO: Hybrid Particle Swarm with Differential Evolution Operator (DEPSO) [17].
   • A Hybrid Particle Swarm and Neural Network Approach for Reactive Power control [18].
   • Swarm Intelligence in Data Mining [23].
   • Swarm Optimization as a New Tool for Data Mining [24].
   • Particle Swarm based Data Mining Algorithms for Classification tasks [25].

In this study, we focused mainly on the PSOs in Low dimensional scenarios and investigate their strengths and weaknesses. The presentation is organized as follows. In section 2 we discuss Basic PSO. In section 3 modified versions of PSO are discussed and the conclusion is presented in section 4.

II. PARTICLE SWARM OPTIMIZATION (Basic PSO)

Particles movement in the original PSO theory, which was introduced by Eberhart and Kennedy, was based on the effect of two elastic forces (e.g. attraction with a random magnitude to the fittest location encountered by the particle and attraction with a random magnitude to the best location observed with any member of swarm)[1]-[11]. In PSO, individuals are called particles, the population is called swarm and in various PSOs, each bit is considered as a dimension. The original PSO equation was as follow:

\[ a_i = \Psi_1 R_1 (x_i - x_p) + \Psi_2 R_2 (x_i - x_g) \]  \hspace{1cm} (1)
\[ v_i(t) = v_i(t-1) + a_i \] \hspace{1cm} (2)

Where \( x_i \) is the best point visited by the swarm, \( x \) is the particle's current location, \( x_p \) is the personal best location of the particle, \( R_1 \) and \( R_2 \) are random variables and \( \Psi_1 \) and \( \Psi_2 \) control the relative proportion of the cognitive and social interactions in the swarm.

In Basic PSO, each particle is known by its position \( (X_{ij}) \) and its velocity \( (V_{ij}) \) vectors, where \( i \) represents the particle index and \( j \) represents the dimension in the search space. The velocity vector of Basic PSO is the combination of three factors (e.g. last act / velocity, social (S) and cognitive (C) knowledge) as it is in equation (3).

Cognitive knowledge contains the position that particle achieves to his highest performance and is denoted by (local best position \( (P_{ij}) \)). Social knowledge contains the position where the particles and their neighbors found their highest group performance in and is denoted by (global best position \( (g_i) \)). Social knowledge can be affected by different neighborhood topologies (e.g. local-subset of the particles and global-all the particles). In local neighborhoods, the standard method is to set neighbors in a predefined way (e.g. using particles with the closest array indices as neighbors modulo the size of the population, henceforth known as a ring topology) regardless of the particles’ positions in the search space.

This method is the most commonly used for following two main motives:

1. In situations, in which, space coordinates represents mental abilities or skills, two very similar individuals may never come to meet in their lifetime, as to elements of the same family, which may differ significantly from each other, but still, they will always be neighbors.
2. this method has less computational effort required to processing the Euclidean distance, specially in problems with high level of dimensions (in Basic PSO, each bit considered as a dimension, with 2 possible values 0 or 1, and particle motion toggled between these two values) [24].

Equations 3 and 4 show the velocity and new position of particles criteria respectively.

\[ \ddot{v}_{ij}(t) = w \times \dot{v}_{ij}(t-1) + C + S \] \hspace{1cm} (3)
\[ C = c_1 \times c_2 \times (\ddot{P}_{ij}(t-1) - \ddot{X}_{ij}(t-1)) \]
\[ S = c_3 \times c_4 \times (\ddot{g}_{ij}(t-1) - \ddot{X}_{ij}(t-1)) \]
\[ \ddot{x}_{i,j}(t) = \dot{x}_{i,j}(t-1) + \dot{v}_{i,j}(t) \]  

(4)

Where \( r_{i,j} \) and \( r_{j} \) are different random values in the range [0, 1] following the uniform distribution. The parameters \( c_1 \) and \( c_2 \) are known as acceleration coefficients. The velocity components of the particle \( v_{i,j} \) is limited to a range as follow:

\[
v_{i,j} = \begin{cases} 
& v_{\min} \text{ if } v_{i,j} \leq v_{\min} \\
& v_{\max} \text{ if } v_{i,j} \geq v_{\max} \\
& v_{i,j} \text{ otherwise}
\end{cases}
\]

(5)

The value of \( V_{\max} \) is defined as one half of the total search range. The term inertia weight \( w \) in equation (3) is decreased linearly with time as suggested in:

\[
w = (w_i - w_\text{f}) \times \frac{\text{MAXITER} - t}{\text{AXITER}} + w_2
\]

(6)

Where, \( w_i \) and \( w_2 \) are the initial and final values, respectively. The inertia weight controls the impact of the previous velocity, which is a large inertia weight favors exploration, while a small inertia weight favors exploitation. Thus, a high inertia weight at the beginning of the search helps in exploring the search space by avoiding local minima, while decreasing the inertia weight as the search proceeds helps in exploiting the search space and converging to the optimal solution. \( t \) is the current iteration number and \( \text{MAXITER} \) is the maximum number of allowed iterations before termination. The local and global best positions of particles at time step \( t \) are updated as:

\[
P_{i}(t) = \begin{cases} 
& P_{i}(t-1) \text{ if } f(x_i(t)) \geq f(P_{i}(t-1)) \\
& x_i(t)
\end{cases}
\]

(7)

III. DIFFERENT OPTIMIZATIONS FOR PSO

A. Particle Swarm Optimization with Spatial Particle Extension (SEPSO)

This modified version of PSO is firstly published by Krink, Vesterstrom and Riget in order to solve the premature convergence of Basic PSO in iterative optimization [2], [3]. The idea is to give particles an extension in space to solve the closely clustering of Basic PSO particles. Krink, Vesterstrom and Riget point out that:

1. Particles convergence towards an optimum (local or global) and stay at that position without a chance to escape
2. Basic PSO is unreliable for the tasks in which the all targets and goals must be found in the specific time or those that the task must be done however it is possible (e.g. real-time problems in which the best solution should be found in a limited time constraint).
3. In the situations which the identified optimum is a local position, the performance would be better if we let some of the particles explore other areas of the search space and environment while the remaining particles stay at this optimum to fine-tune the solution.

For solving the premature convergence problem of Basic PSO, they offer to increase the diversity when particles start to cluster. In their method, if two particles are going to collide (e.g. particles collide and cluster in the situations that they are going to converge towards a position or in a situation which they are flickering near each other), Krink, Vesterstrom and Riget offer one of these three actions in this situation:

1. Random bouncing, where the particles are sent away from the collision in random directions preserving the old speed.
2. Realistic physical bouncing.
3. Simple velocity-line bouncing in which the particles continue to move in the direction of their old velocity-vector, but with a scaled speed (e.g. they have the possibility of making a U-turn and return to where they have come from (by scaling with a negative bounce-factor)).

B. The Gregarious Particle Swarm Optimizer (G-PSO)

This version of PSO is introduced by Pasupuleti and Battiti with the aim of solving the premature convergence problem by giving a reinitialized velocity to the particles which are stuck near the global best position [7]. They call the algorithm Gregarious Particle Swarm Optimization (e.g. global best position is the only shared information between particles during the iterations).

In G-PSO, each particle will either take a step towards the global best position or be reinitialized if it gets very close to the global best position. The new velocity equation is as follow:

\[
\text{if } \| (x_i(t) - g(t-1)) \| \leq \varepsilon \forall j v_{i,j}(t) = \text{rand}_\varepsilon(v_{\min}, v_{\max})
\]

(8)

\[
\text{else } \forall j v_{i,j}(t) = \gamma \times \text{rand}_\varepsilon((0,1)) \times (x_{i,j}(t-1) - g_j(t-1))
\]

\( \gamma \) Parameter determines the step size of each particle in the direction of the global best position and used for controlling particles movement (e.g. large values of \( \gamma \) will make the current particle \( i \) miss the global best position (oscillations) and small values of \( \gamma \) cause small step sizes and a slow convergence around the global best position). As has been mentioned before, particles velocity can change in the range of \([V_{\min}, V_{\max}]\) and the value of \( \gamma \) factor is also bounded in the range of \([\gamma_{\min}, \gamma_{\max}]\) and it is linearly adjusted at the end of every iteration as follow:

\[
\gamma = \begin{cases} 
& \max(\gamma - \delta, \gamma_{\min}) \text{ if } f(g(t)) < f(g(t-1)) \\
& \min(\gamma + \delta, \gamma_{\max}) \text{ otherwise}
\end{cases}
\]

(9)
Pasupuleti and Battiti set the $\delta=0.5$, $\gamma_0=3$ and they also limit $\gamma$ in the range of $[2, 4]$. They control particles performances by changing the value of the $\gamma$ factor in this way: while the step size of $\delta$ is robust they use this method: Each component of the difference $(x_i(t-1)-g(t-1))$ in equation (8) is multiplied by a value factor of 1.5 initially. If the global best value is improved, the value of $\gamma$ decreases up to a minimum value of 2 and as the performance improves; the particles converge towards the global best position, which results in an aggressive search for local minima. If the global best value does not improve during the iterations, the value of $\gamma$ increases up to a maximum value of 4.0, which would make the particles explore the search space by oscillating and flickering around the current minimum, in order to find better directions.

The major difference between Basic PSO and G-PSO is that there is no update of the personal best position here, as the particles do not memorize their past search history and the velocity update equation is based on the global best position. G-PSO has two advantages as below:

1. Because of its greedy search for the local minima, it will find good and promising regions rapidly during the initial phase of the search.
2. Because of its capability to reinitialize with random velocities, particles will always try to find a better global best position and they do not loose the exploration capability.

C. Division of Labor in Particle Swarm Optimization (DOL-PSO)

This method is introduced by Vesterstrom and Riget with the aim of fixing the lack of dynamic velocity adjustment resulting in the inability to hill climb solutions problem [2], [9]. By this method they try to connect the resilience level on the swarm level and the flexibility on the particles level (e.g. the swarm level on the swarm is the level in which the swarm responds to alternations, and the flexibility of the particles level is the level in which particles observe the environment in a dynamic mode).

In this method, each individual particle has a response threshold which will be shown by $\theta_{ij}$. The other parameter which is introduced is $S_{ij}$ which is the level of the task-related stimuli and may have 3 different situations:

I : $S_{ij}<<\theta_{ij}$  
II : $S_{ij}>>\theta_{ij}$  
III : $S_{ij} = \theta_{ij}$

They call parameter $T$ the response function that outputs the probability of individual $i$ engaging in task $j$ given a stimulus $S_{ij}$ and a threshold $\theta_{ij}$.

$$P(X_{ij} = 0 \rightarrow X_{ij} = 1) = T_{ij}(s) = \frac{s^*}{\theta^*_s + s^*} \quad (11)$$

Where, $X_{ij}$ is the state of particle $i$. If $X_{ij} = 0$ corresponds to inactivity and $X_{ij} = 1$ corresponds to performing a given task. $\theta_{ij}$ is denoted as response threshold of particle $i$ with the associated stimulus $s$. $P$ is a probability per time step for a particle to perform the given task. Likewise, an active particle gives up task performance and becomes inactive with probability $p$ per time step:

$$P(X_{ij} = 1 \rightarrow X_{ij} = 0) = p \quad (12)$$

In DOL-PSO, the basic aim is to find an optimization for the convergence problem around the local optimum position in PSO. For such an aim a task as local search is introduced and it is associated with a stimulus $s$ (e.g. defined as the time steps since the last fitness improvement). An individual particle engages in the local search task, whenever it realizes that it has not been successful for a period of time.

The stimulus $s$ factor increases when the fitness does not improve (i.e. when the search task is not performed efficiently enough). The idea is that when particles are engaged in the local search around the optimum position, the diversity decreases and therefore faster convergence occurs. According to the velocity equation in the Basic PSO model, when all particles swarm near the optima, the particles are pulled towards the $g(t)$ (global best position), but the inertia of the particles themselves make them flicker around somewhat aimlessly in their attempt to turn around and return to the point of attraction.

In this modified version, response threshold is a constant which sets in the initial state but Vesterstrom and Riget also use a dynamic response threshold which could have different values during the iterations.

They try to give an age to the response threshold by changing or decreasing it over time. They also decrease the response threshold each time that they needed a local search.

$$\theta_{ij}(t) = b x e^{-at} ; a, b \in \mathbb{R}^+ \quad (13)$$

By such a method they could control the number of particles which are involved and switch to the local search task and they also used a diversity equation to measure the diversity of the swarm as below:

$$\text{diversity}(s) = \frac{1}{|s|} \sum_{i=1}^{N} \sqrt{(\sum_{j=1}^{n}(p_{i,j} - \bar{p}_{j})^2)} \quad (14)$$

where $|s|$ is the swarm-size.

D. Controlling Diversity in Particle Swarm Optimization (ARPSO)

This method is introduced by Vesterstrom and Riget in [2], [4]. In this study they pointed out that a decreasing diversity in the search field can cause premature convergence and fitness stagnation (e.g. the diversity factor can cause the algorithm to switch between the exploring and exploiting modes).
In ARPSO, algorithm is based on switching between two phases (e.g. attraction and repulsion). A new parameter \( \text{dir} \in [-1, 1] \) is introduced to help the algorithm to be able to switch between these phases. In their idea when \( \text{dir} = 1 \) the algorithm works on attraction phase and when \( \text{dir} = -1 \) it works on repulsion phase. In the attraction phase the diversity factor decreases until it reaches to a \( d_{low} \) which is the lowest value for the diversity factor. In this situation algorithm switches to the repulsion phase and therefore the swarm expands and the diversity factor increases during iterations until it reaches to the \( d_{high} \) level. In such a level algorithm switches again to the attraction phase. In their study, parameters were set as \( d_{high} = 0.25 \) and \( d_{low} = 5 \times 10^{-6} \) and a new velocity equation was used as below:

\[
V(t+1) = \omega V(t) + \text{dir} (\phi_1(p(t) - x(t)) + \phi_2(g(t) - x(t)))
\]  

(15)

As results show, almost all of the fitness appeared in the attraction phase shows the importance of the low diversity and therefore omitting the evaluation of the fitness function in the repulsion phase was recommended. Results show that, 11% to 30% of the evaluations are calculated in the repulsion phase and only 1% to 3% of the improvements occur in that phase. Vesterstrom and Riget also recommend clearing the \( g(t) \) every time that the algorithm wants to enter the attraction phase (e.g. the task is solved better and therefore the swarm can explore freely in the search space instead of constantly being dragged and converged towards the old \( g(t) \) during the exploration).

E. Green House PSO

The Tomato-Producing PSO (Green house PSO), a Real World Dynamic Problem, is introduced by Riget and Vesterstrom [2]. During the work they face with some difficulties such as boundaries problem in feasible solution space (e.g. they could not evaluate particles outside of the search limits). There are various solutions for such a problem such as:

1. To force and bring outside particles in to inside space.
2. Assigning fitness penalties to particles outside of the boundaries.

However, both of the solutions have some weaknesses. In the situations that optimal solutions are nearby or at the boundary, there is a huge risk to use first solution because all particles might get trapped at the boundary without a chance to escape and therefore performance would be decreased. This happens when all neighborhood best positions are at the boundary. Also, if all of particle's velocity vector point outside the search space, the particles would have no chance to escape form boundary.

With the punishing outside particles solution, it would be problematic to find an optimum at the boundary, since particles are moving in and out of the search space. They also realized that, using a constant \( w \) is better than a linearly decreasing one. It is essential for the performance of the PSO algorithm in dynamic environments.

Furthermore, they concerned that, online changes of greenhouse that appears by the result of its real time area, have effects on the \( p \) factor. However, they suggested to let particles reset their memory of their best known positions \( p(t) \) at some point during execution. Memory resetting solution was implemented by two different methods: periodic resetting (e.g. based on a simple iteration count) and triggered resetting (e.g. based on the magnitude of the change in the environment and search space). Both of the solutions suffered from some weaknesses.

1. The method was based on dynamic problems consist of a single optimum moving linearly at a constant rate, which does not reflect most real world scenarios.
2. The evaluation function used for measuring the quality of the algorithm was based on geometric distance to the optimum, which, by all means, was not ingenious, and could not reflect the algorithm's quality properly.
3. The triggered resetting solution relied on the entire environment changing at the same rate. The suggested implementation used a sentry particle which had the task to detect any changes in the environment. Therefore, changes could only be detected in the immediate neighborhood of the sentry particle while it might be occurred anywhere in the environment, and outside of sentry particle neighborhood and it might be undetected.

According to the last issues, Riget and Vesterstrom decided to let particles move outside the search space. They evaluate particles which were out side of boundary at the closest boundary solution. They realized that because of the attraction towards their own and their neighborhood's best positions, the particles will return to the search space.

They also considered that, the most important issue in dynamic environments is the memory. They suggested a way to use periodic resetting of best known particle position \( p(t) \). In their idea, the greenhouse state was updated with the best controller setting for one time-step. Therefore, in conclusion, they assume that the environment has changed, and they reset the position vector \( p(t) \) every time a new best control setting was found.

F. Hierarchical Particle Swarm Optimizer with Time Varying Acceleration Coefficients (HPSO-TVAC)

HPSO-TVAC is a self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. In this method, the inertia weight is set to zero in the velocity equation and therefore the velocity of particles and their movement is based only on the cognitive and social components. Although, particles found a local optimum solution rapidly and stagnate due to the lack of momentum, the optimum solution would be highly dependent to the population initialization. Furthermore, if particles stagnate in
any direction and dimension \((v_j=0)\) they would be reinitialized with a random velocity. The acceleration coefficients \(c_1\) and \(c_2\) which were used in velocity equation (3) in the section 2 were varied linearly with time to increase the global search in the beginning parts of the optimization and to make the particles converge toward global optima at the end of the search. The linearly decreasing equations for acceleration coefficients are as follow:

\[
c_1 = [c_{1f} - c_{1i}] \left[ \frac{iter}{\max_iter} \right] + c_{1i} \tag{16}
\]

\[
c_2 = [c_{2f} - c_{2i}] \left[ \frac{iter}{\max_iter} \right] + c_{2i}
\]

Pasupuleti and Battiti in [7] and Ratnaweera and Halgamuge in [5], [6] use this method and their result shows that they could have better performances when they change \(c_1\) from 2.5 to 0.5 and \(c_2\) from 0.5 to 2.5, over the full range of the search. They also prove that by using fixed coefficients and setting them equal to 2 the performance is significantly poor.

G. Particle Swarm Optimizer with mutation and Time Varying Acceleration Coefficients (MPSO-TVAC)

In this method, a mutation factor is introduced as a performance-enhancing strategy to improve global search capability of the swarm and particles by providing more controls on diversity and increasing it. The mutation operator in PSO is conceptually equivalent to the mutation in GA (e.g. in GA, a search for the global optimum solution is mostly guided by crossovers between the generations (crossover operator) and a mutation operator will introduce new genetic materials into the individuals to enhance the search in new areas within the search space).

In MPSO, problems of premature convergence and lack of diversity can be controlled by choosing a particle randomly when the global optimum solution does not improve during certain iterations (generations). Hence a random perturbation (mutation step size which is defined as \(v_{max}\)) would be added to a randomly selected modulus of the velocity vector of the selected particle by a predefined probability (mutation probability). Although researchers’ results in [6] show that MPSO strategy performed poorly by fixed acceleration coefficients, MPSO-TVAC shows competitive results with original PSO.

H. Constriction Factor Particle Swarm Optimization (CPSO)

The idea of using a constriction factor \((K)\) is firstly introduced by Clerc in [19] with the aim of solving the lack of diversity in Basic PSO. Eberhart and Shi also prove that this method can achieve to better quality solutions compared to the original PSO.

Particles converge rapidly in the first part of the search in the Basic PSO and then slow down or stop (lack of diversity). In such a situation particles are mostly stagnated not far away from global best solution. Stacey, Janic and Grundy in [8] suggest that if an extra velocity is added to one or some of these particles there would be chances of generating new solutions which might lead a swarm to the global best solution (e.g. global best will attract all the members of a swarm and by mutating a single particle there will be chance to lead the swarm out of the trapped position if the mutated particle becomes the global best). To use such a mechanism, a mutation operator is introduced to mutate a position vector (solution) with the probability of \(P_{mua}\). In this method the probability of each component to be mutated in a solution is \(l/d\) where \(d\) is the number of components in the vector. This mechanism provides means to escape from local optima and also speed up the search [8].

The equations which are used for this method is as follow:

\[
v_{i}^t = K [v_{i}^t + c_1 r_{1i}(p_{i}^t - x_{i}^t) + c_2 r_{2i}(g_{i} - x_{i}^t)]
\]

\[
K = \frac{2}{2 - \phi - \sqrt{\phi^2 - 4\phi}} , \text{where } \phi = c_1 + c_2
\]

Where \(K\) is only defined for \(\phi>4\), \(c_1\), \(c_2=2.05\) and \(K=0.73\). Clerc in [8], [19] also introduces a modified version for this method which is called CPSO by changing \(r_{1i}\) and \(r_{2i}\) and using new equation as follow:

\[
v_{i}^t = K [v_{i}^t + c_1 \frac{r_{1i}}{r_i}(p_{i}^t - x_{i}^t) + c_2 \frac{r_{2i}}{r_i}(g_{i} - x_{i}^t)]
\]

\[
\text{where } r_i = r_{1i} + r_{2i}
\]

This new elements ensure that when \(c_1\) is equal to \(c_2\) and the sum of \(c_1 \frac{r_{1i}}{r_i} + c_2 \frac{r_{2i}}{r_i}\) is no longer random.

IV. CONCLUSION

This paper presents various modified versions of PSO which are designed to work with the aim of forcing particles to converge to the global optimum. Although these methods try to solve different problems of PSO with the aim of achieving to better performances, most of these algorithms have been implemented to work on specific situations and search spaces, and therefore they do not have the generality or even if they have it their results are unstable in different scenarios. This is because most of the implementations are domain dependent. We should also consider that these methods do not have the ability to solve multi dimensional problems and they still cannot be used in the real world in scenarios with constriction of result timing and also dynamic environments.

We suggest that using a changeable global best position and a velocity factor base on the efficiency of each part of velocity equation (cognitive, social or last velocity) or even
various combinations between them can be more general and effective. We strongly believe that such a policy can help PSO to be more accurate and reliable in real-time and dynamic environments. We can also provide the generality of the global best change diversity in various problem domains.

REFERENCES


[12] Y. Zhao, J. Zheng, Particle Swarm Optimization algorithm in Signal Detection and Blind Extraction, conf, IEEE, Parallel Architectures,


