CHAPTER 4

HEURISTIC APPROACH FOR CONTROLLER TUNING

4.1 INTRODUCTION

This chapter presents the concept of intelligent optimization approach for optimal tuning of power controllers. These algorithms mimic the social behavior of nature, animals, surroundings and are also inspired based on some concepts of physics. They usually find the optimal solution or very close to the optimal solution within short time as compared to other conventional methods. Such algorithm can be applied to solve linear and non-linear problem as well. There are various intelligent optimization techniques based on the heuristic approach to solve computationally complex problems. Though various optimization techniques are available, but proper selection of the method for a specific problem is quite important. This selection depends on numerous factors such as time required to find the solution, accuracy of the desired output, flexibility, robustness of the algorithm, efficiency of the method and ease of algorithm for solving specific problem. In this work, some well-known optimization techniques (Particle Swarm Optimization (PSO), Firefly Algorithm (FA) and Gravitational Search Algorithm (GSA)) have been used to develop knowledge domain structure for all the controllers connected in the system for various operational shifts.

4.2 CONCEPT OF HEURISTIC OPTIMIZATION TECHNIQUES

Since traditional methods sometimes have difficulties in solving the real world problems due to non-linearity, flexibility and multi-dimensionality of the problems, intelligent/soft computing paradigms are the perfect replacement for solving such type of problems. These intelligent optimization algorithms inspired by nature or physics consist of some basic elements that help to solve complex real-life problems. These algorithms usually find the solution very close to the best one with faster time and with ease. These algorithms have mainly two aspects: Exploration and Exploitation.

Exploration determines the ability of the search algorithm to find new solutions far from the current solution in the search domain. It moves the algorithm in the new directions in the quest for optimal solution and also diversifies search space in order to avoid getting trapped in local optima. However, exploitation leads algorithm to search the surrounding space nearby the current solution and intensifying (refining) the search in the vicinity of local optima. So, algorithms which possess both the attributes tend to give best possible solutions. In order to be more reliable as a search algorithm, they need to establish a good ratio between exploitation and exploration. The concept of these populations based search algorithms can be described in three steps: Self-adaptation, co-operation and competition.

a) Self-Adaptation

Performance of search algorithms largely depends on the characteristic of generation's (initial population) distribution of all the members/elements. Self-adaptation improves each member of the population one by one by adjusting search parameters such as population size, recombination probability, and mutation rates etc. It does not only improve the optimal solution but enhances efficiency as well. The task of finding optimal solution gets complicated as problem dynamics changes, thus the parameters need to vary during the evolutionary process. This increases the need of self-adaptability with steady modification in search algorithm.

b) Co-operation:

In the co-operation process, individual solutions exchange the information among themselves to form new solutions close to the optimal point. Each member learns from previous experience and also from the experience of other members. This

helps to arrive at global solution fast with search process evolved and prevent sticking to the local optima.

c) Competition:

In competition process, each member competes with all other members trying to survive. All members understand of being eliminated if not perform well in finding global solution. So along as sharing information with all other members is concerned, each one of the member also tries to outperform leading to fast convergence. This quest of being the best helps the algorithm to perform with accuracy.

All above steps determine the effectiveness of intelligent optimization techniques in search process of optimal solution for any problem. In this thesis, three well-known optimization techniques Particle Swarm Optimization (PSO), Firefly Algorithm (FA) and Gravitational Search Algorithm (GSA) have been used to develop knowledge domain structure for the controllers by obtaining best possible parameters at different system operating conditions.

4.3 PARTICLE SWARM OPTIMIZATION (PSO) TECHNIQUE

Particle Swarm Optimization (PSO) was introduced by Eberhart and Kennedy in 1995 [77]. It is population based search algorithm which mimics the social behavior of bees, birds or school of fishes. It tries to simulate the unpredictable choreography of flocking of birds. PSO is initialized with some population of random candidate solutions. These candidate solutions are called particles, and the values of each condition solution are termed as the position of each particle. The position of each particle within the swarm is represented by a vector in the multidimensional search space. All the particles position has been updated to next position based on the particles

velocity. Each particle velocity is updated based on previous velocity and best position of each particle along with the best position it has explored so far including all the particles.

Particle swarm optimization process is iterative in nature, and the outcome of this algorithm depends on the stopping criteria. This stopping criterion may include fixed number of iteration or based on the desired error response of the objective function. It has been proven that this simple algorithm structure can deal difficult optimization problems efficiently. Originally, PSO was developed for real values based problems, but Kennedy J. and Eberhart reported this algorithm for discrete valued based optimization problems [165].

4.3.1 Original Concept of Particle Swarm Optimization

PSO was originally developed for real values based optimization problems. The main idea was to simply produce computational intelligence by exploiting basic analogous of social behavior of bees and birds along with their interactions. Numbers of particles are placed in the search domain to solve some problems, where each particle calculates the value of the objective function with their respective position. Each particle moves its current position to next position in the search space with the help of its current position, its best position throughout the iteration ('*pbest*') and best position among all the particles throughout the previous iteration ('*gbest*'). Particles velocities are calculated by the Equation (4.1). Based on each particle's velocity new particles position is calculated by Equation (4.2). Next iteration takes place after all the particles have been moved to the new position.

$$v_{id}^{\,t+1} = v_{id}^{\,t} + r1 * c1(pbest^{t} - x_{id}^{\,t}) + r2 * c2(gbest^{t} - x_{id}^{\,t}) \quad (4.1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{4.2}$$

Here, *gbest* is global best, *pbest* is personal best of particle 'i', c1 and c2 are acceleration coefficient and r1, r2 are random numbers in [0, 1]. x_{id}^{t} is the position of i^{th} particle in dimension 'd' for iteration 't' and v_{id}^{t} is the velocity of i^{th} particle in dimension 'd' for iteration 't'. The velocity of each particle is calculated by the formula shown previously and then with the help of random number, particle's position is updated.

4.3.2 Binary Particle Swarm Optimization

PSO was originally developed for real-valued spaces but later defined for discrete valued spaces where the domain of the variables is finite. In BPSO, particle's positions are in the form of zeros and ones, which are converted into real values with the help of binary to the real conversion and then objective function is calculated. In this concept, a particle will decide on "yes" or " no" / "true" or "false". These binary values can be a representation of real value in binary search space. In binary PSO, particle's personal best and global best is updated in real-valued version. The major difference between binary PSO with that of real-valued version is that velocities of the particles are rather defined in terms of probabilities that a bit will change from zero to one or vice versa. The velocity of each particle must be restricted within the range of [0, 1]. The normalization function used here is a sigmoid function. The equations for updating the position of particles are given as follows:

$$v_{id}^{t+1} = \omega * v_{id}^{t} + r1 * c1(pbest^{t} - x_{id}^{t}) + r2 * c2(gbest^{t} - x_{id}^{t}) \tag{4.3}$$

$$S(v_{id}^{t+1}) = \frac{1}{1 + \exp(-v_{id}^{t+1})} \tag{4.4}$$

$$x_{id}^{t+1} = \begin{cases} 1, & \text{if } \text{rand} < S(v_{id}^{t+1}) \\ 0, & \text{otherwise} \end{cases} \quad (4.5)$$

Each particle's velocity is calculated based on the '*pbest'* and '*gbest'* of the entire iteration as shown in Equation (4.3). The position of each particle is updated based on the sigmoidal transformation function given in Equation (4.4). Sigmoid transformation is applied to squash particles velocity into the range [0, 1]. This iterative process goes on until the stopping criteria meet. The final '*gbest*' is the optimal solution of the problem and corresponding objective function is the final cost function.

4.3.3 Parameters Selection of Particle Swarm Optimization Technique

Parameters of PSO govern the effectiveness of the algorithm, so proper selection of these parameters need to be taken care of. These parameters include population of the particles, dimension of the particles, range of particles, acceleration coefficient, inertia weight, and stopping criteria.

Typical range of deciding the population of particles is 30-50 but can go up to 70-80. The size of particles population depends on the search space problem. The value of population will be high for large search space and low for small search space. Dimension of the particles is determined by the number of variables to be optimized for a specific problem. The range of the particles is determined by the problem to be optimized, where for each dimension of the particles, different range can be specified. Acceleration coefficient c1 and c2 determines the direction of the algorithm moving towards local optima or global optima. Here, c1 and c2 represent the weightage of local and global optima respectively for the movement of each particle. Inertia weight (ω) in Equation (4.3) can be interpreted as the fluidity of the medium in which particles move. At the start of the iteration value of ω is high (around 1) but reduces as algorithm

reaches its best optimal position, thus requires precise movement. Stopping criterion may include fixed number of iteration or based on the desired error response of the objective function.

4.3.4 Step Procedure to Develop Knowledge Domain Structure with PSO Technique

PSO is used to build the knowledge domain structure for multi-area power control at various system operating conditions. Power control requires precise real and reactive power regulation dynamically for respective areas of an interconnected power system. Appropriate power control has been ensured by linking system state variables with the operational shift. Optimal retuning of all controllers parameters ensures quick oscillation damping. PSO has been used to find the best possible control parameters for all controllers at specific operating conditions. All these parameters are stored in their respective controller's knowledge domain database.

Integral Time Multiplied by Absolute Error (ITAE) is considered as the objective function to build knowledge domain structure by PSO in power network. Formulation of ITAE is denoted below:

$$J\{e(t)\} = ITAE = \int_0^\infty t \times |e(t)|\, dt \tag{4.6}$$

Here, e(t) represents the sum of all deviated states from their desired value (error) of the entire power network. As shown in the modeling part (Chapter 3) entire network is represented in state space form.

$$\Delta \dot{x} = [A]\Delta x + [B]\Delta u$$

Where $\Delta x = \left[\Delta\omega, \Delta\delta, \Delta E'_q, \Delta E_{fd}, \Delta X5, \Delta UE\right]^T$ with PSS model and

$\Delta x = \left[\Delta\delta, \Delta\omega_1, \Delta\omega_2, \Delta E'_{q1}, \Delta E_{fd1}, \Delta E'_{q2}, \Delta E_{fd2}, \Delta V_{dc}\right]^T$ with FACTS model

Complete system state (Ac) is calculated and then based on the different perturbation, response of all the state variables are derived in time domain. Variations of all the state variables with their desired values (i.e., error) are included in the ITAE stamping with their time. The complete objective function calculation is based on the summation of all the state variables deviation. These state variables deviation depend upon the tuning of controller's parameters which are tuned by PSO. Let N numbers of controllers are connected in the entire power system. The variable of each controller is denoted by X. (In this research work 2 types of controllers are connected-Power System Stabilizers (control parameters: T1, T2 and Kc) and FACTS devices (control parameters-(m_e, m_b, δ_e and δ_b)).

Objective function (ITAE):

$$F(X) = J\{e(t)\} = \int_0^\infty t \times |e(t)|\, dt \tag{4.7}$$

So total numbers of variables needed to be tuned by PSO are:

X={X1, X2, ... XN}; N=total number of variables

These are the procedural step to develop knowledge domain structure by tuning of controller's parameters with PSO optimization technique.

1. Define objective function F(X), where X=(X1, X2,…..XN); X=controllers parameters with their boundary.

2. Generate initial population of all the particles with their positions, Xi (where i=1:N).

3. Find fitness values with Equation (4.7) and calculate '*pbest'* and '*gbest'* with the help of fitness function.

4. Calculate particles velocity using current velocity, '*pbest'* and '*gbest'* with Equation (4.3).

5. Calculate the value of sigmoidal function for each particle using Equation (4.4).

6. Find new position of each particle with their velocity using Equation (4.5).

7. Go to step 3 until a stopping criterion is satisfied.

Flowchart of PSO technique has been shown in Figure 4.1. After stopping criteria meets, final optimal tuned parameters of PSS and FACTS devices are obtained for particular system operating condition, which is then stored in their respective knowledge domain stamped with their operational shift. Development of the complete KD structure continues for various system operating conditions similarly. In this concept, parameters of controllers are tuned offline for most of the expected system operating conditions (may be based on historical shift observed earlier), then stored in respective knowledge domain and when in online condition some operating condition occurs other than the stored condition, knowledge domain will be updated with newer shift over and above old one and corresponding knowledge domain database is changed. The knowledge domain up-gradation is ensured with changing system conditions. Thus, new sets of parameters are available which can be mapped with inference mechanism dynamically to have better performance of system recovery even with relatively larger perturbation. The hardware realization can be achieved with changing digital technology and intelligent automation.

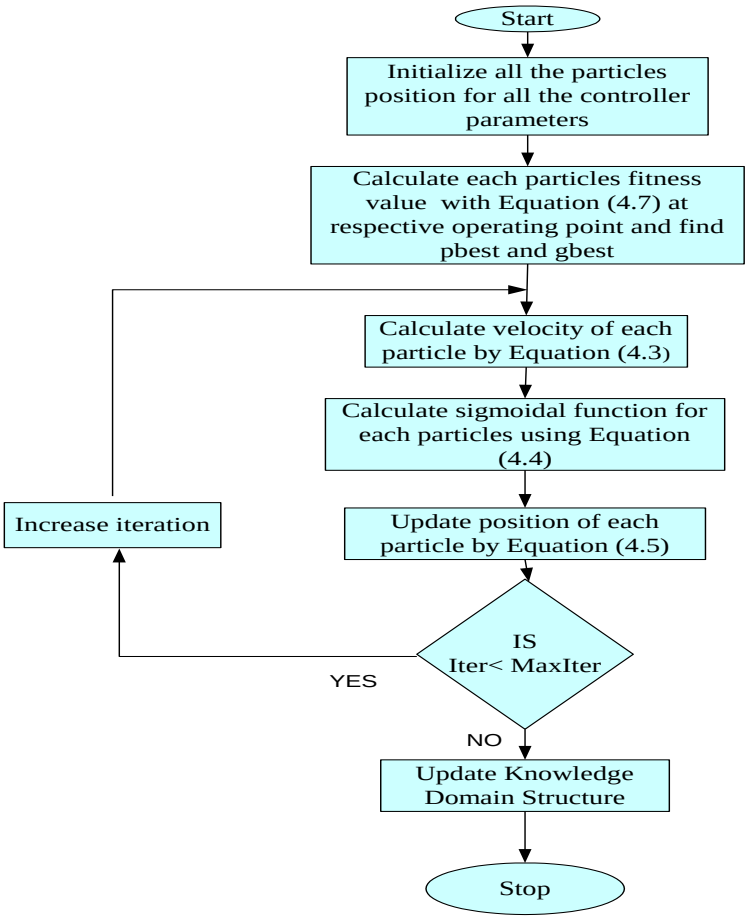

Figure 4.1 Flowchart of particle swarm optimization to develop knowledge domain structure

4.4 FIREFLY ALGORITHM (FA)

Firefly algorithm was developed by the Xin-She Yang and Hingshi He in 2008 to solve the optimization problems [166]. The concept of this algorithm is based on the flashing behavior of fireflies. They are characterized by their unique flashing lights. These flashing lights work as signals for mating. They attract their mates due to pheromones. These behaviors of fireflies helped to develop an optimized problem known as firefly algorithm. In this algorithm, physical formula of light intensity is used which decreases the light intensity as increase in distance. However, light absorption becomes weak with increase in distance from the light source. The same pattern has been associated with the objective function to be optimized. In this algorithm, some rules have been idealized for corresponding to the behavior of fireflies [167-168].

- All the fireflies attract to each other.
- The attractiveness of each firefly is proportional to the brightness and both decrease with the increase in the distance. So with two flashing fireflies, brighter one attracts the less bright firefly.
- The brightness of each firefly is directly proportional to the cost of objective function.

The light intensity varies according to the inverse square law i.e.:

$$I(r) = \frac{Is}{r^2} \tag{4.8}$$

Where I(r) is the light intensity at a distance 'r' and *Is* is the intensity of the source. Moreover, air absorbs light which weakens as the distance increases.

When the medium is given with a fixed light absorption coefficient γ, light intensity can be written as:

$$I(r) = I_0 e^{-\gamma r^2} \tag{4.9}$$

Here, I_0 is the initial light intensity. The flashing light is formulated by calculating the objective function to be optimized. Since firefly's attractiveness is proportional to light intensity seen by adjacent fireflies, we can define attractiveness β of firefly as:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{4.10}$$

The distance between any two fireflies 'i' and 'j' at x_i and x_j respectively is the Cartesian distance. β_o is the attractiveness of the source firefly.

$$r_{ij} = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{4.11}$$

Here k is the dimension of firefly. The movement of a firefly 'i' is attracted to another more attractive firefly 'j' is determined by:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2} (x_j - x_i) + \alpha\varepsilon \tag{4.12}$$

Where the second term is due to attraction while the third term is randomization with α being the randomization parameter and ε is the vector of random numbers.

4.4.1 Step Procedure to Develop Knowledge Domain Structure with FA Technique

Firefly Algorithm (FA) has been used to develop knowledge domain structure for the complete network at various system operating conditions. Proper retuning of all the controller's parameters ensures quick oscillation damping in the network. FA is used to find the best possible control parameters of all controllers at specific operating conditions and stored in their respective controller's knowledge domain.

The objective function used to generate knowledge domain by using FA is ITAE (minimization problem) and given in Equation (4.7).

Let N numbers of controllers are connected in the entire power system. The variable of each controller is denoted by X. So total numbers of variables needed to be tuned by FA are:

X={X1, X2, ... XN}

These are procedural steps for tuning parameters of controllers with Firefly Algorithm:

1. Define objective function F(X), where X=(X1,X2,…..XN); X=controllers parameters
2. Generate initial population of fireflies Xi (where i=1:N)
3. Find light intensity Ii at Xi by Equation (4.7) and define absorption coefficient γ.
4. While (t<Iteration)

For k=1:N

For m=1:N

If (Ik<Im), move firefly k towards m; end (if)

Evaluate new solution and update light intensity.

End for m.

End for k.

5. Rank the fireflies and find the optimal best solution.
6. Store all the values of variables (Controller's parameters) in their respective knowledge domain.

Development of complete knowledge domain structure continues for various system operating conditions. Parameters of controllers are tuned offline for most of the expected system operating conditions (may also be linked with historical shift observed derived from real-time information available with load dispatch control centers). Flowchart of FA based knowledge domain structure development is shown in Figure 4.2.

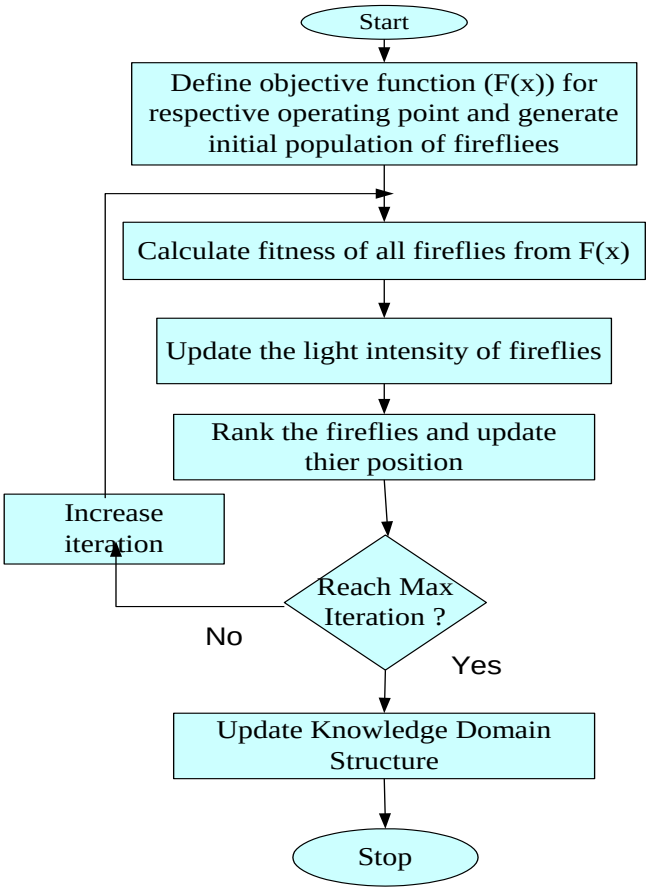

Figure 4.2 Flowchart of firefly algorithm to develop knowledge domain structure

4.5 GRAVITATIONAL SEARCH ALGORITHM (GSA)

GSA is a new intelligent technique introduced for the optimization problem. This algorithm is developed by Esmat Rashedi *et al*. in 2009, which is based on the Newton's gravitational law with interaction of masses [169]. In GSA, search agents are considered as the object (or candidate solutions) and the performance of each object is measured by their masses. All these objects/agents interact with each other by gravitational law and the law of motion. This results in global movement of all the objects towards the objects with heavier masses because heavier the mass, bigger the attraction force. Heavy mass of the object corresponds to the better solution of the problem. As the iteration process goes on each agent tries to gain heavy mass and move towards the optimal solution.

Consider, N number of objects in d dimensions, then position of each agent 'i' is defined as:

$$X_i = (X_i^1, ...X_i^d, ...X_i^n) \text{ for i=1,2,…N,} \quad (4.13)$$

Where X_i^d represents the position of ith agent in dth dimension.

The force acting on mass 'i' due to mass 'j' is:

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) * M_{aj}(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t)) \quad (4.14)$$

Here, $G(t)$ is the gravitational constant which determines the performance of GSA as it value decreases with increase in time, M_{aj} is the active gravitational mass of 'j' agent and M_{pi} is the passive gravitational mass related to agent 'i'. ε is small constant and R_{ij} is the Euclidian distance between agent i and agent j.

$$R_{ij}(t) = \| X_i(t), X_j(t) \|_2 \text{ and } G(t) = G(G_0, t) \quad (4.15)$$

Total force and acceleration on a single agent 'i' in a dimension d from all the other

agents are:

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^d(t),\ a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \tag{4.16}$$

Here, M_{ii} is the inertial mass of i[th] agent.

Velocity of each agent is updated by:

$$v_i^d(t+1) = rand_i * v_i^d(t) + a_i^d(t) \tag{4.17}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{4.18}$$

And the masses of each agent are updated by:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)},\ M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{4.19}$$

Here $best(t)$ and $worst(t)$ both depend on the objective problem.

4.5.1 Step Procedure to Develop Knowledge Domain Structure with GSA Technique

Gravitational search algorithm has been used to build the knowledge domain structure for multi-area power control at various system operating conditions. GSA is used to find the best possible control parameters of all controllers at specific system operating conditions. All these parameters are stored in their respective controller's knowledge domain.

Objective function minimized by GSA is given in Equation (4.7). Let N numbers of controllers are connected in the entire power system. The variable of each controller is denoted by X.

So total numbers of variables needed to be tuned by GSA are:

X={X1, X2, ... XN}

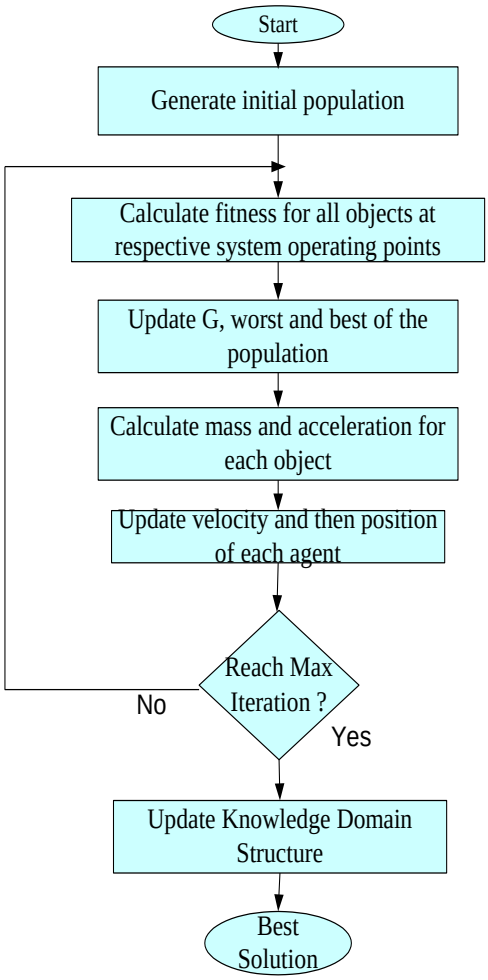

Figure 4.3 Flowchart of gravitational search algorithm to develop knowledge domain structure

These are the step procedure to incorporate tuning of parameters of controllers with gravitational search algorithm:

1. Define objective function F(X), where X=(X1,X2,…..XN); X=controllers parameters with their boundary.
2. Generate initial population of agents Xi (where i=1:N)
3. Find fitness values with Equation (4.7) and calculate *best* and *worst* agents with the help of fitness function.
4. Calculate masses of each agent with Equation (4.19).
5. Calculate total force and acceleration for each agent with Equation (4.16).
6. Update the velocities of each agent with Equation (4.17).

7. Move each agent to the new position according to Equation (4.18).
8. Go to step 3 until a stopping criteria satisfied.

After the stopping criteria meet, the final optimal tuned parameters of PSS and FACTS devices are obtained for particular system operating condition which is then stored in their respective knowledge domain stamping with their operational shift. Flowchart of GSA based knowledge domain structure development has been shown in Figure 4.3.

4.6 CONCLUSION

The basic concept of population-based intelligent optimization techniques using heuristic approach has been presented. Since these optimization techniques are very useful for solving the non-linear optimization problem in minimum time and the results have been found very close to the optimum solution. Such optimization techniques can be more effective not only in initial design of power controller but may be helpful further in tuning of the controllers as operational shift occurs in power network. All above discussed optimization techniques have been used to develop the knowledge domain structure for all power controllers such as PSS and FACTS devices connected in power system. Step procedures for all the three optimization technique to develop the knowledge domain structure have been reported in this chapter. The knowledge domain thus built has been used to design the effective controller for changing operational shift. The detailed application of knowledge domain building block and inference mechanism has been discussed in Chapter 5.