CHAPTER IV

ALGORITHM OF ENHANCED ESTIMATION OF CENTROID (EEOC)

4.1. Introduction

Now a day, for construction and maintenance of any system, Scientist, engineers, economists, market analyst, and manager always has to take myriad innovative as well as administrative decisions. The decision making must be taken in an optimal way on account of the world turn out to be increasingly unpredictable and aggressive step by step. Apparently, optimization act as a noteworthy part of getting the best result under certain circumstances. Statistically, an optimization problem has a fitness function which portrays the issues under an arrangement of requirements that represents the solution space for the problem. In recent years, plethora optimization methods have been developed for solving various types of optimization problems. The popular method for solving complex engineering problems are particle swarm optimization, neural networks, genetic algorithm, ant colony optimization, artificial immune systems and fuzzy optimization.

Optimization determines the best-suited solution to problem under given circumstances. For example, a manager needs to take many technological and managerial plans at several times. The final destination of the plans is either to minimize the effort required or to maximize the desired benefit. Optimization refers to both minimization and maximization tasks. Since the maximization of any function \( f \) is mathematically equivalent to the minimization of its additive inverse \( -f \), the term minimization and optimization are utilized interchangeably. The minimization problem can be defined as follows

Given \( f: S \to \mathbb{R} \) where \( S \subseteq \mathbb{R}^{N_d} \) and \( N_d \) is the dimension of the search space \( S \)

Find \( x^* \in S \) such that \( f(x^*) \leq f(x), \forall x \in S \) \hspace{1cm} \ldots \ldots \ (4.1)
The variable \( x^* \) is called the global minimizers of \( f \) and \( f(x^*) \) is called the global minimum value of \( f \). The process of finding the global optimal solution is known as global optimization. An optimization problem may be linear or nonlinear. Apparently, nonlinear optimization problems are complicated to solve. Based on the problem characteristics, the optimization problems are classified for examples, constrained optimization, unconstrained optimization, global optimization, and local optimization.

### 4.2. Particle Swarm Optimization (PSO)

In 1995, Dr. Kennedy and Dr. Eberhart, introduced a new optimization algorithm [34] entitled as Particle Swarm Optimization (PSO). It is a novel population based stochastic search algorithm and an alternative solution to the complex non-linear optimization problem. The basic idea behind this algorithm was originally inspired by the simulation of the social behaviour of animals such as bird flocking, fish schooling and so on. It is working in the view of the natural process of group communication to share individual knowledge when a group of birds or fish search food or migrate. Even though all birds and fish do not know where the best position, if any member can find out a desirable path to go, then the rest of the members will follow quickly. In PSO, each member of the population is named as particles and the population is named as a swarm. Starting with a randomly initialized population and moving in randomly chosen direction, each particle goes through the searching space and remembers the best previous position of itself and its neighbours. Particle of a swarm communicates good position to each others as well as dynamically adjust their own position and velocity derived from the best position of all particles. The following step begins when all particles have been moved. At last, all particles tend to fly towards better and better positions over the searching process until the swarm move to close to an optimum of the fitness function equation eq 4.1.
The PSO method is becoming very popular because of its simplicity of implementation as well as ability to swiftly converge to a good solution [72]. It does not require any gradient information of the function to be optimized and uses only primitive mathematical operators. As compared with other optimization methods, it is faster, cheaper and more efficient. In addition, there are few parameters to adjust in PSO. That is the reason PSO is an ideal optimization problem solver in optimization problems. PSO is well suited to solve the nonlinear, non-convex, continuous, discrete, integer variable type problems.

Kennedy and Eberhart first established a solution to the complex non-linear optimization problem by imitating the behaviour of bird flocks. They generated the concept of optimization function by means of a particle swarm [74]. Consider the global optimum of a n-dimensional function defined by

\[ f(x_1, x_2, x_3, \ldots, x_n) = f(X) \tag{4.2} \]

where \( x_i \) is the search variable, which represents the set of free variables of the given function. The aim is to find a value \( x^* \) such that the function \( f(x^*) \) is either a maximum or a minimum in the search space.

Consider the function given by

\[ f_1 = x_1^2 + x_2^2 \tag{4.3} \]

and \[ f_2 = x_1 \sin (4\pi x_2) - x_2 \sin (4\pi x_1 + \pi) + 1 \tag{4.4} \]

![Unimodel and Multimodel](image)

(a) Unimodel   (b) Multimodel

Figure 4.1 Plot of the function \( f_1 \) and \( f_2 \).
From the figure 4.1 (a) the ‘unimodel’ function has only one global minimum of the function \( f_i \) is at \((x_1, x_2) = (0, 0)\). i.e. at the origin of function \( f_i \) in the search space. Nonetheless, to find the global optimum is complicated for ‘multimodel’ functions, which has multiple local minima. Figure 4.1. (b) demonstrates the function \( f_2 \) which has a rough search space with multiple peaks, so many agents have to start from different initial locations and continue exploring the search space until at least one agent reach the global optimal position. All agents can communicate and share their information among themselves while this process.

The PSO algorithm is a multi-agent parallel search technique which maintains a swarm of particle \( S \) and each particle represents a potential solution in the swarm. All particles fly through a multidimensional search space where each particle is adjusting its position according to its own experience and that of neighbours. Suppose \( x_i^{t} \) denote the position vector of particle \( i \) in the multidimensional search space (i.e \( S \)) at time \( t \), then the position of each particle is updated in the search space by

\[
x_i^{t+1} = x_i^{t} + v_i^{t+1} \text{ with } x_i^0 \sim U(x_{\text{min}}, x_{\text{max}}) \\
\ldots (4.5)
\]

where, \( v_i^{t} \) is the velocity vector of particle that drives the optimization process and reflects both the own experience, knowledge and the social experience, knowledge from the all particles;

\( U(x_{\text{min}}, x_{\text{max}}) \) is the uniform distribution where \( x_{\text{min}} \) and \( x_{\text{max}} \) are its minimum and maximum values respectively.

In this manner, in a PSO strategy, all particles are initiated randomly and evaluated to compute fitness together with finding the personal best and global best.

Basically, two PSO algorithms, namely the Global Best (gbest) and Local Best (lbest) PSO, have been developed which differ in the size of their neighbourhoods. These algorithms are discussed in the following sections.
4.2.1. Global Best PSO

The global best PSO (gbest PSO) is a strategy where the position of each particle is influenced by the best-fit particle in the entire swarm. It utilizes a star social network topology where the social information obtained from all particles in the entire swarm. For the star topology one particle is selected as a ‘hub’ which is connected to all other particles in the swarm. However, all the other particles are only connected to the hub. In gbest PSO method, each individual particle, \( i \in \{1,2,...,n\} \) where \( n > 1 \), has a current position in search space \( x_i \), a current velocity, \( v_i \), and a personal best position in search space where particle \( i \) had the smallest value as determined by the objective function \( f \), considering a minimization problem. In addition, the position yielding the lowest value amongst all the personal best \( P_{best,i} \) is called the global best position which is denoted by \( G_{best} \). The following equations (4.6) and (4.7) define the personal and global best values are updated respectively.

Considering minimization problems, then the personal best position \( P_{best,i} \) at the next time step, \( t + 1 \) where \( t \in [0,...,N] \), is calculated as,

\[
P_{best,i}^{t+1} = \begin{cases} 
    P_{best,i} & \text{if } f(x_i^{t+1}) > P_{best,i} \\
    x_i^{t+1} & \text{if } f(x_i^{t+1}) \leq P_{best,i}
\end{cases} \quad \text{.........(4.6)}
\]

where \( f: S \rightarrow \mathbb{R} \) is the fitness function.

The global best position \( G_{best} \) at time step \( t \) is calculated as

\[
G_{best} = \min \{ P_{best,i} \} , \text{ where } i \in \{1,2,...,n\} \text{ and } n > 1 \quad \text{.........(4.7)}
\]

In this manner, to be noted that, the personal best \( P_{best,i} \) is the best position that the individual particle has visited since the first time step. On the other hand, the global best position \( G_{best} \) is the best position discovered by any of the particles in the entire swarm. The following flowchart 4.1 portrays the gbest PSO algorithm.
Fig. 4.2 Flow of Gbest PSO
For *gbest* PSO method, the velocity of particle *i* is calculated by

\[ v_{ij}^{t+1} = v_{ij}^t + c_1 r_{1j}^t [P_{best,j}^t - x_{ij}^t] + c_2 r_{2j}^t [G_{best,j}^t - x_{ij}^t] \] …… (4.8)

where, \( v_{ij}^t \) is the velocity of particle *i* in dimension *j* at time *t*;

\( x_{ij}^t \) is the position vector of particle *i* in dimension *j* at time *t*;

\( P_{best,j}^t \) is the personal best position of particle *i* in dimension *j* found from initialization through time *t*;

\( G_{best,j}^t \) is the global best position of particle *i* in dimension *j* found from initialization through time *t*;

\( r_{1j}^t \) and \( r_{2j}^t \) are random numbers from uniform distribution \( U(0,1) \) at time *t*.

\( C_1 \) and \( C_2 \) is the positive acceleration constants which are used to level the contribution of the cognitive and social components respectively;

### 4.2.2. Local Best PSO

The local best PSO strategy just allows each particle to be influenced by the best-fit particle chosen from its neighbourhood, and it reflects a ring social topology. This social information exchanged within the neighbourhood of the particle, denoting local knowledge of the environment. For this situation, the velocity of particle is calculated by

\[ v_{ij}^{t+1} = v_{ij}^t + c_1 r_{1j}^t [P_{best,j}^t - x_{ij}^t] + c_2 r_{2j}^t [L_{best,j}^t - x_{ij}^t] \] …… (4.9)

where \( L_{best,j}^t \) is the best position that any particle has had in the neighbourhood of particle found from initialization through time *t*.

Thus, the *gbest* PSO algorithm has every particle obtains the information from the best particle in the entire swarm, whereas in the *lbest* PSO algorithm, each particle obtains the information from only its immediate neighbourhood in the swarm.

The following Flowchart 2 summarizes the *lbest* PSO algorithm:
Fig. 4.3 Flow of Lbest PSO
4.3. PSO - Geometrical Illustration

The update velocity for particles consist of three components in equations (4.8) and (4.9) respectively. Consider a movement of a single particle in a two dimensional search space.

Figure 4.4 velocity and position update for a particle in a 2-dimensional search space.

Figure 4.4 demonstrates the three velocity components contribute to move the particle towards the global best position at time steps t and t+1 respectively.

Figure 4.5. Velocity and position update for Multi-particle in gbest PSO

The figure 4.5 demonstrates the position updates for more than one particle in a two dimensional search space and this figure portrays the gbest PSO. The optimum position is
denoted by the symbol ‘*’. Figure 4.5 (a) depicts the initial position of all particles with the global best position. The cognitive component is zero at $t = 0$ and all particles are only attracted towards the best position by the social components. The global best position does not change. Figure 4.5 (b) depicts the new position of all particles with the global best position after the first iteration i.e. at $t = 1$.

![Figure 4.6 Velocity and position update for multi-particle in lbest PSO.](image)

Figure 4.6 demonstrates all particles are attracted by their immediate neighbour in the search space using lbest PSO and there are some subsets of particles where one subset of particles is defined for each particle from which the local best particle is then selected. Figure 4.6 (a) portrays particles $a$, $b$ and $c$ move towards particle $d$, which is the best position in subset 1. In subset 2, particles $e$ and $f$ move towards particle $g$. Similarly, particle $h$ moves towards particle $i$, so does $j$ in subset 3 at time step $t = 0$. Figure 4.6 (b) for time step $t = 1$, the particle $d$ is the best position for subset 1 so the particles $a$, $b$ and $c$ move towards $d$.

### 4.4. PSO Algorithm Parameters

PSO has a few parameters that may influence its performance. For any given optimization problem, some of these parameter values and choices have a large impact on the
efficiency of the PSO strategy, and other parameters have small or no effects. The basic parameters are swarm size (number of particle), number of iterations, velocity components, and acceleration coefficients. Furthermore, PSO, is also influenced by inertia weight, velocity clamping and velocity constriction.

### 4.4.1. Velocity Components

The velocity components are very important for updating particle’s velocity. There are three terms of the particle’s velocity in the equation (4.8) and (4.9).

1. The term \( v_{ij}^t \) is called inertia component that provides a memory of the previous flight direction that implies, movement in the immediate past. This component represents as a momentum which prevents to drastically change the direction of the particles and to bias towards the current direction.

2. The term \( c_{1}r_{1j}^t [P_{best,i}^t-x_{ij}^t] \) is called cognitive component which measures the performance of the particles relative to past performances. This component looks like an individual memory of the position that was the best for the particle. The effect of the cognitive component represents the tendency of individuals to return to positions that satisfied them most in the past. The cognitive component referred to as the nostalgia of the particle.

3. The term \( c_{2}r_{2j}^t [G_{best}-x_{ij}^t] \) for \textit{gbest} PSO or \( c_{2}r_{2j}^t [L_{best,i}^t-x_{ij}^t] \) for \textit{lbest} PSO, is called social component which measures the performance of the particle \textit{i} relative to a group of particles or neighbours. The social component’s effect is that each particle flies towards the best position find out by the particle’s neighbourhood.

### 4.4.2. Acceleration Coefficients

The acceleration coefficients \( C_1 \) and \( C_2 \), together with the random values \( r_1 \) and \( r_2 \), maintain the stochastic influence of the cognitive and social components of the particle’s velocity.
respectively. The constant $C_1$ expresses the amount of certainty a particle has in itself, while $C_2$ expresses the amount of confidence a particle has in its neighbours [4]. There are some properties of $C_1$ and $C_2$:

- When $C_1 = C_2 = 0$, then all particles continue flying at their current speed until they hit the search space’s boundary. Therefore, from the equations (4.8) and (4.9), the velocity update equation is calculated as

$$v_{ij}^{t+1} = v_{ij}^t \quad \text{……. (4.10)}$$

When $C_1 > 0$ and $C_2 = 0$, all particles are independent. The velocity update equation will be

$$v_{ij}^{t+1} = v_{ij}^t + c_1 r_{ij}^t [P_{best,i}^t - x_{ij}^t] \quad \text{……. (4.11)}$$

On the other hand, when $C_2 > 0$ and $C_1 = 0$, all particles are attracted to a single point (i.e. Gbest) in the entire swarm and the update velocity will become

$$v_{ij}^{t+1} = v_{ij}^t + c_2 r_{ij}^t [G_{best}^t - x_{ij}^t] \quad \text{for gbest PSO} \quad \text{………. (4.12)}$$

$$v_{ij}^{t+1} = v_{ij}^t + c_2 r_{ij}^t [L_{best,ij}^t - x_{ij}^t] \quad \text{for lbest PSO} \quad \text{………. (4.13)}$$

When $C_1 = C_2$, all particles are attracted towards the average of $P_{best,i}^t$ and $G_{best}$.

When $C_1 \gg C_2$, each particle is more strongly influenced by its personal best position, resulting in excessive wandering. On the other hand, when $C_2 \gg C_1$ then all particles are much more influenced by the global best position, which causes all particles to run prematurely to the optima.

Apparently, $C_1$ and $C_2$ are static with their optimized values being found empirically. Wrong initialization of $C_1$ and $C_2$ may result in divergent or cyclic behaviour. From the different empirical researches, it has been proposed that the two acceleration constants should be $C_1 = 2$.
4.5. Inertia Weight

There was a weakness inherent in velocity update equations (4.8) and (4.5) that was fixed by the introduction of an inertia weight. The inertia weight, denoted by ‘\(w\)’, is considered to replace by adjusting the influence of the previous velocities in the process, i.e., it controls the momentum of the particle by weighting the contribution of the previous velocity. The inertia weight ‘\(w\)’ will at every step be multiplied by the velocity at the previous time step, i.e., in this manner, in the gbest PSO, the velocity equation of the particle with the inertia weight changes from equation (4.8) to

\[
 v_{ji}^{t+1} = w \times v_{ji}^{t} + c_1 r_1 \left[ P_{best,j}^{t} - x_{ji}^{t} \right] + c_2 r_2 \left[ G_{best}^{t} - x_{ji}^{t} \right] 
\]

\[
 ...........(4.14)
\]

In the lbest PSO, the velocity equation changes in a similar way as the above velocity equation do.

The inertia weight was initially introduced by Shi and Eberhart in 1999 to reduce the velocities over time, to control the exploration and exploitation abilities of the swarm, and to converges the swarm more accurately and efficiently.

The inertia weight technique is very useful to ensure convergence. However, there is a disadvantage of this method is that once the inertia weight is decreased, it cannot increase if the swarm needs to search new areas. This method is unable to recover its exploration mode.

A PSO algorithm includes particle initialization, parameter selection, iteration terms, function evaluation, and stopping condition. The first step of the PSO is, to initialize the swarm and control the parameters, the second step is, calculates the fitness function and define the iteration numbers, and the last step is, to satisfy stopping condition. In general, large velocity has large momentum and consequently large position update. In this manner, such large initial position updates can cause particles to move away from boundaries in the feasible region, and the algorithm needs to take more iteration before settling the best
solution. The performance of each particle is measured using a fitness function. i.e how close the particle is from the global optimum.

4.6. The fitness function

Mathematically, an optimization problem has a fitness function; describing the problem under a set of constraints which represents the solution space for the problem. A PSO algorithm needs a fitness function. This fitness function is needed by each particle to determine its own best position so far, and the best position found so far by any particle in its neighbourhood. A fitness function for a PSO algorithm concerned with clustering has to be based on an appropriate cluster validity index. The smaller index value indicates the better solution. Fitness value is one of the parameters to measure the accuracy of the result for PSO.

The PSO algorithm is an iterative optimization process and repeated iterations will continue until a stopping condition is satisfied. Within one iteration, a particle determines the personal best position, the local or global best position, adjusts the velocity, and a number of function evaluations are performed. Function evaluation implies one calculation of the fitness or objective function which computes the optimality of a solution. If $n$ is the total number of particles in the swarm, then $n$ function evaluations are performed for each iteration.

In the basic PSO, one of the major problems is the lack of diversity when particles start to converge to the same point. To prevent this problem, several methods have been developed to continually inject randomness, or chaos, into the swarm. It is important to remember that continual injection of random positions will cause the swarm never to reach an equilibrium state. Kennedy and Eberhart first introduced the advantages of randomly reinitializing particles and referred to as craziness.
4.7. Particle Swarm Clustering (PSC)

A clustering method that is based on PSO is named as Particle swarm clustering (PSC). The algorithm [68] finds the centroid of a user specified number of clusters, where each cluster group together similar patterns.

With regards to clustering, a single particle represents the $N_c$ cluster centroid vectors. That is, each particle $x_i$, is constructed as follows,

$$X_i = (m_{i1}, \ldots, m_{ij}, \ldots, m_{iN_c})$$ .................(4.15)

where $m_{ij}$ refers to the $j^{th}$ cluster centroid vector of the $i^{th}$ particle in cluster $C_{ij}$. In this way, a swarm represents a number of candidates clustering for the current data vectors. The fitness of particle is easily measured as a quantization error,

$$J_e = \frac{\sum_{j=1}^{N_c} \sum_{z_p \in C_{ij}} d(z_p, m_j)}{|C_{ij}| \cdot N_c}$$ .................(4.16)

where $d(z_p, m_j) = \sqrt{\sum_{k=1}^{Nd}(z_{pk} - m_{jk})^2}$, .................(4.17)

where $k$ subscripts the dimension.

$|C_{ij}|$ is the number of data vectors belonging to cluster $C_{ij}$, i.e the frequency of that cluster.

Using the standard gbest PSO, data vectors can be clustered as follow:

1. Initialize each particle to contain $N_c$ randomly selected cluster centroids
2. For $it = 1$ to $it_{max}$ do
(a) For each particle $i$ do
(b) For each data vector $z_p$
   i. Calculate the Euclidean distance $d(z_p, m_{ij}) \ orall \ 	ext{cluster centroid } C_{ij}$
   ii. Assign $z_p$ to cluster $C_{ij}$ such that $d(z_p, m_{ij}) = \min_{\forall C_{ij}} \{d(z_p, m_{ij})\}$
   iii. Calculate the fitness using equation (4.16)
(c) Update the global best and local best position

(d) Update the cluster centroids using equations (4.14) and (4.5).

where \(it_{\text{max}}\) is the maximum number of iterations.

### 4.7.1. Differences between Particle Swarm Clustering and PSO

The main structural differences between the PSC and PSO algorithms as follows:

(i) In PSC, the particles altogether compose a solution to the data clustering problem.

(ii) The PSC does not use an explicit cost function to evaluate the quality of the particles. Instead, the Euclidean distance is used as a measure to assess the dissimilarity between a particle and an object, and particles move around the space in order to represent statistical regularities of the input data.

(iii) A self-organizing term, which moves the particle toward the input object, was added to the velocity equation.

(iv) In the PSO algorithm, all the particles in the swarm are updated in each iteration. In the PSC, the particles to be updated are defined by input objects, i.e. only the winner - the one closest to the input datum is updated according to equation (4.8) and equation (4.5).

### 4.8. The Algorithm of Enhanced Estimation Of Centroid (EEOC)

In this work, a lightweight clustering algorithm [14] called Enhanced Estimation Of Centroid (EEOC) is proposed. It is commonly analogous to a concept of individual decision making according to expected utility.

This work suggested that a reduction in the frequency of a Distance Matrix (DM) restore would significantly reduce the computation time. It is a sensible approach to diminish the frequency of movement update.
In order to reduce the time complexity, in this work, a modified version of the centroid estimation known as the Enhanced Estimation of Centroid is used. In the modified version of the PSC (EEOC), only winning particles are found out for the consecutive iterations. Because of this, the time complexity is reduced to one fourth. For finding out the winning particle, the following change is carried out in PSC algorithm,

$$f_{win}(m) = f(m) \in \min (e (a_i(m) - f_i(m)))$$

for each particle \( f \)

Obtain the elements of the centroid cluster

$$g_i^{cluster} = \Pi g f_i(m)$$

// \( g \) – centroid of the cluster \( \in \) position of the cluster

$$C_i = \text{size} (g_i^{cluster})$$

Calculate the updated position

According to the expected utility hypothesis, decisions are made based on the evaluation of the utility and risk of every combination of available choices. In every decision made there are consequences which will affect not only the decision maker but also the neighbourhood to which the decision maker interacts. The decision maker will also have a preference towards a specific choice which is subjectively less risky.

In EEOC, the term position \( f \) is also analogous to the decision making behaviour of an individual and the influence of the environment that causes the individual to make a decision to change their behaviour. Inspired by the logical paradigm presented in this proposal, the restoration scheme is incorporated in this mindset.

The restoring rule of the EEOC can be summarized as follows.

The particle velocity is bounded by \( Max_v \).
\( f_i(t) \) denotes the position and the best position of a particle \( i \) in relation to the input pattern \( j \).

\( c_j(t) \) represents the position of a particle that has been closest to the input pattern \( j \).

For each iteration, each particle position is restored only once.

This arrives after all possible data points which are closer to the particle have been individually considered by the particle. This means position update occurs only \( i \) times per iteration, where \( i \) is the number of particles in the set. The Distance Matrix (DM) and best positions are updated after all particle positions are updated. This means the Distance Matrix (DM) is restored only once per iteration. The overall minimum computation is defined to store the best position combination.

\( f_i(m) \) and \( a_i(m) \) denote position and best position of a particle \( i \) in relation to the input pattern \( j \).

\( c_i(m) \) represents the position of a particle that has been closest to the input pattern \( j \).

\( S(m) \) represents the best position combination that has achieved an overall minimum according to a given fitness function \( f_1(m) \).

\( \delta x \) is defined by using the following equations (4.20) and (4.21).

\[
\text{Equ 4.14 } \Rightarrow \delta f_i(m+1) = w(t) \delta x_i(m) \quad \ldots \ldots \text{(4.20)}
\]

\[
\text{Equ 4.5 } \Rightarrow f_i(m+1) = f_i(m) + \delta f_i(m+1) \quad \ldots \ldots \text{(4.21)}
\]

where \( w(m) \) is the inertia weight.

**Algorithm:**

\[
P = \text{EEOC} \left( \text{data\_matrix}, \text{iter\_func\_max}, \text{max}_p, p_c, \phi \right)
\]

Initialize the number of particles as \( p_c \)

Calculate the distances of each particle’s position \( a \) and the input pattern of the particle’s position as \( c \)
Update the distance and velocity until $t < \text{iter\_func}_{\text{max}}$

Find the closest particle $(i, j)$

\[
DM = e(g_j, f_i): II i, j \quad \text{//DM – distance matrix}
\]

\[
J = \min \{e(g_j, f_i) : i \in \{1, 2, ..., p_c\}\}
\]

// Fitness function parameter $e(g_j, f_i)$ is calculated as above.

// Restore the special and overall best of the particle

// The fitness function / objective function can be calculated such as

\[
a_{DM}(m + 1) = \begin{cases} f_j(m) & \text{if } e(g_j, f_j(m)) < e(g_j, a_j(m)) \\ a_j(m) & \text{otherwise} \end{cases}
\]

// $f_j(m)$ denotes the position.

// $a_j(m)$ denotes the best position

// $e(g_j, f_j(m))$ – Previous position

// $e(g_j, a_j(m))$ – Current position

\[
c_j(m + 1) = \begin{cases} f_j(m) & \text{if } e(g_j, f_j(m)) < e(g_j, a_j(m)) \\ c_j(m) & \text{otherwise} \end{cases}
\]

// $C_j(m)$ represents the position of the particle that has closest to the input pattern.

// For each particle distance is calculated based on the distance, it’s calculate the fitness function. If the current fitness value is better than the previous one current value take as a best fitness. This process goes till the number of iterations is finished. //

Restore the velocity and the position

\[
m = m + 1
\]

// normally, the inertia weight, $\delta$ value is 0.4 to 0.9 randomly.

// Here it is 0.9

\[
f_i(m + 1) = f_i(m) + \delta f_i(m)
\]

Find the closest data point for each particle based on minimum distance
\[ DMf^{\text{min}}(m) \quad Jf^{\text{min}} = \min (DM, i) \]

Find the closest particle for each data point based on minimum distance

\[ DMg^{\text{min}}(m) \quad Jg^{\text{min}} = \min (DM, j) \]

Update \( a_i(m), c_i(m) \)

\( a_i(m) \) – best position,

\( c_j(m+1) \) – closest to the input pattern

\[
\begin{align*}
  a_i(m + 1): \Pi i = & \left\{ f_jx^{\text{min}}(m) \text{ if } DMf^{\text{min}}_i(m) < DMf^{\text{min}}_{i-1}(m) \right. \\
  & a_i(m) \quad \text{otherwise} \\
  \end{align*}
\]

\[
\begin{align*}
  c_j(m + 1): \Pi j = & \left\{ f_jx^{\text{min}}(m) \text{ if } DMf^{\text{min}}_j(m) < DMf^{\text{min}}_{j-1}(m) \right. \\
  & a_j(m) \quad \text{otherwise} \\
  \end{align*}
\]

// If the condition satisfies the criteria, no change in the cluster centroid, otherwise

\( a_j(m) \) is updated. //

\[
S(m + 1) = \left\{ f_i(t): \Pi i \text{ if } f(t (f_i(m)) \text{ if } f(t(S(m)) \text{ otherwise}} \right. \\
\]

// \( S(m) \) – represent the best position combination that has an achieved overall
minimum according to a given fitness function //

// \( S(m) \) – act as a global best

Find the winning particle

\[ f_{\text{win}}(m) = f(m) \in \min (e (a_i(m) - f_i(m))): \Pi i \]

for each particle \( f \)

Obtain the elements of the centroid cluster

\[ g_i^{\text{cluster}} = \Pi g f_i(m) \]

// \( g \) – centroid of the cluster \( e \) position of the cluster

\[ C_i = \text{size} (g_i^{\text{cluster}}) \]

Calculate the updated position

end
\[ m = m + 1 \]

The brief explanation of the proposed algorithm is illustrated as above. The short form the above algorithm is rewrite as follows.

Algorithm (EEOC):

Step 1: Initialize the number of particles

Step 2: Calculate the distance of each particle position

Step 3: Update the distance and velocity

Step 4: Find the closest particle

Step 5: Restore the special and overall best of the particle

Step 6: Restore the velocity and the position

Step 7: Find the closest data point for each particle

Step 8: Find the closest particle for each data point

Step 9: Update the special and overall best of the particle

Step 10: Find the winning particle

Step 11: Obtain the element of the centroid cluster

Step 11: Calculate the updated position

Step 12: Repeat from step 2 till converge the result.

Step 13: Stop

This EEOC is the modification of PSC. The modification done in PSC is, the Distance Matrix (DM) and best positions are updated after all particle positions are updated. This means the Distance Matrix (DM) is restored only once per iteration. The overall minimum computation is defined to store the best position combination. This EEOC is used in intra band part; that is segmented the HSI. In the inter band part; bands are reduced by applying any one of unsupervised clustering method. This reduced band is given as input to intra part;
for further processing i.e segmentation. The segmentation is carried out on the reduced band by EEOC. It may greatly quickened restoration time. It may significantly reduce the time complexity and efficiency of each position update. The upcoming chapters, namely Chapter V, VI, and VII are using this algorithm as for segmentation of reduced band in intra band cluster part.