Chapter 5

Particle Swarm Optimization

5.1. Introduction

In this chapter the fundamental concepts of PSO is presented. Next section deals with the introduction to PSO and its basic algorithm. PSO suffers from a serious limitation of premature convergence. Section 5.3 addresses this and other limitations of using PSO in image clustering. In the next section 5.4 some proposed improvements in PSO have been presented. Finally, based on some experimental results, a selection is made as which version of PSO to be used in the present work and conclusion of this study is presented.

5.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a swarm intelligence optimization technique that was inspired by the behaviour of flocks of birds [Kennedy and Eberhart, 1995]. It is a kind of intelligence that is based on social-psychological principles and provides insights into social behavior, as well as contributing to engineering applications. The recent years have seen tremendous growth in the use of PSO in various areas of computing ranging from networking, classification, scheduling, and training of artificial neural network to feature extraction and image processing.

The advantage of PSO over many of the other optimization algorithms is its relative simplicity and ease of use. PSO shares many similarities with evolutionary computation
techniques such as Genetic Algorithms (GA) [Goldberg 1993]. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. The potential solutions, called particles, fly through the problem space by following the current optimum particles.

In the basic PSO a problem is given, and some way to evaluate a proposed solution to it exists in the form of a fitness function. A communication structure or social network is also defined, assigning neighbors for each individual to interact with. Then a population of individuals defined as random guesses at the problem solutions is initialized. These individuals are called the candidate solutions. They are also known as the particles, hence the name particle swarm. An iterative process to improve these candidate solutions is set in motion. The swarm of individuals (called \textit{particles}) flies through the search space.

The position of a particle is influenced by the best position visited by itself (i.e. its own experience) and the position of the best particle in its neighborhood (i.e. the experience of neighboring particles). When the neighborhood of a particle is the entire swarm, the best position in the neighborhood is referred to as the global best particle, and the resulting algorithm is referred to as a \textit{gbest} PSO. When smaller neighborhoods are used, the algorithm is generally referred to as a \textit{lbest} PSO [Shi and Eberhart, 1998a]. The performance of each particle (i.e. how close the particle is from the global optimum) is measured using a fitness function that varies depending on the optimization problem. Each particle in the swarm is represented by the following characteristics:

\( x_i \): The \textit{current position} of the particle;
$v_i$: The current velocity of the particle;

$y_i$: The personal best position of the particle.

$\hat{y}_i$: The neighborhood best position of the particle.

The personal best position of particle $i$ is the best position (i.e. the one resulting in the best fitness value) visited by particle $i$ so far. Let $f$ denote the objective function. Then the personal best of a particle at time step $t$ is updated as

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & \text{if } f(x_i(t+1)) < f(y_i(t)) \end{cases}$$  \hfill (5.1)

The gbest model: In this model, the best particle is obtained from the entire swarm by selecting the best personal best position. The position of the global best particle is given by,

$$\hat{y}(t) \in \{y_0, y_1, \ldots, y_s\} = \min \{f(y_0(t)), f(y_1(t)), \ldots, f(y_s(t))\}$$  \hfill (5.2)

where $s$ denotes the size of the swarm. The velocity of particle $i$ is updated using the following equation.

$$v_{i,j}(t+1) = w v_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + c_2 r_{2,j}(t) (\hat{y}_j(t) - x_{i,j}(t))$$  \hfill (5.3)
Where \( w \) is the inertia weight, \( c_1 \) and \( c_2 \) are the acceleration constants, and \( r_{1,j} \) and \( r_{2,j} \) are factors lying between \((0,1)\).

In the equation 5.3 there are additional three terms defined as follows:

The *inertia weight* term \((w)\): [Shi and Eberhart 1998b]. This term serves as a memory of previous velocities. The inertia weight controls the impact of the previous velocity: a large inertia weight favors exploration, while a small inertia weight favors exploitation.

The *cognitive component* \((y_i(t) - x_i)\): This term represents the particle's own experience as to where the best solution is.

The *social component* \((y(t) - x_j)\), which represents the belief of the entire swarm as to where the best solution is.

The position of particle \( i \), \( x_i \), is then updated using the following equation:

\[
x_i(t+1) = x_i(t) + v_i(t+1) \tag{5.4}
\]

Algorithm 5.1 presents the main steps of basic PSO algorithm. The particles in the swarm are updated according to equations 5.3 and 5.4. This updation takes place for a specified number of iterations or when the velocity updates are close to zero. There have been many versions of PSO proposed time to time on the basis of accuracy, speed or overall performance. Most important of them are Binary PSO [Kennedy and Eberhart 1997], Guaranteed Convergence PSO (GCPSO), Clamped PSO, Hybrid PSO,
Coevolutionary PSO, Repulsive PSO, Multi-objective PSO, Adaptive PSO (APSO), Discretized PSO etc.

```
Initialize population

Do

    For particle i=1 to Swarm size S

    if \( f(x_i) < f(p_i) \), then \( p_i = x_i \)

    \( p_g = \{ p_i | \min f(p_j) \quad j=1,2,...N \} \)

    For \( d=1 \) to Dimension \( D \)

    Update \( v_{id} \) using equation 5.3

    If \( v_i > v_{\text{max}} \) then \( v_i = v_{\text{max}} \)

    else if \( v_i \leq -v_{\text{max}} \) then \( v_i = -v_{\text{max}} \)

    Update \( x_{id} \) using equation 5.4

Next d

Next i

Until termination criteria is met.
```

**Algorithm 5.1 The basic PSO**
5.3 Limitations of PSO

PSO and other stochastic search algorithms have a major drawback of premature convergence. Although PSO finds good solutions much faster than other evolutionary algorithms, it usually cannot improve the quality of the solutions as the number of iterations is increased [Angeline 1998]. As the swarm iterates, the fitness of the global best solution improves (decreases for minimization problem). It could happen that all particles being influenced by the global best eventually approach the global best, and from there on the fitness never improves despite however many runs the PSO is iterated thereafter. The particles also move about in the search space in close proximity to the global best and not exploring the rest of search space. This phenomenon is called 'convergence'. PSO usually suffers from premature convergence when strongly multi-modal problems are being optimized.

The rationale behind this problem is that, for the gbest PSO, particles converge to a single point, which is on the line between the global best and the personal best positions. This point is not guaranteed to be even a local optimum. Another reason for this problem is the fast rate of information flow between particles, resulting in the creation of similar particles (with a loss in diversity) which increases the possibility of being trapped in local optima [Riget and Vesterstrom 2002]. If the inertial coefficient of the velocity is small, all particles could slow down until they approach zero velocity at the global best. The selection of coefficients in the velocity update equations affects the convergence and the
ability of the swarm to find the optimum. One way to come out of the situation is to
reinitialize the particles positions at intervals or when convergence is detected.

Numerous techniques for preventing premature convergence have been proposed.
Some research approaches investigated the application of constriction coefficients and
inertia weights. Many variations on the social network topology, parameter-free, fully
adaptive swarms, and some highly simplified models have been created. Most important of
them are discussed in the next section.

5.4 Improvements in Convergence Behavior of PSO

5.4.1 Inertia weight model: The inertia weight term, c0, which was first introduced by Shi
and Eberhart [1998a], serves as a memory of previous velocities. The inertia weight
controls the impact of the previous velocity: a large inertia weight favours exploration,
while a small inertia weight favours exploitation [Shi and Eberhart, 1998b]. The modified
velocity update equation is given by equations 5.3 and 5.4.

5.4.2 Constriction factor model: A constriction factor can be used to choose values for w,
c1 and c2 to ensure that the PSO converges. The modified velocity update equation is
defined as follows:

\[ v_{i,j}(t+1) = \chi (v_{i,j}(t) + c_1r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + c_2r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t))) \]  (5.5)

Here, \( \chi \) is the constriction factor defined as follows:

\[ \chi = \frac{2}{2 - \phi - \sqrt{\phi^2 - 4\phi}} \]  (5.6)
and,

\[ \phi = c_1 + c_2, \quad \phi > 4 \]

Use of the constriction factor and velocity clamping together generally improves both the performance and the convergence rate of the PSO [Eberhart and Shi 2000].

### 5.4.3 Guaranteed Convergence PSO (GCPSO)

The basic PSO converges prematurely because the velocity update equation depends only on the term \( w v_i(t) \). GCPSO [Van den Bergh 2002] avoids this by using a different velocity update equation, 5.7.

\[
v_{r,j}(t+1) = -x_{r,j}(t) + \hat{y}_j(t) + w v_{r,j}(t) + \rho(t)(1 - 2 r_{2,j}(t)) \tag{5.7}
\]

The resulting equation for position update is given by the following the equation 5.8.

\[
x_{r,j}(t+1) = \hat{y}_j(t) + w v_{r,j}(t) + \rho(t)(1 - 2 r_{2,j}(t)) \tag{5.8}
\]

The term \( \rho(t) \) defines the area in which a better solution is searched. The value of \( \rho \) is given is initialized to 1.0, with \( \rho(t+1) \) defined on the basis of number of successes and failures.

\[
\rho(t+1) = \begin{cases} 
2 \rho(t) & \text{if } \# \text{ successes } > s_c \\
0.5 \rho(t) & \text{if } \# \text{ failures } > f_c \\
\rho(t) & \text{otherwise}
\end{cases} \tag{5.9}
\]

Van den Bergh suggests repeating the algorithm until \( \rho \) becomes sufficiently small, or until stopping criteria are met. Stopping the algorithm once \( \rho \) reaches a lower bound is
not advised, as it does not necessarily indicate that all particles have converged – other particles may still be exploring different parts of the search space. It is found that GCPSO has guaranteed local convergence whereas the original PSO does not.

5.4.4. Attractive and Repulsive PSO (ARPSO)

There are two phases between which ARPSO [Riget and Vesterstrøm 2002] alternates. In the attraction phase, PSO is used for fast information flow, as such particles attract each other and thus the diversity reduces. In this phase 95% of fitness improvements can be achieved. This observation shows the importance of low diversity in fine tuning the solution. In the repulsion phase, particles are pushed away from the best solution found so far thereby increasing diversity. ARPSO was found to give better results than PSO and GA in most of the test cases.

5.4.5. Multi-start PSO (MPSO)

Proposed by Van den Bergh [2002] MPSO tries to make GCPSO a global search algorithm. It works as follows:

1. Randomly initialize all the particles in the swarm.

2. Apply the GCPSO until convergence to a local optimum. Save the position of this local optimum.

3. Repeat Steps 1 and 2 until some stopping criteria are satisfied.

In Step 2, the GCPSO can be replaced by the original PSO. Several versions of MPSO were proposed by Van den Bergh [2002] based on the way used to determine the convergence of GCPSO.
5.4.6. Techniques using Mutation

Some techniques have been proposed which use a hybrid of PSO with Gaussian mutation [Higashi and Iba 2003] or hybrid lbest- and gbest- PSO with a non-uniform mutation Operator [Esquivel and Coello 2003]. These hybrid techniques have given better results than PSO and GPSO in all the experiments conducted. One more similar technique is Dissipative PSO (DPSO) [Xie et al. 2002] that adds random mutation to PSO in order to prevent premature convergence. DPSO introduces negative entropy via the addition of randomness to the particles. The results showed that DPSO performed better than PSO when applied to the benchmarks problems.

5.4.7. Differential Evolution PSO (DEPSO)

DEPSO [Zhang and Xie 2003] uses a differential evolution (DE) operator [Storn and Price 1997] to provide the mutations. A trait point is calculated as follows:

If \((r_1(t) < p_c \ OR \ j = k_d)\) then

\[
\tilde{y}_{i, j}(t) = \hat{y}_{j}(t) + \frac{(y_{1, j}(t) - y_{2, j}(t)) + (y_{3, j}(t) - y_{4, j}(t))}{2}
\]

\(5.10\)

where \(r_1(t), k_d, y_1(t), y_2(t), y_3(t)\) and \(y_4(t)\) are randomly chosen from the set of personal best positions. To avoid disorganization of the swarm, \(y_i(t)\) is mutated instead of \(x_i(t)\). DEPSO works by alternating between the original PSO and the DE operator such that equation 5.3 and 5.4 are used in the odd iterations and equation 5.9 is used in the even iterations. The performance of DEPSO has been found to be better than other popular versions of PSO in case of benchmark functions.
5.4.8. Fitness-Distance Ratio PSO (FDR-PSO)

A new term was added to the velocity update equation by Veeramachaneni et al. [2003]. The new term allows each particle to move towards a particle in its neighborhood that has a better personal best position. The modified velocity update equation is given by:

\[ v_{i,j}(t+1) = w v_{i,j}(t) + \psi_1 (y_{i,j}(t) - x_{i,j}(t)) + \psi_2 (\hat{y}_{j}(t) - x_{i,j}(t)) + \psi_3 (y_{n,j}(t) - x_{i,j}(t)) \quad (5.11) \]

where, \( \psi_1, \psi_2 \) and \( \psi_3 \) parameters specified by users. Here each \( y_n \) is chosen in a way such that the following is maximized.

\[ \frac{(f(x_i(t)) - f(y_n(t)))}{|y_{n,j}(t) - x_{i,j}(t)|} \quad (5.12) \]

The performance of FDR-PSO is found to be better than PSO, ARPSO, DPSO, SOC PSO etc. In addition to above techniques, a scoring based method was developed by Chandra et al. [2009] to control the convergence of PSO. The following algorithm was used for this purpose:

5.5 Experimental results

Out of the PSO versions discussed in section 5.4 three promising algorithms on the basis of previous results [van den Berg and Engelbrecht 2005] were selected and experimented with some benchmark functions as detailed in table 5.1. These algorithms are PSO, Attractive and Repulsive PSO (ARPSO), Guaranteed Convergence PSO
(GCPSO) and Fitness-Distance Ratio based PSO (FDR-PSO). The results are shown in table 5.2.

**Table 5.1** : Some benchmark functions used for testing PSO

<table>
<thead>
<tr>
<th>Function</th>
<th>Definition</th>
<th>Type</th>
<th>Range</th>
<th>Initial Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>$f_s(x) = \sum_{d=1}^{D} x_d^2$</td>
<td>Unimodal</td>
<td>[-50,50]</td>
<td>[25,40]</td>
</tr>
<tr>
<td>Quartic</td>
<td>$f_q(x) = \sum_{d=1}^{D} dx_d^4$</td>
<td>Unimodal</td>
<td>[-20,20]</td>
<td>[10,16]</td>
</tr>
<tr>
<td>Rosenbrook</td>
<td>$f_{Ro}(x) = \sum_{d=1}^{D-1} 100(x_{d+1} - x_d)^2 + (x_d - 1)^2$</td>
<td>Unimodal</td>
<td>[-100,100]</td>
<td>[50,80]</td>
</tr>
<tr>
<td>Griewank</td>
<td>$f_g(x) = \frac{1}{4000} \sum_{d=1}^{D} x_d^2 - \prod_{d=1}^{D} \cos(\frac{x_d}{\sqrt{d}}) + 1$</td>
<td>Multi-modal</td>
<td>[-600,600]</td>
<td>[300,500]</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>$f_{Ra}(x) = \sum_{d=1}^{D} x_d^2 + 10(1 - \cos(2\pi x_d))$</td>
<td>Multi-modal</td>
<td>[-5.12,5.12]</td>
<td>[1,4.5]</td>
</tr>
</tbody>
</table>

**5.5.1 Parameter Setting**: The various parameters of PSO are to be defined by user. These parameters include population size, inertia weight, acceleration constants etc. and are vary from one problem to another.
Population Size: The PSO researchers have suggested the population size to be 2n to 5n where n is the number of decision variables. But, most researches have taken it to be constant. In this study the population size was varied from 20 to 50 and best results were obtained at population size of 35.

Inertia Weight: A linearly decreasing inertia weight of the order (0.9-0.4) is found to give good result.

Acceleration Constant: c1=c2= 1.8

Initialization of Swarms: The particle swarms were initialized by the Gaussian distribution given by,

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$ (5.13)

Diversity: Diversity is measured by the following equation:

$$Div(S(t)) = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\sum (x_{ij}(t) - \overline{x_j(t)})^2}$$ (5.14)

From table 5.2, it is evident that GCPSO and FDR-PSO give better results than others in most of the benchmark functions. These two versions of PSO are hybrid with Scoring based method (SBM) and tested again on the same functions. The results obtained are shown in table 5.3. From table 5.3 it appears that FDR – PSO, when clubbed with scoring based method, gives better results in most of the benchmark functions, as long as the convergence behavior is concerned
Table 5.2. Results of PSO, ARPSO, GCPSO and FDR-PSO on benchmark functions
(First row shows mean, second row shows diversity while the third one indicates
deviation). The best mean values are shown in **bold**.

<table>
<thead>
<tr>
<th>Function</th>
<th>PSO</th>
<th>ARPSO</th>
<th>GCPSO</th>
<th>FDR-PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>1.1956e-46</td>
<td>9566e-46</td>
<td>0.9566e-46</td>
<td>1.0956e-46</td>
</tr>
<tr>
<td></td>
<td>1.9868e-18</td>
<td>1.9868e-18</td>
<td>1.9868e-18</td>
<td>2.59868e-18</td>
</tr>
<tr>
<td></td>
<td>4.8961e-8</td>
<td>4.8961e-8</td>
<td>4.8961e-8</td>
<td>4.8741e-8</td>
</tr>
<tr>
<td>Quartic</td>
<td>12.3426e-12</td>
<td>11.4628e-11</td>
<td><strong>28.3426e-14</strong></td>
<td>14.4427e-12</td>
</tr>
<tr>
<td></td>
<td>11.6743e-16</td>
<td>14.7645e-16</td>
<td>21.6743e-15</td>
<td>15.5482e-16</td>
</tr>
<tr>
<td></td>
<td>16.9812e-6</td>
<td>17.67412e-6</td>
<td>36.9812e-14</td>
<td>19.8773e-7</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>21.9265</td>
<td>4.4057</td>
<td><strong>3.9951</strong></td>
<td>9.9928</td>
</tr>
<tr>
<td></td>
<td>4.2378</td>
<td>2.9044</td>
<td>1.8737</td>
<td>2.5279</td>
</tr>
<tr>
<td></td>
<td>8.6274e+03</td>
<td>4.1212</td>
<td>3.9593</td>
<td>3.1689</td>
</tr>
<tr>
<td>Griewank</td>
<td>0.031646</td>
<td>0.0110</td>
<td>0.0116</td>
<td><strong>0.0047</strong></td>
</tr>
<tr>
<td></td>
<td>0.000710</td>
<td>0.0008</td>
<td>0.0015</td>
<td>1.6313e-08</td>
</tr>
<tr>
<td></td>
<td>0.025322</td>
<td>0.0191</td>
<td>0.0197</td>
<td>0.01266</td>
</tr>
<tr>
<td></td>
<td>0.01002</td>
<td>0.4279</td>
<td>0.0002</td>
<td>0.5437</td>
</tr>
<tr>
<td></td>
<td>17.2321</td>
<td>13.3956</td>
<td>11.9854</td>
<td>8.3498</td>
</tr>
</tbody>
</table>
Table 5.3 Results of GCPSO and FDR-PSO on benchmark functions by applying scoring based method.

<table>
<thead>
<tr>
<th>Function</th>
<th>GCPSO + SBM</th>
<th>FDR-PSO + SBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>0.8956e-47</td>
<td>0.0756e-48</td>
</tr>
<tr>
<td></td>
<td>2.0021e-19</td>
<td>1.2564e-18</td>
</tr>
<tr>
<td></td>
<td>3.4561e-10</td>
<td>3.7684e-8</td>
</tr>
<tr>
<td>Quartic</td>
<td>13.4526e-15</td>
<td>16.2157e-15</td>
</tr>
<tr>
<td></td>
<td>19.3743e-15</td>
<td>15.5482e-16</td>
</tr>
<tr>
<td></td>
<td>31.9812e-14</td>
<td>19.8773e-7</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>1.0054</td>
<td>1.8799</td>
</tr>
<tr>
<td></td>
<td>1.0834</td>
<td>2.4537</td>
</tr>
<tr>
<td></td>
<td>2.6955</td>
<td>2.4589</td>
</tr>
<tr>
<td>Griewank</td>
<td>0.0111</td>
<td>0.0022</td>
</tr>
<tr>
<td></td>
<td>0.0011</td>
<td>0.9314e-08</td>
</tr>
<tr>
<td></td>
<td>0.0112</td>
<td>0.0112</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>10.2741</td>
<td>8.6516</td>
</tr>
<tr>
<td></td>
<td>0.0001</td>
<td>0.5125</td>
</tr>
<tr>
<td></td>
<td>9.8837</td>
<td>6.7354</td>
</tr>
</tbody>
</table>
5.6 Conclusions

This chapter presented a brief introduction to PSO. Then the some modified versions of PSO were detailed to tackle the major drawback of PSO, i.e. premature convergence. The four algorithms namely PSO, GCPSO, ARPSO and FDR-PSO described in the chapter use different Gaussian initialization scheme for generating the swarm population. Two best versions were chosen from the preliminary comparative study of these versions. These versions were then hybrid with SBM to get the best version to be used in this work. It was found that FDR-PSO, when combined with SBM gives considerably better results.