CHAPTER 4

PARTICLE SWARM OPTIMIZATION

4.1 INTRODUCTION

Particles Swarm Optimization (PSO) is an evolutionary computation technique originally developed by Kennedy and Eberhart (1995). The PSO is motivated from the stimulation of social behavior instead of evolution of nature as in the other evolutionary algorithms (genetic algorithms, evolutionary programming, evolutionary strategies, and genetic programming). PSO is sociologically inspired, since the algorithm is based on sociological behavior associated with bird flocking. It is a population based evolutionary algorithm. Similar to the other population based evolutionary algorithms, PSO is initialized with a population of random solutions.

The algorithm maintains a population of particles, where each particle represents a potential solution to an optimization problem Unlike the most of the evolutionary algorithms, each potential solution (individual) in PSO is also associated with a randomized velocity, and the potential solutions called particles, are then flown through the problem space. Let S be the size of the swarm, each particle i can be represented as an object with several characteristics. A population of particles is initialized with random position X_i and velocities V_i objective function F_i is evaluated using the particles positional coordinates as input values. Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far. This value is called p_{best}. Another best value that is tracked by the global version of the swarm is the overall best value, and its location obtained so far by any particle in the population. This location is called g_{best}. At each time step velocity of each particle flying toward its g_{best} and p_{best} location is changed. Acceleration is weighted by random terms, with separate random numbers being generated for acceleration towards p_{best} and g_{best} location.

At each time step position and velocities are adjusted and the function is evaluated with new coordinates. When the particle discovers a pattern that is better than any it has found previously, it stores the coordinates in the vector $p_{best\ id.}$ The difference between the best point found by a particular agent and the individual's current positions is stochastically added to the current velocity causing the trajectory to oscillate around the point. Further each particle is defined within the context of a topological neighborhood comprising itself and some other particles in the population. The stochastically weighted difference between the neighborhood's best position g_{bestd} and the individual's current position is also added to its velocity, adjusting it for the next time step. These adjustments to the particle's movement through the space cause it to search around the two best positions.

- **Particle (X)**: It is the candidate solution represented by a d dimensional vector, where d is the number of optimized parameters. At time t, the i^{th} particle $X_i(t)$ can be described as $X_i(t)=[\ X_{i1}\ (t), X_{i2}(t)\ldots\ldots..\ X_{id}(t)]$, where X's are the optimized parameters and X_{id} (t) is the position of i^{th} particle with respect to the d^{th} dimension; i.e. the value of the d^{th} optimized parameter in the i^{th} candidate solution.

- **Population X(t)**: It is a set of n particle at time t, i.e. $X(t)=[\ X_1(t),X_2(t)\ \ldots..\ \ X_n(t)]$.

- **Swarm**: It is an apparently disorganized population of moving particles that tend to cluster together while each particle seems to be moving in a random direction.

- **Particle velocity V(t)**: It is the velocity of the moving particles represented by an d- dimensional vector. At time t, the ith particle velocity $V_i(t)$ can be described as $V_i(t)= [V_{i1} (t), V_{i2}(t) \ldots\ldots V_{id}(t)]$, where V_{id} is the velocity of i^{th} particle with respect to the d^{th} dimension. The velocity update step is specified separately for each dimension $d \in 1 \ldots n$, so that V_{id} denotes the d^{th} dimension of the velocity vector associated with the i^{th} particle. The velocity update is then given by equation (4.1)

$$\begin{aligned} V_{id}^{(t+1)} = {} & w\, V_{id}^{(t)} + c_1 \ \text{rand}_1 \, (p_{\text{best id}}^{(t)} - X_{id}^{(t)}) \\ & + c_2 \ \text{rand}_2 \, (g_{\text{best d}}^{(t)} - X_{id}^{(t)}) \end{aligned} \tag{4.1}$$

  where $V_{id}^{(t)}$ and $X_{id}^{(t)}$ are the velocity and position of particle i, in d dimensional space respectively. $p_{\text{best id}}^{(t)}$ best position of individual i in d dimensional space until generation t; $g_{\text{best d}}^{(t)}$ is the best position of the group in d dimension until generation t; w is the inertia weight factor controlling the dynamics of flying; c_1 cognitive parameter and c_2 social parameter; rand_1 and rand_2 are random variables in the range [0,1].

From the definition of the velocity in the equation (4.1) it is clear that c_2 regulates the maximum step service in the direction of the global best particle, and c_1 regularizes the step size in the direction of the personal best position of the particle. The value of V_{id} is clamped to the range $[-V_{imax}, V_{imax}]$ to reduce the likelihood that the particle might leave the search space. The position of each particle is updated using the new velocity vector for that particle, so that

$$X_{id}^{(t+1)} = X_{id}^{(t)} + V_{id}^{(t+1)} \tag{4.2}$$

- **Inertia Weight (w):** The inertia weight controls the exploration and exploitation of the search space because it dynamically adjusts velocity. The inertia weight is employed to control the effect of the previous velocities on the current velocity. This makes compromise between a global and (wide ranging) and local (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration (searching new areas) while a small one tend to facilitate local exploration. A properly chosen inertia weight can provide balance between the global and local exploration of the swarm, which leads to a better solution. It is better to initially set the inertia weight to a large value in order to make better global exploration of the search space and gradually decrease it to get more refined solution. A linearly decreasing inertia weight changes the search from global to local linearly.

  Many problems require the search algorithm to have nonlinear search ability. By deriving some statistical features from the obtained result, in each iteration will help to understand the PSO search and calculate the proper inertia weight for next iteration. In this work the inertia weight w will decrease when the number of generation increases. It decreases linearly during the optimization run according to

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times iter \tag{4.3}$$

where w_{max} is the maximum value of inertia weight and w_{min} is the minimum value, iter is the current iteration and $iter_{max}$ is the maximum number of iterations.

- **Cognitive parameter (c_1) and social parameter (c_2)**: In the PSO algorithm each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. The fitness value is stored and called as particle best $p_{best.}$ Another best version tracked by the global version of PSO is the overall best value and it is location obtained so far by any particle in swarm (g_{best}). PSO at each step consist of changing the velocity of each particle (accelerating) towards its p_{best} and g_{best} locations. The accelerating constant c_1 and c_2 represent the weighting of the stochastic acceleration term that pulls each particle towards p_{best} and g_{best} positions. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement towards target regions.

  By trail and error it is found that acceleration constant c_1 and c_2 equal to 2.0 gives good results but it is not a usual rule. This value gives fast global convergence to the optimum solution for most of the problem. Increase in the value did not have much effect in the convergence rate. Local minima are avoided by small local neighborhood but faster convergence is obtained by larger global neighborhood and in general global neighborhood is preferred. $c_{1,}$ $c_{2,}$ are acceleration constants which change the velocity of a particle towards $p^{(t)}_{best\,id}$ and $g^{(t)}_{best\,d}$ (generally somewhere between $p^{(t)}_{best\,id}$ and $g^{(t)}_{best\,d}$). Thus adjustment of these constants changes the amount of tension in the system.

- **Personal best**: The personal best position associated with the particle i is the best position that the particle has visited (a previous value of X_i), yielding the highest fitness value for that particle. For a minimization task, a position yielding the smaller function value is regarded as having fitness. The symbol f(X) will be used to denote the objective function that is being minimized. The update equation is

$$p_{\text{best id}}^{(t+1)} = \begin{cases} X_{id}^{(t)} \text{ if } & f(X_{id}^{(t+1)}) \geq f\left(p_{\text{best id}}^{(t)}\right) \\ X_{id}^{(t+1)} \text{ if } & f(X_{id}^{(t+1)}) < f\left(p_{\text{best id}}^{(t)}\right) \end{cases} \tag{4.4}$$

- **Global best (g_{best})**: The g_{best} offers a faster rate of convergence at the expense of robustness. This g_{best} maintains only a single best solution called the global best particle, across the entire particle in the swarm. This particle act like an attractor, pulling all the particles towards it. Eventually all particles will converge to this position, so if it is not updated regularly, the swarm may converge prematurely

4.2 GENERAL PSO ALGORITHM

PSO algorithm consists of the following steps:

Step 1: Initialization: Initialize a population of particles with random position and velocities in d dimensional problem space. Confine the search space by specifying the lower and upper limits of each decision variable. The populations of points are initialized with the velocity and position set to fall into the pre-specified or allowed range and satisfying the equality and inequality constraints.

- Evaluate the fitness of each particle in terms of pareto-dominance.
- Record the non dominated solutions found so far and save them in archive.
- Initialize the memory of each individual where the personal best position is stored.
- Choose the global best position $g^{(t)}_{best\,d}$ from the archive.
- Increase the generation number.

Step 2: Velocity updating: At each iteration, the velocities of all particles are updated according to the equation (4.1) which is:

$$V^{(t+1)}_{id} = w\,V_{id}^{(t)} + c_1\ rand_1\ (p^{(t)}_{best\,id} - X^{(t)}_{id}) + c_2\ rand_2\ (g^{(t)}_{best\ d} - X^{(t)}_{id})$$

where $V_{id}^{(t)}$ and $X_{id}^{(t)}$ are the velocity and position of particle i, in d dimensional space respectively. $p^{(t)}_{best\,id}$ best position of individual i in d dimensional space until generation t; $g^{(t)}_{best\,d}$ is the best position of the group in d dimension until generation t; w is the inertia weight factor controlling the dynamics of flying; c_1 and c_2 are accelerating constants; $rand_1$ and $rand_2$ are random variables in the range [0,1].

The first part of the equation (4.1) is the momentum part of the particle. The inertia weight w represents the degree of the momentum of the particles. The second part of the equation (4.1) is the cognition part which represents the independent thinking of the particle itself. The third part of equation (4.1) is the social part which represents the collaboration among the particles. V_{max} is an important parameter it determines the resolution with which the regions around the current solutions are searched. If V_{max} is too high the PSO facilitates global search, and particles might fly pass good solutions. If V_{max} is too small the PSO facilitates local search and particles may not explore sufficiently beyond locally good regions. They could be trapped at local minima unable to move far enough to reach a better position in the problem space. V_{max} is often set at about 10-20% of the dynamic range of the variable on each dimension.

The population size selected is problem dependent. Population size of 20-50 is most common. Under the multi –objective environment the number of non dominated solutions is directly linked to the population size. So a larger population is preferred

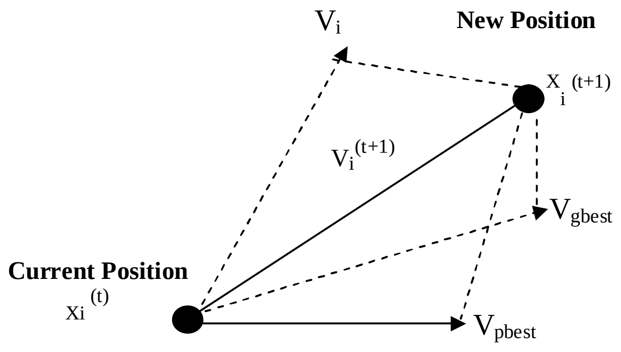

Figure 4.1 Concept of changing a particle position in PSO

Step 3: Position updating: Between successive iterations, the position of all particles are updated according to the equation (4.2) which is

$$X_{id}^{(t+1)} = X_{id}^{(t)} + V_{id}^{(t+1)}$$

where $X_{id}^{(t+1)}$ is the new position and $X_{id}^{(t)}$ is the previous position and $V_{id}^{(t+1)}$ is the new velocity.

Figure 4.1 indicates the concept of changing the position of a particle in PSO.

- Check all the imposed constraints to ensure the feasibility of all the potential solutions. If any element of individual violates its inequality constraints then the position of the individual is fixed to its maximum/minimum operating point as shown in equation (4.5)

$$X_{id}^{t+1} = \begin{cases} X_{id}^{t} + v_{id}^{t+1} & \text{if} \quad X_{id}^{min} \le X_{id}^{t} + v_{id}^{t+1} \le X_{id}^{max} \\ X_{id}^{min} & \text{if} \quad X_{id}^{t} + v_{id}^{t+1} < X_{id}^{min} \\ X_{id}^{max} & \text{if} \quad X_{id}^{t} + v_{id}^{t+1} > X_{id}^{max} \end{cases} \tag{4.5}$$

  where $X_{id}^{min}, X_{id}^{max}$ is the minimum and maximum position of the particle in the d dimensional space.

- Update the archive which stores non dominated solution.

Step 4: Memory updating: Update particle best position $p_{best\,id}^{(t)}$ and global best position $g_{best\,d}^{(t)}$ using equation (4.6).

$$\left.\begin{aligned} P_{best\,id}^{(t+1)} &\leftarrow X_{id}^{(t+1)} \text{ if } \quad f(X_{id}^{(t+1)}) < f\left(P_{best\,id}^{(t)}\right) \\ g_{best\,d}^{(t+1)} &\leftarrow X_{id}^{(t+1)} \text{ if } \quad f(X_{id}^{(t+1)}) < f\left(g_{best\,d}^{(t)}\right) \end{aligned}\right\} \tag{4.6}$$

where f(X) is the objective function to be minimized. Compare particles fitness evaluation with particles $p_{best\,id}^{(t)}$. If current value is better than $p_{best\,id}^{(t)}$ then set $p_{best\,id}^{(t+1)}$ value equal to the current value and the $p_{best\,id}^{(t+1)}$ location equal to the current location in d dimensional space. Compare fitness evaluation with the population’s overall previous best. If the current value is better than $g_{best\,d}^{(t)}$ then reset $g_{best\,d}^{(t+1)}$ to the current particles array index and value.

Step 5: Termination criteria examination: The algorithm repeats Step 2 to Step 4 until a sufficient good fitness or a maximum number of iterations/epochs are reached. Once terminated, the algorithm outputs the points of $g_{best\,d}^{(t)}$ and f($g_{best\,d}^{(t)}$) as its solution.

Optimal Parameters of PSO: For all the problems considered the following parameters give optimal results:

Population size	=	50
Number of iterations	=	100

$c_1, c_2 = 2$

w is varied from 1.4 to 0.4.

Maximum velocity V^{max} is limited to 10% of the dynamic range of the variables on each dimension.

The software is written in Matlab 6.5 platform and executed in Pentium IV, 3.0 GHz system.

4.3 PSO FLOW CHART

The flow chart explains the sequence of steps to be carried out in the PSO algorithm.

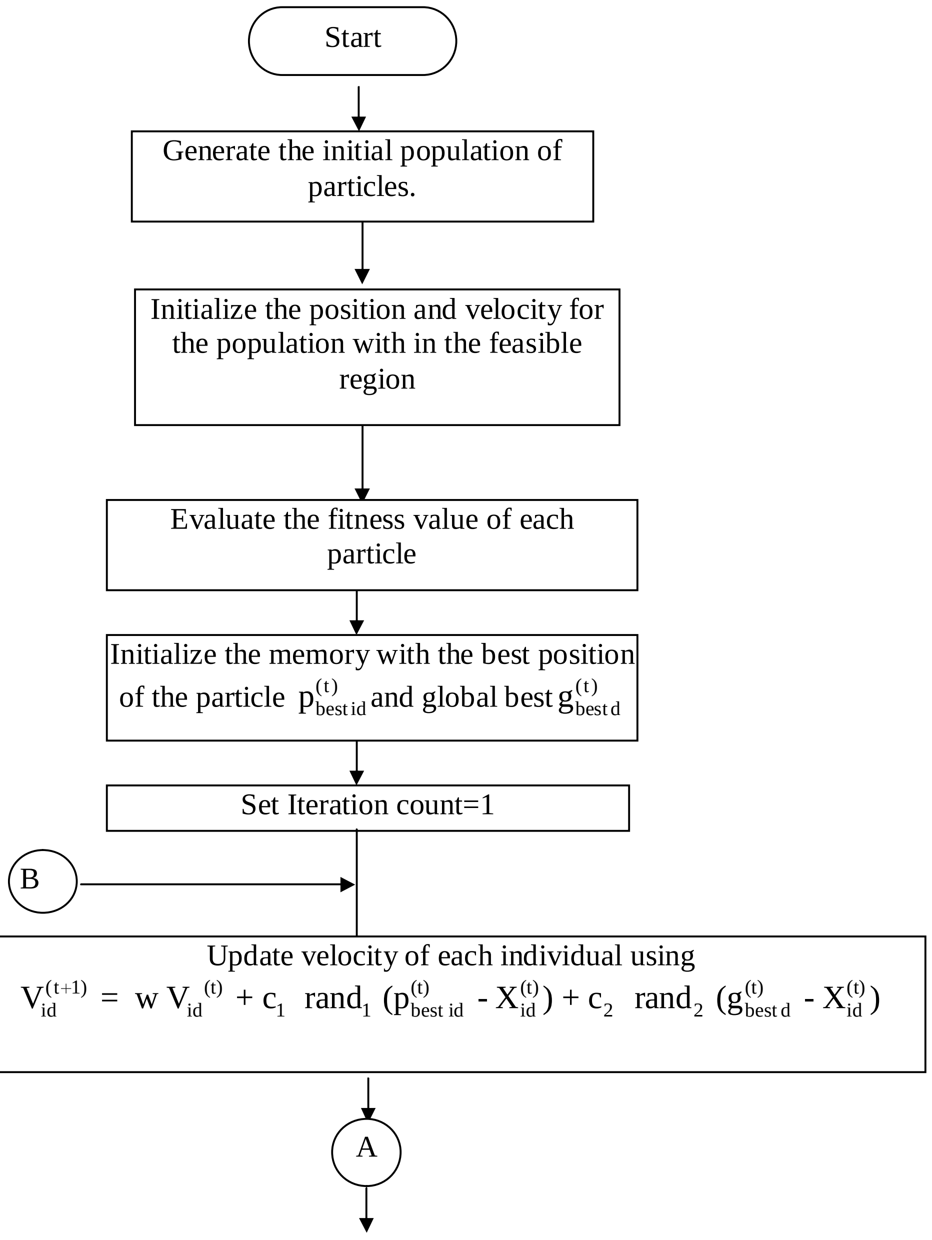
Start
Generate the initial population of particles.
Initialize the position and velocity for the population with in the feasible region
Evaluate the fitness value of each particle
Initialize the memory with the best position of the particle $p_{best\,id}^{(t)}$ and global best $g_{best\,d}^{(t)}$
Set Iteration count=1
B
Update velocity of each individual using
$V_{id}^{(t+1)} = w V_{id}^{(t)} + c_1 \ rand_1 \ (p_{best\,id}^{(t)} - X_{id}^{(t)}) + c_2 \ rand_2 \ (g_{best\,d}^{(t)} - X_{id}^{(t)})$
A

Figure 4.2 Flow chart for PSO algorithm

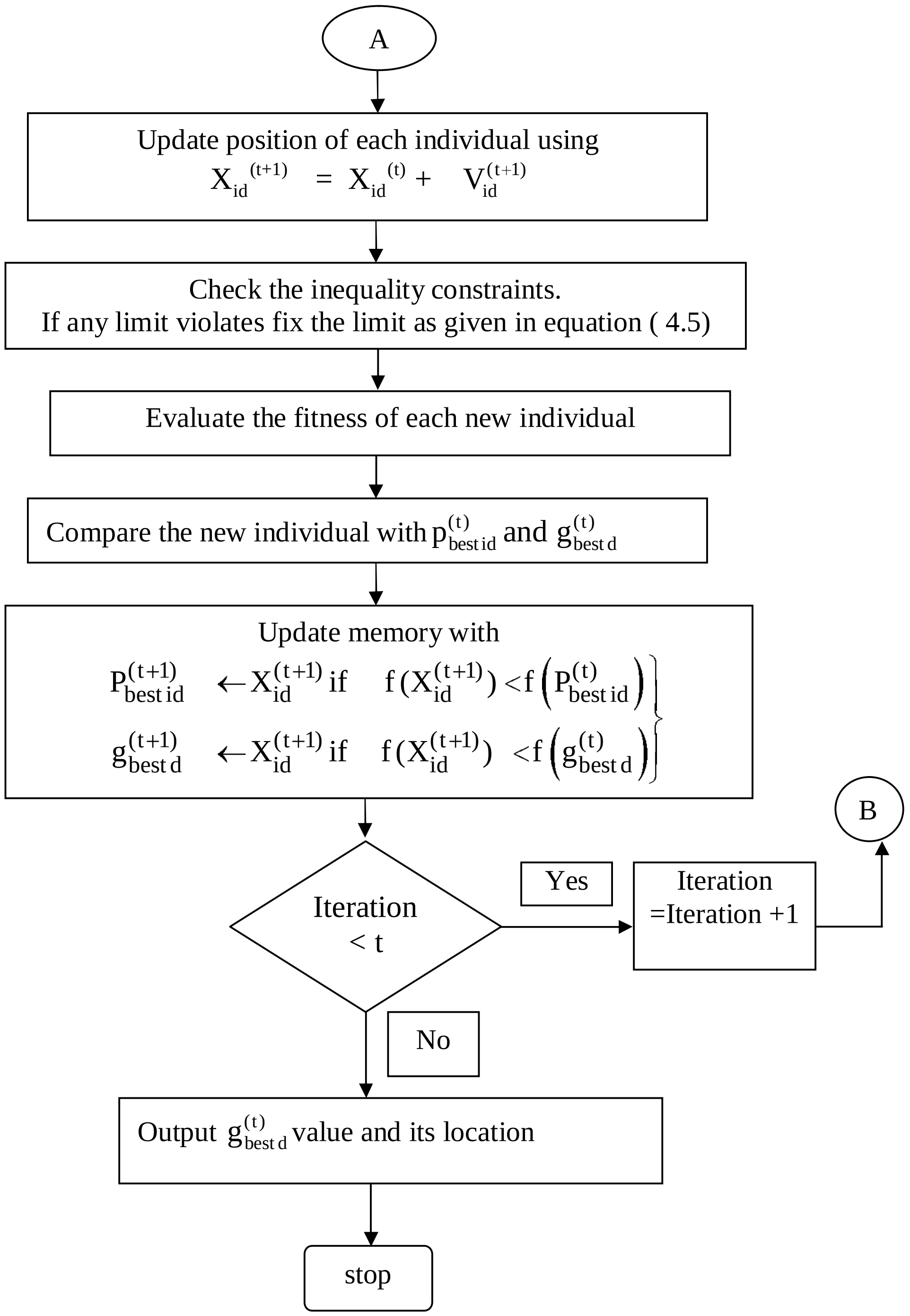
A
Update position of each individual using
$X_{id}^{(t+1)} = X_{id}^{(t)} + V_{id}^{(t+1)}$
Check the inequality constraints.
If any limit violates fix the limit as given in equation (4.5)
Evaluate the fitness of each new individual
Compare the new individual with $p_{best\,id}^{(t)}$ and $g_{best\,d}^{(t)}$
Update memory with
$P_{best\,id}^{(t+1)} \leftarrow X_{id}^{(t+1)}$ if $f(X_{id}^{(t+1)}) < f\left(P_{best\,id}^{(t)}\right)$
$g_{best\,d}^{(t+1)} \leftarrow X_{id}^{(t+1)}$ if $f(X_{id}^{(t+1)}) < f\left(g_{best\,d}^{(t)}\right)$
Iteration < t
Yes
Iteration =Iteration +1
B
No
Output $g_{best\,d}^{(t)}$ value and its location
stop

Figure 4.2 (Continued)

4.4 ADVANTAGES OF PSO

PSO has the following advantages:

i) It is used for discrete / continuous or discontinuous functions and variables. It is less sensitive to the nature of the objective function, i.e. convexity or continuity.

ii) It is a derivative free algorithm unlike many conventional techniques.

iii) It has less parameter to adjust unlike many other competing evolutionary techniques. There are no crossover and mutation operators as in GA and evolutionary programming.

iv) The function evaluation is by means of basic mathematic and logic operations. It is easy to implement.

v) It has the flexibility to be integrated with other optimization techniques to form a hybrid tool.

vi) It has the potential of efficient computation with very large numbers of concurrently operating processors. So it has the ability to escape local minima.

vii) It can handle objective functions with stochastic nature.

viii) It does not require a good initial solution to guarantee its convergence.

4.5 CONCLUSION

In this work stochastic model is formed using Є-constraint/ weighting method and the PSO algorithm is applied to solve the following power system optimization problems:

i) Stochastic thermal power dispatch-modeled using Є-constraint method.

ii) Stochastic thermal power dispatch-modeled using weighting method.

iii) Stochastic economic emission dispatch-modeled using weighting method.

iv) Stochastic multi-objective hydrothermal scheduling-modeled using weighting method.

v) Stochastic combined heat and power dispatch-modeled using weighting method.

vi) Stochastic multi-objective active and reactive power dispatch-modeled using weighting method.