CHAPTER-1

INTRODUCTION

1.1 OVERVIEW

Electrical power systems are composed and worked to meet the nonstop variety of power demand. With the increment in energy demand, the size of our electrical power system is additionally expanding day by day and making electrical power system muddled. Rebuilding of the power sector has motivated new improvements in power system operation and arranging. To satisfy the energy need, amounts of power plants are joined in parallel to supply the system load by interconnection of systems. These days, real test is to satisfy the consumer's demand for power at least cost. Any given power system consisting of various generating stations, having their own particular characteristic operating parameters, are utilized to take care of the total consumer demand. Generally, the cost of operating these generators is not proportional with their output, therefore the test for power utilities is to attempt to adjust the total load among generators. It gets complicated when utilities attempt to represent the transmission loss and seasonal changes. Rebuilding of the power sector has motivated new improvements in power system operation and arranging. Although distributed generation (D.G) and demand response projects are not extremely recent, the present power system operational requirements is to give electricity in a conservative and proficient way, for the expanding rate of demand growth.

Economic Load Dispatch (ELD) is a technique to plan the power generator outputs as for the load demands, and to work the power system most economically, or as such, we can say that primary target of economic load dispatch is to designate the optimal power generation from distinctive units at the least cost conceivable while meeting all system constraints. The unpredictability of ELD relies on upon the components like size of the system, generator attributes and system constraints.

Generating the definite measure of power as demanded meeting all of the losses is crucial and takes after a complex methodology. ELD can be used to figure and to plan the obliged amount of power to be generated among the different generating units in the system . The operation of the different generating systems in a economical way is constantly viewed as a important factor in power industry while considering different enhanced type of limitations with respect to power generation, power demand and operational limits of the generating units.

This thesis is concerned with the ELD of all thermal system only. It is to be note that

all the generating units in a system don't took part in the economic dispatch. Nuclear units and huge steam units are keep running at consistent MW setting as it is desirable (because of some specialized reasons) to keep up the yield of such units at as consistent a level as could reasonably be expected. Rest of the units those partake in economic dispatch will be called controllable units. Fuel costs in base-load units then show up as an altered cost and don't show up in the economic dispatch issue. We consider the minimisation of those costs that, by fitting proper methodology, we can control it, i.e. the fuel costs in the controllable units.

The issue of economic operation of a power system or ideal power flow can be state as: Allocating the load (MW) among the different units of generating stations and among the different generating stations in such ways that, the general cost of generation for the given load demand is least.

This is an optimization problem, the objective of which is to minimize the generation cost function subject as per the general inclination of a given arrangement of straight and non-direct equality and inequality constraints. The issue is dissected, solved and afterward executed under online state of the power system. The information for the issue originates from conventional power flow study. For a given load demand, power flow study can be utilized to compute of active and reactive power generations, line flows and losses. The answer for this issue can't be ideal unless generally every one of the limitations of the system are fulfilled. We examine the economic scheduling issue in the following sections, however first we consider the requirements that should be addressed.

The majority of the industrial practices known so far consider a quadratic function of real power output generated for the working fuel cost of generators and valve point loading effect if considered in the event of thermal power plants calls for brokenness in the optimization problem. Accordingly, ELD problem show non-linear and non-convex elements because of the presence of valve point loading effects. Established optimization techniques incorporate constantly differentiable functions which are hard to handle and now and again, does not converge to optimum solution. The evolutionary computational techniques can handle such non-convex and non-differential objective functions effectively and give a practically ideal solution in an interval of time. Numerous meta-heuristic calculations can proficiently understand non-continuous, non-linear and non-convex optimization problems as being what is indicated techniques don't force any requirements on the given problem. Like these evolutionary algorithms, randomly took after artificial intelligence techniques, for example, say Particle Swarm Optimization (PSO) is likewise connected to numerous complicated optimization fields.

In this thesis work one of great heuristic method PSO is utilized. This strategy is in

view of the experience picked up from the investigation of artificial life and psychological exploration. Eberhart and Kennedy created PSO, in light of the similarity of the swarm of birds and the school of fish. One of the principle objectives is to look at how natural creatures carry on as a swarm and to reconfigure the swarm model computationally. It is understood that the PSO methods can give a top notch arrangement with basic implementation and quick convergence. PSO algorithm has been produced for the nonlinear consistent optimization issue to accomplish the best compromise arrangement. In this work the expense of generation is taken as the objective which is should have been be minimized. Adjustments in PSO have been indicated in some recent research papers. These changes makes PSO algorithm to demonstrates more preferable result over the Basic PSO. In this thesis work modifications have been done in PSO which is named as MPSO (Modified Particle Swarm Optimization). This change is done to enhance the speed and it gives better results.

1.2 AIM AND APPROACH

Our fundamental thought process in this thesis work is to solve the ELD (economic load dispatch) problem by the utilization of Modified Particle Swarm Optimization (MPSO) recognizing the cost of generation and the transmission line losses of the system for which IEEE 5, 14, & 30 bus system have been studied. The optimum point or the best compromise arrangement have been completed by utilizing MPSO as a part of which we take the best esteem result from the particles by which we figure out the cost of generation and the transmission line losses of the system.

The work has been done in the following way:

a. Analysing about Particle Swarm Optimization and organizing its algorithm in MATLAB R2013a.
b. Explanation of distinct mathematical benchmark functions using SBPSO.
c. Formulation of Economic Load Dispatch (ELD) recognizing the cost of generation and the transmission line losses of the system for IEEE 5, 14, & 30 Bus System utilizing SBPSO.
d. Generation of non inferior sets of IEEE 5, 14, & 30 bus systems.
e. Accomplishment of optimum point for IEEE 5, 14 and 30 bus system with lesser no of kount acknowledging the cost of generation and the transmission line losses of the system

1.3 LITERATURE REVIEW

Basic PSO: Eberhart and Kennedy [1] described a concept for the optimization of nonlinear functions using particle swarm methodology is introduced. The evolution Benchmark of the paradigm is described, and applications, including nonlinear function optimization and neural network training, was proposed. The relationships between particle swarm optimization and both artificial life and genetic algorithms was described.
Ajith Abraham ,Hi Guo and Hanpo Liu[2] implemented PSO and ACO algorithms on same mathematical benchmark function as Griewank function ,Schwefel function, Quadratic function and also an real world applications as travelling sales man problem and data mining . They also analysed and discussed the results in detail.

Karin Zielinski and Rainer Laum [3]evaluated various stopping criteria's on the basis of movements made by the particles ,improvements based criteria's and distribution based criterion for a constrained single – objective particle swarm optimisation algorithm. In this paper ,optimisation of power allocated scheme for code division multiple Access (CDMA) system has been considered : Improvement based movement based and distribution based stopping criterion .these criterions have been compared on the basis of convergence rate and success performance. It was further observed that there was no use of combining different criteria's on this incorporates adjustments of more parameters.

MODIFIED:

W.B. Laydon et al [4] used kernel to provide the values for each particle of a swarm which guides the unit as a whole. They solved one- dimensional multi-model 3-peaks and rastrigin function problem using kernel.

Marks S. Voss[5] introduced principal component PSO I e PCPSO in which particles are made to fly in two separate spaces simultaneously, one in traditional n- dimentional space and a rotated m-dimensional z- spaces. Where $m \leq n$.PCPSO algorithm has been introduced using Greiewank function.

Jaco F.Schutte and Albert A. Grroenwold [6] studied the variants of PSO algorithms and applied to Dixan – Szego test set. The variations studied were –constant inertia weight, linear inertia reduction, limit on maximum velocity, construction factor, dynamic inertia and maximum velocity reduction. It was observed that constriction and dynamic inertia weight

both affect reliability and cost. Dynamic inertia reduction was found to be less sensitive than constriction factor.

Wei-Bing Liu and Xian Jia Wang [7]introduced Evolutionary game (EGPSO) in which the behaviour of particles are modelled using Replicator dynamics and multi –start technique. This technique overcomes premature convergence and has better convergence property than traditional pso.

Wei Zu et al [8] proposed a new technique PSOED based on particle equilibrium distribution in which a sub-optimum trap i.e. clustering of particles within a subarea of problem scope is avoided.This technique is applied to various benchmark functions and is shown to be better than basic pso and GA.

Sevkan et al [9] introduced multi-dimensional pso (MD-PSO)where swarm particles can seek both positional and dimentional optima. They also proposed FGBF(Fractional Global Best Function) technique to avoid premature convergence. This technique is then applied to multi model dynamic environment and is shown to track the global optima with minimum error.

Touahria[10] illustrated the effect of excluding the redundant particle from current iteration.

Huanhuan Ji et al [11] proposed a bi-swarm particle swarm optimization with cooperative co-evolution (BPSO-CC).In this model second swarm was generated from the first swarm which conducts the local search. The proposed technique was implemented on size benchmark functions in dimensions of 100 to 500.

Zhe Li and Yong chen [12] have introduced a new pso with parallel processing and color quantization.

Weidong Ji and keai Wang[13] combined PSO with gradient method, which avoids immature convergence.

Ismail et al [14] proposed a novel multi –state particle swarm optimization (MSPSO) to solve discrete problems. It was applied to two benchmark instances of travelling salesman problem. (TSP). The proposed technique was compared with Binary Particle swarm optimization (Bin-PSO). It was observed that the proposed technique gave better solution compared Bin-PSO and was found to be simple in complexity.

Kyle Robert Harison [15] hybridized GA with PSO. They also claimed new version of pso avoids premature convergence.

Nikhil Padhya et al [16] suggested three different PSOs with boundary handling approaches in this paper ,the authors have proposed two boundary handling methods- Inverse parabolic spread Distribution and Inverse parabolic confined Distribution. These were compared with existing boundary handling methods- Random, Periopdic, Set on boundary, SHR and Exponential Distribution for four test functions.

Zahara Beheshti et al [17] proposed binary accelerated PSO. They have shown that new pso requires only common controlling parameters viz no. of genewrations and population size.

Luis Miguel Rios and Nikolaos Sahinidias[18] presented a review of derivative free algorithms including PSO for constrained problems.They combined twenty – two(22) such algorithm and implemented on a test set of 502 problems. It was observed that all solvers provided the best solution for at least some of the test problems and there is no single solver which provides best result for all the problems.

Zhimin chen et al [19] presented an organizational adjustment PSO based particle filter (OAPSO-PF) algorithm which allowed the particles to adopt to environment and reach the global optimum.

Lin Lu et. al. [20] developed a hierarchical structure poly –particle swarm optimization(HSPPSO). Approach using the hierarchical structure concept of control theory. This algoritm was implemented on four benchmark functions –Spherical, Rosenbrock, Griewank and Rastrigin and was also compared with PSO. HSPPSO was found to search better for global optimum, and converged faster.

Kyle Robert Harison [21] hybridized GA with PSO. They also claimed new version of pso avoids premature convergence.

Xan Zhe ping et al [22] also presented a pso with two sub population. Liu Jin –yue et al [62] proposed linearly decreasing weight PSO (LDWPSO) algorithm. And introduced mutation operator to improve the global and local search . Ability of the algorithm was tested on two non linear functions- Ackeley and Rastrigin functions and compared the performance with standard pso. It was observed that this improved algorithm increased the convergence speed as well as the global search capability.

Nikhil Padhya et al [23] suggested three different PSOs with boundary handling approaches in this paper ,the authors have proposed two boundary handling methods- Inverse parabolic spread Distribution and Inverse parabolic confined Distribution. These were compared with

existing boundary handling methods- Random, Periopdic, Set on boundary, SHR and Exponential Distribution for four test functions . Inverse parabolic spread Distribution was found to be the most robust and consistent method.

Zahara Beheshti et al [24] proposed binary accelerated PSO. They have shown that new pso requires only common controlling parameters viz no. of genewrations and population size.

Luis Miguel Rios and Nikolaos Sahinidias[25] presented a review of derivative free algorithms including PSO for constrained problems.They combined twenty – two(22) such algorithm and implemented on a test set of 502 problems. It was observed that all solvers provided the best solution for at least some of the test problems and there is no single solver which provides best result for all the problems.

Zhimin chen et al [26] presented an organizational adjustment PSO based particle filter (OAPSO-PF)algorithm which allowed the particles to adopt to environment and reach the global optimum.

Lin Lu et. al. [27] developed a hierarchical structure poly –particle swarm optimization(HSPPSO)

Approach using the hierarchical structure concept of control theory. This algoritm was implemented on four benchmark functions –Spherical, Rosenbrock, Griewank and Rastrigin and was also compared with PSO.HSPPSO was found to search better for global optimum, and converged faster.

ELD:

Megahed *et al.* [28] developed a method for solving the economic load dispatching problem by changing it from constrained nonlinear programming problem to a sequence of constrained linear programming problems. The formulation of the load scheduling is exact in the sense that all the system voltages, active and reactive generation, as well as the phase angles are considered as independent variables. In addition, the effect of bus voltages on the loads is taken into consideration

Happ [29] reviewed the progress of optimal dispatch, also called economic load dispatch, since its inception to the present in chronological sequence. The classic single area as well as multi area cases is summarized, and the important theoretical work in optimal load flows

suggested to date reviewed. Approaches to the optimal load flow taken by industry are also reported, as well as an itemization of problems that still remain to be solved.

Kwatny and Athay [30] presented the coordination of the economic load dispatch and regulation functions of automatic generation control in electric power systems. The point of view taken is that such coordination appropriately takes place at the regulation or load frequency control level. Thus, the coordinating controller is obtained through the formulation of a suitably extended load frequency control problem in the context of linear multivariable control theory.

Aoki and Satoh [31] presented an efficient method to solve an economic load dispatch problem with dc load flow type network security constraints. The conventional linear programming and quadratic programming methods cannot deal with transmission losses as a quadratic form of generator outputs. In order to overcome this defect, the extension of the quadratic programming method is proposed, which is designated as the parametric quadratic programming method. The upper bounding technique and the relaxation method are coupled with the proposed method for the purpose of computational efficiency. The test results show that the proposed method is practical for real-time applications.

Lin and Viviani [32] presented a method to solve the economic power dispatch problem with piecewise quadratic cost functions. The solution approach is hierarchical, which allows for decentralized computations. An advantage of this approach is the capability to optimize over a greater variety of operating conditions. Traditionally, one cost function for each generator is assumed. In this formulation multiple intersecting cost functions are assumed. This method has application to fossil generation units capable of burning gas and oil, as well as other problems which result in multiple intersecting cost curves for a particular unit. The results show that the solution method is practical and valid for real-time application.

Ramanathan [33] presented an extremely fast, simple, efficient and reliable economic load dispatch algorithm. The algorithm utilizes a closed form expression for the calculation of the Lambda, as well' as taking care of total transmission loss changes due to generation change, thereby- avoiding ,any iterative processes in the calculations. The closed form expression

presented for Lambda can be used with 'any type of incremental transmission loss calculation. For this algorithm, penalty factors are derived based upon the Newton's method.

Walters and Sheble [34] used genetics-based algorithm to solve an economic dispatch problem for valve point discontinuities. The algorithm utilizes payoff information of candidate solutions to evaluate their optimality. Thus, the constraints of classical Lagrange techniques on unit curves are circumvented. The formulations of an economic dispatch computer program using genetic algorithms are presented and the program's performance using two different encoding techniques is compared. The results are verified for a sample problem using a dynamic programming technique.

Abdullah and Bakar [35] presented the implementation of hybrid particle swarm optimization for solving Economic-Emission Load Dispatch Problem (EELD). They studied, the hybrid Evolutionary programming (EP) and Particle Swarm Optimization (PSO) named Evolutionary Particle Swarm Optimization (EPSO) was proposed. The effectiveness of the EPSO algorithm has been tested on the IEEE 30 bus system and the results obtained were compared with the other reported algorithms. The results also reveal the capability of the proposed EPSO for obtaining the best fuel cost compared to PSO.

Bhattacharya and Chattopadhyay [36] presented a novel Particle Swarm Optimizer combined with Roulette selection operator to solve the economic load dispatch (ELD) problem of thermal generators of a power system. Several factors such as quadratic cost functions with valve point loading, transmission Loss, generator ramp rate limits and prohibited operating zone were considered in the computation models. The experimental results showed that the proposed modified PSO method is indeed capable of obtaining solution in less time and in fewer numbers of iterations.

Dasgupta and Banerjee [37] explains different techniques has used to solve these problems. Recently, the soft computing techniques has widely used in practical applications. They showed successful implementation of four evolutionary algorithms, namely particle swarm optimization (PSO), particle swarm optimization with constriction factor approach (PSOCFA), particle swarm optimization with inertia weight factor approach (PSOIWA) and

particle swarm optimization with constriction factor and inertia weight factor approach (PSOCFIWA) algorithms to economic load dispatch problem. Power output of each generating unit and optimum fuel cost obtained using all four algorithms has been compared.

N.K.Jain and Uma Nangia [38] explained Particle Swarm Optimization converges to local optima, especially in some complex issue like optimization of high dimension function. They observed that the traditional particle swarm optimization algorithms converses rapidly during the initial stage of a search, but in course of time becomes steady considerable and gets trapped in a local optima. They presented four evolutionary optimization models (IPSO 1, 2, 3, 4) based on the particle swarm optimization algorithms for Economic Load Dispatch considering cost of generation. Their analysis suggests that IPSO (Improved Particle Swarm Optimization) significantly improves the performance with less no of iteration. They implemented different IPSO to ECONOMIC LOAD DISPATCH to get optimum value of cost with less no of iteration.

Liu, Han and Zhou [39] explained chaotic particle swarm optimization (CPSO) algorithm to solve the optimal dispatch problem, adopting the adaptive inertia weight to accelerate the convergence speed. They improved hybrid optimization algorithm from particle swarm optimization (PSO) algorithm by chaotic searching in the neighborhood to avoid getting into the local optimum, with the algorithm steps listed in the paper. They considered a numerical example and analyzed, verifed the validity of the hierarchical optimization mode and CPSO.

1.4 PLAN OF THESIS

This thesis has been arranged in five chapters. The contents of the chapters are briefly outlined as indicated below:

Chapter 1 is an introduction chapter. It describes the overview of thesis, aim and approach of the problem taken up for thesis work and literature review.

Chapter 2 describes the introduction, algorithm, flowchart and mathematical formulas of Particle Swarm Optimization.

Chapter 3: Explores the concepts of Selection Based Particle Swarm Optimization algorithm in MATLAB R2013a and its application on various mathematical benchmark functions. Analysis of various parameters in SBPSO algorithm has also been carried out.

Chapter 4: Discusses the solution of Economic Load Dispatch by SBPSO for IEEE 5, 14 and 30 bus systems.

Chapter 5: Conclusion and the future work directions have been discussed.

References and Appendix are at the end of the thesis.

CHAPTER-2

PARTICLE SWARM OPTIMIZATION

2.1 INTRODUCTION

A modern heuristic optimization method, for example, simulated annealing, evolutionary algorithms, neural networks, and ant colony have been given much consideration by numerous specialists because of their capacity to find a practically global optimal solution in EDPs. One of these modern heuristic optimization ideal models is the particle swarm optimization (PSO).

PSO is a kind of evolutionary algorithm taking into account a populace of individuals and roused by the simulation of social conduct instead of the survival of the fittest individual. It is a populace based evolutionary algorithm. Like the other populace based evolutionary algorithms, PSO is initialized with a populace of random solutions. Not at all like a large portion of the evolutionary algorithm is solution (individual) in PSO likewise connected with a randomized velocity, and the potential solutions, called particles, are then "flown" through the problem space. PSO algorithm has been produced for the nonlinear consistent optimization issue to accomplish the best compromise arrangement. In this work the expense of generation is taken as the objective which is should have been be minimized. Adjustments in PSO have been indicated in some recent research papers.

The most striking distinction in the middle of PSO and the other evolutionary algorithms is that PSO picks the way of collaboration over rivalry. The other algorithms normally utilize some type of annihilation, survival of the fittest. Interestingly, the PSO populace is steady and individuals are not crushed or made. Individuals are influenced by the best execution does not have hereditary operators like crossover in the middle of individuals and of their neighbors. Individuals inevitably meet on optimal points in the problem domain. Also, the PSO traditions transformation, and other individuals never substitute particles during the run. Instead, the PSO refines its pursuit by attracting the particles to positions with great solutions.

Additionally, contrasted with genetic algorithms (GAs), the information sharing mechanism in PSO is significantly diverse. In GAs, chromosomes offer information with one

another. So the entire population moves like an one gathering towards an optimal zone. It is a restricted information sharing component. The advancement searches for the best solution. In PSO, every one of the particles have a tendency to converge to the best solution rapidly, comparing with GA, even in the local version in most cases.

Particle swarm advancement is fundamentally a population-based searching procedure which searches in parallel and does not prompting elimination of any of the arrangement. It is a sort of searching method which follows its movement taking into account the advancing development of birds rushing or fish schooling search for nourishment. It introduces particles randomly into the predetermined search space and these particles accumulate data through their corresponding positions. The alteration in position of every molecule in the swarm is in light they could call their own personal best experience and their neighbor's experiences. In PSO, just g_{best} (or p_{best}) gives out the data to others. The inertia weight presented at the velocity updating step is responsible for the energy of every molecule. It was presented by Shi and Eberhart and it meets expectations by weighing the inclusion of the past for the purpose of dispensing with the necessity for velocity clamping. Estimation of w chooses the particle velocity overhauling and in this manner entire framework bearing is taking into account its quality. With w having a quality more prominent than 0, divergent behavior of the framework come about because of expansion in framework's velocity and accordingly particles neglect to acquire the best position. With w being under 0, deceleration of particles happens which bring about moderate convergence to the best solution. Static inertia qualities or versatile qualities may lead to convergent solution.

PSO has various key elements that turn out to be extremely dependable in distinctive applications where conventional optimization techniques may come up short when compared with other evolutionary optimization algorithms. Following are a advantages of the PSO over other conventional algorithms:

- It makes use of basic logical and mathematical functions which are easy to handle and operate with a succession of methodology.
- It has less working parameters which are easy to work.
- It uses real esteemed operational integers which dispenses with the requirement of converting binary form to real coded structure as in GA.
- It is better in terms of time utilized for computation and memory storage requirement.

2.2 CONCEPT OF SWARM AND PARTICLE

The term swarm has a premise in the literature. Specifically, the authors utilize the term as per a paper by Millonas [40], who added to his models for applications in artificial life, and enunciated five fundamental standards of swarm intelligence.

- *Proximity principle*: As per this standard, the population ought to have the capacity to do basic space and time computations.

- *Quality principle*: As per this standard, the population ought to have the capacity to react to quality factors in nature.

- *Principle of diverse response*: As per this standard, the population ought not to confer its activities along unreasonably narrow channels.

- *Principle of stability*: As per this standard, the population ought not change its method of conduct each time the environment changes.

- *Principle of adaptability*: As per this standard, the population must have the capacity to change conduct mode when it's justified the computational cost..

3.3 BASIC PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization has establishes in two fundamental segment strategies. Maybe more clear are its binds to artificial life and to bird flocking, fish schooling, and swarming hypothesis specifically. It is likewise related, be that as it may, to evolutionary computation, and has binds to both genetic algorithms and evolution strategies. Particle swarm optimization contains an exceptionally straightforward idea, and standards are executed in a couple lines of PC code. It requires just primitive mathematical logic, and is computationally reasonable as far as both memory need and speed. Early testing has discovered the execution to be powerful with a few sorts of problems. PSO has likewise been exhibited to perform well on genetic algorithm test

functions, and it has all the earmarks of being a promising methodology for robot task learning.

PSO is initialized with a gathering of random particles (solutions) and then scans for optima by redesigning generations. In every iteration, every molecule is overhauled by taking after two "best" values. The first is the best solution (fitness) it has accomplished in this way. The fitness quality is additionally put away. This quality is called pbest. Another "best" esteem that is followed by the particle swarm optimizer agent is the best esteem, got so far by any particle in the populace or it is the best esteem among every one of the estimations of pbest. This best esteem is a global best and called gbest. At the point when a particle participates of the populace as its topological neighbors, the best esteem is a local best and is called pbest..

The PSO idea comprises of, at every single time step, changing the velocity (accelerating) every particle toward its pbest and gbest areas (global adaptation of PSO). Acceleration is weighted by a random term, with isolated random numbers being created for acceleration toward pbest and gbest areas. There is likewise a local form of PSO in which, in addition to pbest, every particle stays informed regarding the best solution, called lbest, achieved inside of a topological neighbourhood of particles.

The (original) process for actualizing the global variant of PSO is as per the following:

1) Initialize a population (array) of particles with random positions and velocities on ***d*** dimensions in the problem space.
2) For every particle, assess the desired optimization fitness function in d dimensions.
3) Compare particle's fitness assessment with particle's pbest. In the event that present value is superior to pbest, then set pbest worth equivalent to the present value, and the pbest location equivalent to the present location in d-dimensional space.
4) Compare fitness evaluation and the populace's general previous best. On the off chance that present quality is superior to gbest, then reset gbest to the present particle's array record and values.
5) Modifies the velocity and position of the particle according to equations (2.1) and *(2.2)*, respectively:

$$V_i^{k+1} = W*V_i^k + C_p * \mathbf{R_p} * (Xpbest_i - X_i^k) + C_g * \mathbf{R_g} * (Xgbest - X_i^k) \quad \ldots..(2.1)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \qquad \text{…..(2.2)}$$

6) Circle to step 2 until a criterion is met, usually a sufficiently best fitness or a most extreme number of iterations reaches.

Particles' velocities on every single dimension are clipped to a most extreme velocity *Vmax*. On the off chance that the sum of accelerations would cause the velocity on that dimension to surpass *Vmax*, which is a parameter specified by the user, then the velocity on that dimension is constrained to *Vmax*.

Vmax is hence a vital parameter. It determines the resolution, or fineness, with which regions between the current position and the objective (best so far) position are searched. On the off chance that *Vmax* is too high, particles may fly past great solutions. On the off chance that *Vmax* is too small, then again, particles may not investigate sufficiently past generally great regions. Truth be told, they could get to be caught in local optima, not able to move sufficiently far to achieve a superior position in the problem space.

The acceleration constants Cp and Cg in mathematical eqn. (2.1) represent the weighting of the stochastic acceleration terms that force every particle toward pbest and gbest positions. Thus, adjustment of these constants modifies the measure of "tension" in the system. Low values permit particles to meander a long way from target regions before being pulled back, while high values result in sudden development toward, or past, target regions. Early involvement with particle swarm optimization (experimentation, mostly) drove us to set the acceleration speed constants Cp and Cg every equivalent to 2.0 for almost all applications. Vmax was thus the main parameter we routinely adjusted, and we regularly set it at about 10-20% of the dynamic scope of the variable on every dimension.

Based, in addition to other things, on findings from social simulations, it was chosen to design a "local" version of the particle swarm. In this version, particles have data just they could call their own and their neighbours bests, instead of that of the entire group. Instead of moving toward a sort of stochastic average of pbest and gbest (the best area of the entire group), particles move toward points characterized by pbest and "lbest", which is the list of the particle with the best evaluation in the particle's neighbourhood.

On the off chance that the area size is characterized as two, for instance, particle(i) compares its fitness esteem with particle(i-1) and particle(i+l). Neighbors are

characterized as topological neighbors; neighbors and neighbourhoods don't change in a run. For the neighbourhood version, the main change to the process characterized in the six steps before is the substitution of pld, the area of the neighbourhood best, for X_{gbest}, the global best, in equation (2.1) . Early experience (once more, for the most part experimentation) prompted neighbourhood sizes of around 15 percent of the populace size being used for some applications. So, for a populace of 40 particles, an area of six, or three topological neighbours on every side, was not unusual.

The populace size selected was issue problem dependent. Populace sizes of 20-60 were presumably most common. It was found out from the get-go that smaller populations than were normal for other evolutionary algorithms (such as hereditary algorithms and evolutionary writing computer programs) were optimal for PSO in terms of minimizing the aggregate number of evaluations (populace size times the number of generations) expected to acquire a sufficient solution.

2.4 PSO ALGORITHM PARAMETERS

There are some parameters in PSO calculation that may influence its execution. For any given optimization problem, some of these parameter's values and choices have substantial effect on the efficiency of the PSO method, and different parameters have small or no impact. The distinctive PSO parameters are number of particles or swarm size, velocity components, acceleration coefficients and number of iterations illustrated beneath.

2.4.1 Swarm size

Populace size or swarm size is the quantity of particles "s" in the swarm. Countless number of size generates bigger parts of the search space to be secured per iteration. Countless number of particles may decrease the quantity of iterations need to get a superior optimization result. In contrast, large amounts of particles improve the computational complexity per iteration, and additional lengthy. From an expansive number of exact studies, **it has been shown that most of the implementations in PSO use an interval of s ε [10,60] for the size of swarm.**

2.4.2 Velocity clamping

Starting PSO studies used Cp = Cg= 2.0. Although great results have been accomplished, it was seen that velocities fastly blasted to substantial values, especially for particles at an extensive distance from their global best (^y) and personal best (y_i) positions. Consequently, particles have vast no. of position updates, with particles abandoning the boundaries of their search space. Velocities are braced to control the increase in velocity.

$v_i^{(k+1)} = v'^{(k+1)}_i$ if $v'^{(k+1)}_i < V_{max}$

V_{max} if $v'^{(t+1)}_{ij} \geq V_{max}$

Velocity clamping does not stay away from a particle from leaving the boundaries of its search space, it limits the particle step sizes, in this way different behavior is restricted.

2.4.3 Iteration numbers

The no. of iterations to acquire a best result depends on the problem. A low number of iterations may end the search process rashly, while a substantial no. of iterations have the consequence of unnecessary included computational complex nature and make the convergence slow.

2.3.4 Acceleration coefficients

The acceleration coefficients Cp and Cg, joined with the arbitrary values Rp and Rg, keep up the stochastic impact on the cognitive and social segments of the particle's velocity separately. The steady Cp demonstrates the amount of certainty a particle has on itself, while Cg communicates the amount of certainty a particle has on its neighbors. There are a few properties of Cp and Cg.

Cp = Cg = 0 speaks to all particles keep flying at their present velocity until they hit the limit of the search space. Subsequently, the redesign mathematical statement for velocity is ascertained as

$V_i(k+1) = V_i(k)$

$C_p > 0$ and $C_g = 0$ represents that all particles are autonomous. The velocity update mathematical statement for this condition will be

$$V_i(k+1) = w * V_i(k) + C_p * R_{p*} (y_i(k) – X_i(k)) \quad \text{……. (2.3)}$$

Unexpectedly, Cp > 0 and Cg = 0 represents that all particles are pulled in towards a solitary point in the whole swarm and the redesign in the velocity will be as under:

$$V_i(k+1) = V_i(k) + C_p * R_{p*} (\hat{y}_i(k) – X_i (k)) \quad \text{..…… (2.4)}$$

Cp = Cg represents that every one of the particles are pulled in towards the average of pbest and gbest .

Cp >> Cg represents that every particle is emphatically impacted by its own best position, bringing about an increased wandering. Interestingly, when Cg >> Cp then the majority of the particles are a great deal more affected by the global best position, which causes all particles to converge rashly to the optima.

Ordinarily, Cp and Cg are static, with exactly discovering the optimized values. Wrong initialization of Cp and Cg may result in cyclic or dissimilar conduct. From the diverse experimental investigates, this has been suggested that the two acceleration constants must be Cp = Cg =2.

2.4.5 Inertia weight

Shi and Eberhart[41] presented the inertia weight with take out the requirement for velocity clamping and to still limit the dissimilar behaviour. The force of the particles is controlled by the inertia weight (w) by measuring the commitment of the past velocities, basically it is utilized to control the amount of memory of the last flight bearing will influence the new velocity. The velocity mathematical statement altered to

$$V_i(k+1) = w * V_i(k) + C_p * R_p * (y_i(k) – X_i(k)) + C_g * R_g * (\hat{y}_i(k) – X_i(k)) \quad \text{....(2.5)}$$

Shi and Eberhart presented the idea of inertia weight in 1999 to decrease the velocities after some time (or iterations), to control the misuse and investigation capacities of the swarm and to unite the swarm all the more productively and precisely.

When w≥ 1 then the velocities increment with time and particles can scarcely redirect their headings to return towards ideal, and the swarm diverges. When w ≤ 1 then little force is just spared from the introductory step and speedy changes to directions are situated simultaneously. When w =0 particles velocity vanishes and all particles move without learning of the last velocity in every step.

The inertia weight may be actualized either as rapidly changing values or an fixed values. Beginning usage of utilized a fixed value for the entire procedure for all particles, however now progressively changing inertia values is utilized in light of the fact that this parameter controls the exploration and exploitation of the search space. Different methodologies were proposed time to time were mulled over in detail.

The inertia value is typically high at first, which permits all particles to freely move in the search space in the introductory steps and decreases with time. In this way, the procedure shifts from the exploratory mode to the exploitative mode. This diminishing in inertia weight has delivered great results in a large portion of optimization problems. To control the harmony in the middle of local and global exploration and to acquire speedy joining and to achieve an ideal, the inertia weight whose value decreases directly with the increment in iteration number is situated in like manner by the accompanying mathematical statement.

$$w(i+1) = w_{max} - (((w_{max} - w_{min}) * k) / (it_{max})) , \quad w_{max} > w_{min} \qquad (2.6)$$

where, w_{min} and w_{max} are the final and initial values of the inertia weight respectively, it_{max} is the maximum iteration no., 'k' is the current iteration number. Regularly, the inertia weight decreases straightly from 0.9 to 0.4 over the full run. Trelea have characterized a condition that (w < ((Cp + Cg)/2)- 1) ensures the convergence. Cyclic or Divergent conduct can happen in the process if this condition is not fulfilled.

The method of inertia weight is truly valuable to guarantee convergence. However there is a disservice of the inertia weight system that once the inertia weight is diminished, it can't build regardless of the possibility that the swarm needs to look new zones. This technique is not ready to recover its exploration mode.

2.4.6 Neighbourhood Topologies

A neighbourhood must be characterized for every particle. This neighbourhood decides the degree of social connection inside of the swarm and impacts a specific particle's development. Less cooperation happens when the neighbourhoods in the swarm are little. For little neighborhood, the meeting will be slower however it may enhance the nature of arrangements. For bigger neighborhood, the merging will be speedier yet the risk that occasionally meeting happens prior. To take care of this issue, the search procedure begins with little neighborhoods size and after that the little neighborhoods size is expanded over the long run. This procedure guarantees an at first high diversity with speedier convergence as the particles move towards a promising pursuit region.

The PSO algorithm is social collaboration among the particles in the whole swarm. Particles correspond with each other by trading data about the achievement of every particle in the swarm. At the point when a particle in the entire swarm discovers a best position, all particles move towards this particle. This execution of the particles is dictated by the particle's neighborhood. Researchers have dealt with adding to this execution by planning distinctive sorts of neighborhood structures. Some area structures or topologies are talked about underneath:

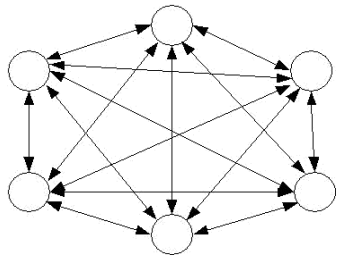

Figure 1 Star or gbest

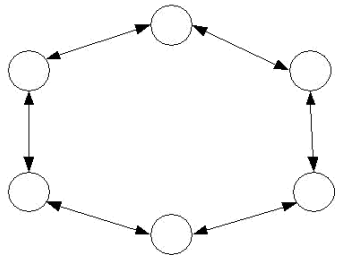

Figure 2 Ring or lbest

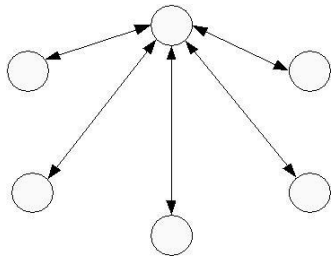

Figure 2.3 Wheel

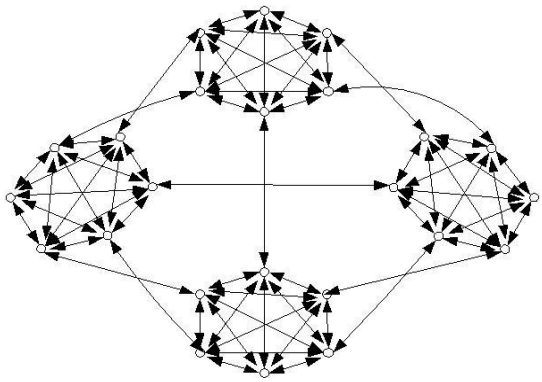

Figure 2.4 Four Clusters

Figure 1 clarifies the star topology, where every particle is associated with each other particle. This topology fundamentally prompts meeting at a quicker rate than different topologies, yet there is an opportunity to be caught in local minima. Since all particles are mindful about one another, this topology is known as the gbest PSO.

Figure 2 shows the ring topology, where every particle is related to its quick neighbors. In this specific procedure, when better result is found by one particle then this molecule offers it to its quick neighbors and these quick neighbors offers it to their individual quick neighbors, until it picked up by the last particle. In this manner the best result found is spreaded gradually in a ring made by all particles. Convergence is moderate, yet an awesome piece of the inquiry space is secured than with the star topology. It is known as the lbest PSO.

Figure 3 demonstrates the wheel topology, in this stand out of the particle (a focal particle) partners to the others, and all information are imparted through this specific particle. This focal particle thinks about the best execution of all the particles in the swarm and changes its own position towards the best execution molecule break down without anyone else's input lastly the new position of the central particle is imparted to every one of the particles.

Figure 4 demonstrates four groups topology, where four coteries (or bunches) are associated with one edge between inverse groups and two edges between neighboring bunches. There are more diverse topologies or neighborhood structures (for occurrence, Von Neumann topology, the pyramid topology etc), yet there is no single best topology still known not the required optimum for all varieties of optimization problems.

2.5 MATHEMATICAL FORMULAS:

The modification of the particle's position can be numerically demonstrated according the following mathematical equation:

$$\mathbf{V_i^{k+1} = W*V_i^k + C_p*R_p*(Xpbest_i - X_i^k) + C_g*R_g*(Xgbest - X_i^k)} \quad \mathbf{\ldots(2.1)}$$

Where;

V_i^k : velocity of agent i at iteration k,

W : inertia weight factor,

C : weighting factor,

R :uniformly distributed random number between 0 and 1,

X_i^k : present position of agent i at iteration k,

Xpbest :best of agent i,

Xgbest : gbest of the group.

$V_i^{(k)}$ which is the velocity of i^{th} particle at iteration 'k' must lie in the range

$$V_{min} \leq V_i(k) \leq V_{max}$$

In the equation (1)

The term R_p *($Xpbest_i - X_i^k$) is called particle memory influence,

The term R_g *(Xgbest-X_i^k) is called swarm influence.

The Position of the Particles is updated by the following equation:

$$\mathbf{X_i^{k+1} = X_i^k + V_i^{k+1}} \qquad \text{……(2.2)}$$

The following inertia weight factor is usually utilized in (1)

$$\mathbf{W = Wmax-[(Wmax-Wmin) * k]/maxIter} \qquad \text{…….(2.7)}$$

2.6 FLOWCHART:

The general computational system for the Particle Swarm Optimization is as per the following:

1. Before to the iteration begins, initialize the particles with random position and velocity vectors.
2. For each of the particle's position (X_i^k) compute the value of objective function (F: we likewise call it fitness function).
3. When F(X_i^k) is not exactly (X_{pbesti}), then allocate the value of X_{pbesti} as X_i^k (do it for every one of the particles).
4. Focus the best value of X_{pbesti} consider its fittest value. In the event that f(best of X_{pbesti})is not exactly f(X_{gbesti}), than allot the value of best of X_{pbesti} to the X_{gbesti}.
5. Calculate the new velocity vector utilizing mathematical equation (2.1) and the new position vector utilizing mathematical equation (2.2).
6. Repeats the circle until either the stopping criteria met or the maximum iteration is achieved.
7. After iteration finishes, give Xgbesti as the optimal solution and the fitness relating to it as the optimum value.

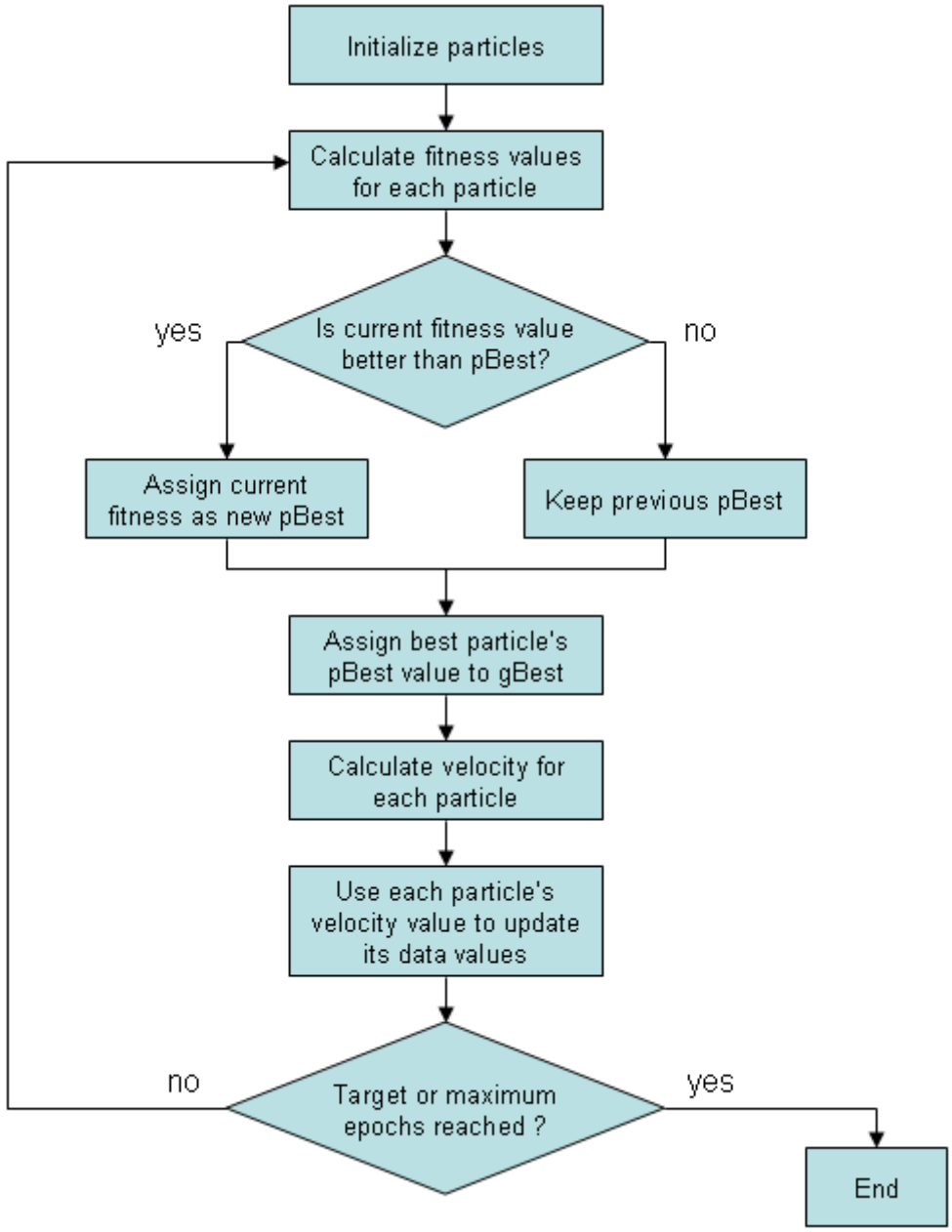

Fig 5 BPSO flowchart

2.6.1 Steps involved in particle swarm optimization in MATLAB

1 Using the zero command of MATLAB initialize all the variable matrices.

2 Set the values of the random no.'s 'Rp' & 'Rg' assigned to the personal and global best expressions respectively.

3 Set the values of acceleration constants ‘Cp’ & ‘Cg’ assigned to the personal and global best expressions respectively.

4 Set the tolerance value.

5 Generate the random position values of particles for all the variables (eg. X_1, X_2) and also generate the random velocities values of particles (eg. V_1, V_2) for all the variable.

6 Calculate the fitness for the assumed values of the positions of the particles.

7 Using the above fitness personal best and global best values for all variables are deduced.

8 Using the previous iteration values of personal best, global best and velocity vectors new velocities are generated in the current iteration using the equation:

$$\mathbf{V_i^{k+1} = W*V_i^k + C_p*R_p*(Xpbest_i - X_i^k) + C_g*R_g*(Xgbest - X_i^k)}$$

9 Using the new velocity vector and the old position vector, a new position vector is generated for all the variables.

10 Calculate fitness using the new positions in the current iteration.

11 Using the new fitness values the personal and global best values are updated.

12 The difference between the past and the present fitness is calculated and check against the tolerance value, if inside of the tolerance iteration stops and global best value is the solution else iteration stream does a reversal to past step for further updating.

3.7 ADVANTAGES AND DISADVANTAGES OF PSO

A PSO is considered as a standout amongst the most intense strategies for determining the non-smooth global optimization problems and has numerous focal points when contrasted with other heuristic optimization techniques, which are as take follow:

- PSO is a derivative-free technique just like as other heuristic optimization techniques.

- PSO is easy in its concept and coding implementation.

- PSO is less sensitivity to the nature of the objective function compared to the conventional mathematical approaches and other heuristic methods.

- PSO has limited number of parameters including only inertia weight factor and two acceleration coefficients.

- PSO seems to be somewhat less dependent of a set of initial points compared to other evolutionary methods, implying that convergence algorithm is robust.

- PSO techniques can generate high-quality solutions within shorter calculation time and stable convergence characteristics.

The major *drawback* of PSO, like in other heuristic optimization techniques, is that it lacks a solid mathematical foundation for analysis to be overcome in the future development of relevant theories. Also, it can have some limitations for real-time ED applications such as 5-minute dispatch considering network constraints since the PSO is also a variant of stochastic optimization techniques requires relatively a longer computation time than mathematical approaches. It has the problems of dependency on initial point and parameters, difficulty in finding their optimal design parameters, and the stochastic characteristic of the final outputs.

CHAPTER-3

SELECTION BASED PSO AND ITS APPLICATION TO MATHEMATICAL BENCHMARK TEST FUNCTIONS

3.1 SELECTION BASED PSO:

Basic PSO is an optimization technique which is used to get an optimize value for any function. PSO is a modern heuristic optimization method which is the best optimization technique known so far. In this thesis PSO is modified as a Selection Based PSO (SBPSO).

In Selection Based PSO or SBPSO, modification has been done in a basic PSO method. The selection procedure in a basic PSO of size of particles is random. Size of particles is specified in an algorithm and these particles remain same for the complete problem till the optimization has been reached. E.g. if in an optimization problem size of particles have been taken 20 to optimize the problem, then for the complete optimization problem size of particles will remain the same as 20 and all the particles will get optimize to a best possible value. In SBPSO, selection procedure of size of particles has been changed. The size of particles is specified before any optimization problem at random in the 0^{th} iteration. In SBPSO the size of particles is decreased in each iteration by some decrement factor. In the 1^{st} iteration the size of particle is same as initial size and in subsequent iteration the size of particles decreases by some decrement factor. The size of particle decrease to the '**t**' times of value in every iteration. For example if the size of particles is taken as 160 and the factor by which it decrease is 2. So in each iteration size of the particles will decrease 2 times of the size of particles in previous iteration. In the 1^{st} iteration there will be 160 particles, in the 2^{nd} iteration it will decrease 2 times and becomes 80, in 3^{rd} iteration it will become 40 and so on.

This decrement of particles in each iteration is being done in a way that particle for which the value of function is less will be selected and the particles with higher function values will get discarded. So, in each iteration the size of particles changes by the factor 't'. The particles are sorted according to their function value and the size

found by the factor ‘t’ will select the top sorted particles (based on minimum function value) for the next iteration and this procedure will be repeated in each iteration.

In SBPSO, the particle size goes on decreasing in each iteration by p/t will either become a fraction or less than the minimum number of particles required to optimize a function. Therefore, the ‘q’ has been fixed to minimum number of particles required to optimize a function and for subsequent iteration the size of particles will be forced to ‘q’. For SBPSO, let

p= total size of particles in 0^{th} iteration.

t= decrement factor.

q= minimum no. of particles to be selected.

n= integer numbers.

From the conclusion one formula is determined so by which the value of kount can be calculated. This formula helps in calculating the value of kount.

$$\mathbf{kount} = \left(\mathbf{2} * \mathbf{p} + \sum_{\boldsymbol{n=1}}^{\boldsymbol{k}} \frac{\boldsymbol{p}}{\boldsymbol{t^n}}\right) + \mathbf{q} * (\mathbf{i} - \mathbf{n} - \mathbf{1}),\ \frac{\boldsymbol{p}}{\boldsymbol{t^n}} \geq \mathbf{q} \qquad \textbf{…… 3.1}$$

where n=1,2,3,4…..k.

i = number of iteration in which problem converge.

With the values of all the variables in equations p, q, t, n and i the value of kount will be calculated. This calculated value is similar to the value observed. All the result obtained are satisfying this formula. All the kount values obtained are similar with the values calculated by this kount formula.

3.2 STEPS FOR THE PROCEDURE OF SBPSO

The sequence for the solution of function by SBPSO is explained as follows:

1. Fix the size of particles ‘p’ in swarm and set the no. of maximum iterations it_{max} and tolerance value T.
2. Fix algorithm constants Cp, Cg, Rp, Rg, w.
3. Fix the factor of decrement ‘t’ and a last size of particle is to be chosen say ‘q’.
4. Generate X_i^k and V_i^k, the initial random positions (i.e. generations) and velocity (i.e. updation factor) respectively.
5. Set iteration kount = 0.
6. At 0^{th} iteration the personal and global best positions (i.e. generations) are same as the initial random positions (i.e. generations).
7. Calculate the function for each particle.
8. Increase the iteration kount by 1 using kount = kount + 1 in the every evaluation of function.
9. Calculate the velocity (i.e. positions updating factor) of each particle using eq.(2.1).
10. Calculate the new positions (i.e. X) of the particles by evaluating eq.(2.2).
11. Sort the particles with accordance to their function values in increasing order.
12. Size of the particles is updated by the factor p/t say new ‘P’, and the size of particles updated is ‘P’.
13. Save new P equal to p.
14. Select the top ‘p’ number of particles from the sorted particles with accordance to their function values in increasing order.
15. If p < q, make ‘p’ equal to ‘q’ and initialize the value of ‘t’ equals to 1 and go to next step.
16. Calculate ELD function for the new size of particles (P or p) and new positions that has generated.
17. Update Xpbest and Xgbest values by comparing ELD function values.
18. Check if the stopping criteria are satisfied, if not then go to step 7, else stop.
19. Get output of function and variable value.

3.3 FLOWCHART OF SBPSO

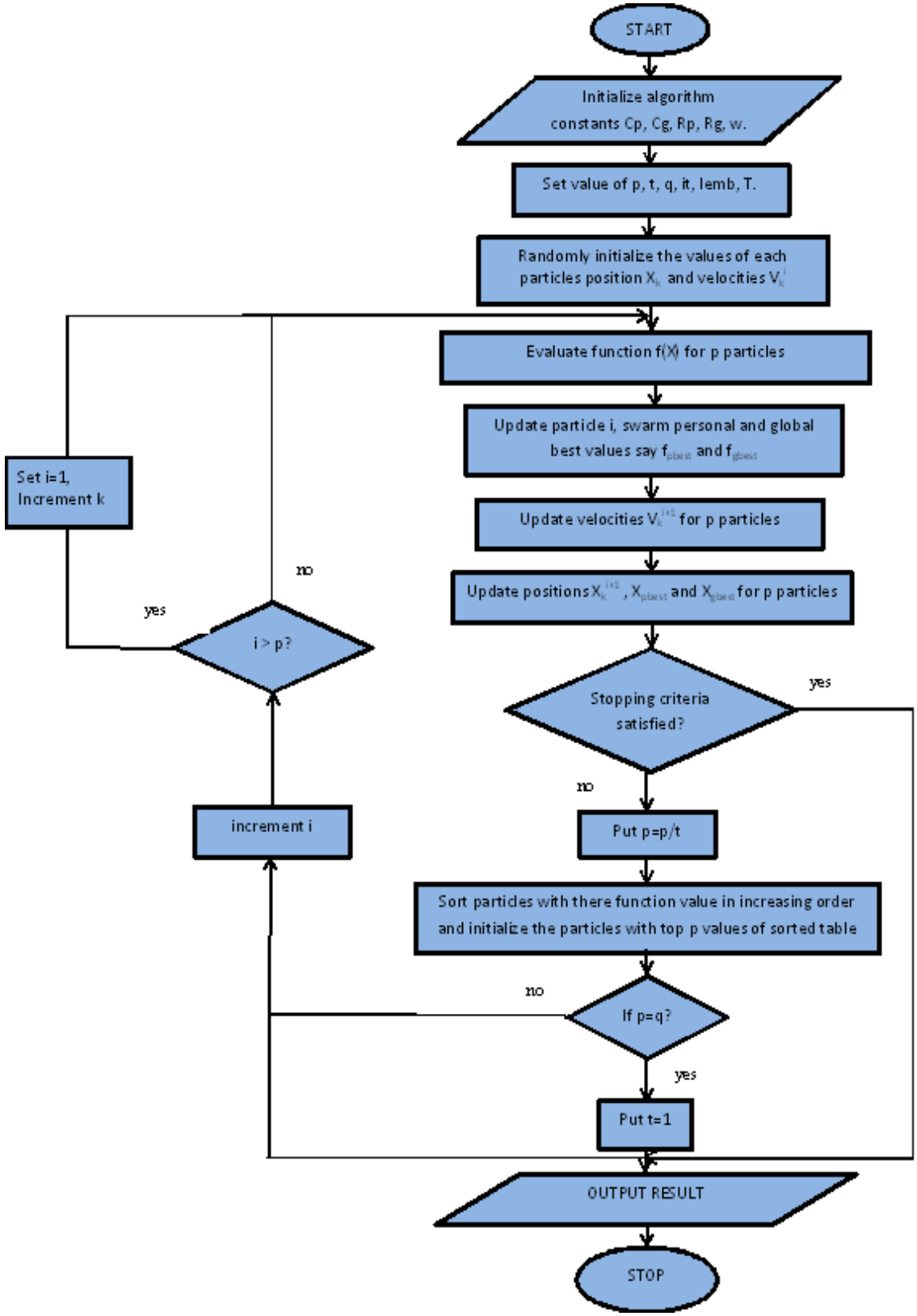

Fig 6 SBPSO evaluation flowchart

3.4 MATHEMATICAL BENCHMARK FUNCTIONS:

Artificial landscapes the second name given to the **Test functions**, are extremely helpful to assess characteristics of optimization algorithms. For this situation of application of Particle Swarm Optimization to the mathematical benchmark functions, the PSO algorithm can be connected specifically to the specific mathematical function, i.e. with no change. As the mathematical functions are single objective functions and no uniformity criteria on the fitness functions values, no further plan for objective function is obliged and the inequality constraints on the variables, if present, are dealt with in the PSO algorithm itself. Utilizing particle swarm optimization the essential steps for taking care of the optimization problem is same as talked about before yet in the event that a few adjustments are given then we can utilize it for any kind of objective function. Here, we have utilized PSO for the optimization of some mathematical benchmark functions, which are as per the following:

- Rosenbrock's function

$$f(X_1,X_2)=100*(X_2- X_1^2)^2+(X_1-1)^2 \quad \text{.....(3.2)}$$

- Booth's function

$$f(X_1,X_2) = (X_1 + 2 * X_2 - 7)^2 + (2 * X_1 + X_2 - 5)^2 \quad \text{.....(3.3)}$$

- Beale's function

$$f(X_1,X_2) =(1.5-X_1+X_1*X_2)^2 + (2.25-X_1+X_1*X_2^2)^2+ (2.625-X_1+X_1*X_2^3)^2 \quad \text{.....(3.4)}$$

- Sphere's function

$$f(X_1, X_2, X_3)= X_1^2 + X_2^2 + X_3^2 \quad \text{.....(3.5)}$$

- Rastrigin's function

$$f(X_1,X_2) = 20 + (X_1^2) - (10 * \cos (2*pi*X_1)) + (X_2^2) - (10 * \cos (2*pi*X_2)) \quad \text{..... (3.6)}$$

3.5 VALUE OF PARAMETERS USED IN SELECTION BASED PARTICLE SWARM OPTIMIZATION (SBPSO)

The various parameters of particle swarm optimization are as follows:

1. No. of particles in the swarm, **p**.
2. Max. no. of iteration, **it**.
3. Decrement factor, **t**.
4. Minimum no. of particles to be selected, **q**.
5. Random no. for personal and global factors $\mathbf{R_p}$ and $\mathbf{R_g}$.
6. Acceleration constant for the personal and global factors, $\mathbf{C_p}$ and $\mathbf{C_g}$.
7. Tolerance value, **T**.
8. Inertia weight, **w.**

The values of these parameters for optimizing various mathematical benchmark functions were chosen as:

1. it= 1000
2. R_p=0.5 and R_g=0.6
3. C_p=2 and C_g=2
4. T= 10
5. w=0.6.

3.6 COMPUTATIONAL RESULTS

Various benchmark functions and results obtained after application of PSO has been discussed as follows:

3.6.1 Rosenbrock Function

$$f(x) = \sum_{k=1}^{n-1} [100(X_{k+1} - X_k^{\,2})^2 + (1 - X_k)^2]$$

where , $-2.048 < X_k < 2.048$, k = 1,2,3....,n;

For two variables X_1 and X_2

$$F(X_1,X_2)=100*(X_2- X_1^2)^2+(X_1-1)^2$$

Minimum value and range for the function are as follows:

- Min. Value: $f(X_1,X_2)=0$
- Range : $-2.048 < X_1,X_2 < 2.048$
- Optimal values of $X_1 = X_2 = 0$

Table 3.1 shows the application of SBPSO for the Rosenbrock function. Column (1) of this table shows the numbers of particles 'p'. Column (2) shows decrement factor 't'. Column (3) of this table shows 'q' which is minimum numbers of particles required to optimize a function. Column (4), (5) and (6) show the function value, kount and number of iterations respectively.

S.No. 1 of this table shows the result of Basic PSO (BPSO) for minimum particle size of '10' required to optimize the Rosenbrock function. The remaining rows show the result of SBPSO.

For S.No. 2 to 11, the decrement factor 't' is fixed to 2 for various values of initial particle size varying from 20 to 640 in multiples of 2. It is observed that for particle size 20 to 160, the number of iterations and the value of kount is less compared to BPSO. It is observed that by taking t=2 (decrement factor) for a minimum value of 10 particles (i.e. q=10), the kount is being decreased in the range of 20 to 160 particles (i.e. 'p'). There is a gradual decrease in iteration in the same range. It is observed that by taking t=2 (decrement factor) for a minimum value of 5 particles (i.e. q=5), the kount is being decreased in the range of 20 to 320 particles (i.e. 'p'). Although the function value is not as accurate as that in basic PSO but it is optimum. There is a gradual decrease in iteration in the same range.

For a minimum value of 10 particles (q=10), for initial size of 80, 160, 320 and 640 on increasing decrement factor to 4 & 8, kounts decreases for 80 & 160 particles compared to the result shown at S.No. 1 for Basic PSO (BPSO). It is to be noted that BPSO didn't converge when particle size is chosen to '5'. Similarly, for a minimum value of 5 particles (q=5), for initial size of 80, 160 and 320 on increasing decrement factor to 4 &8 , kounts decreases for all size of particles and iterations also decreases.

Table 3.1 Results of SBPSO to Rosenbrock's Function by varying p, t and q

	1	2	3	4	5	6
S.No.	**p**	**t**	**q**	**F**	**Kount**	**Iteration**
1	10	1	10	-19	1210	121
2	20	2	10	-16	1150	112
3	20	2	5	-2	500	92
4	40	2	10	-14	1050	97
5	40	2	5	-7	550	91
6	80	2	10	-21	1210	102
7	80	2	5	-7	675	93
8	160	2	10	-15	1110	69
9	160	2	5	-11	885	88
10	320	2	10	-18	1770	88
11	320	2	5	-12	1270	70
12	80	4	10	-22	1200	104
13	80	4	5	-5	630	92
14	160	4	10	-16	1170	83
15	160	4	5	-7	805	90
16	320	4	10	-21	1640	93
17	320	4	5	-12	1100	75
18	160	8	10	-7	1130	81
19	160	8	5	-4	930	80
20	320	8	10	-21	1530	87
21	320	8	5	-7	1080	82
22	640	8	10	-16	2390	105
23	640	8	5	-12	1725	74

Table 3.2 Verification of kount formula by varying p, t and q

	1	2	3	4	5	6	7
S.No.	P	t	q	i	n	Count observed	Count calculated
1.	10	1	10	121	10	1220	1220
2.	20	2	10	112	2	1150	1150
3.	20	2	5	92	3	500	500
4.	40	2	10	97	3	1050	1050
5.	40	2	5	91	4	550	550
6.	80	2	10	102	4	1210	1210
7.	80	2	5	93	5	675	675
8.	160	2	10	69	5	1110	1110
9.	160	2	5	88	6	885	885
10.	320	2	10	88	6	1770	1770
11.	320	2	5	70	7	1270	1270
12.	80	4	10	104	2	1200	1200
13.	80	4	5	92	3	630	630
14.	160	4	10	83	3	1170	1170
15.	160	4	5	90	3	805	805
16.	320	4	10	93	3	1640	1640
17.	320	4	5	75	4	1100	1100
18.	160	8	10	81	2	1130	1130
19.	160	8	5	80	2	930	930
20.	320	8	10	87	2	1530	1530
21.	320	8	5	82	3	1080	1080
22.	640	8	10	105	3	2390	2390
23.	640	8	5	74	3	1725	1725

Table 3.2 shows the mathematically calculated value of kount by the formula in the equation (3.1) for Rosenbrock function. Column (1) of this table shows the numbers of particles ‘p’. Column (2) shows decrement factor ‘t’. Column (3) of this table shows ‘q’ which is minimum numbers of particles required to optimize a function. Column (4), shows ‘i’ which is the value of iteration in which function converge. Column (5) shows ‘n’ which is an integer number for which the series is computing. Column (6) show the obtained kount value from the PSO function evaluation and column (7) shows the kount value calculated from equation (3.1).

Both the calculated kount value and observed kount value are same for all computed arrangement. Same value for both kount verifies that the formula observed from the computed result of rosenebrock function. The equation (3.1) is the verified formula for the calculation of kount value for all the mathematical benchmark function taken in this thesis.

3.6.2 Booth's Function

$$F(X_1,X_2) = (X_1 + 2 * X_2 - 7)^2 + (2 * X_1 + X_2 - 5)^2$$

Minimum value and range for the function are as follows:

- Optimal values of $X_1=1$, $X_2=3$.

Table 3.3 shows the application of SBPSO for the Booth's function. Column (1) of this table shows the numbers of particles 'p'. Column (2) shows decrement factor 't'. Column (3) of this table shows 'q' which is minimum numbers of particles required to optimize a function. Column (4), (5) and (6) show the function value, kount and number of iterations respectively.

S.No. 1 of this table shows the result of Basic PSO (BPSO) for minimum particle size of '10' required to optimize the Booth's function. The remaining rows show the result of SBPSO.

For S.No. 2 to 11, the decrement factor 't' is fixed to 2 for various values of initial particle size varying from 20 to 640 in multiples of 2. It is observed that for particle size 20 to 160, the number of iterations and the value of kount is less compared to BPSO. It is observed that by taking t=2 (decrement factor) for a minimum value of 10 particles (i.e. q=10), the kount is being decreased in the range of 20 to 160 particles (i.e. 'p'). There is a gradual decrease in iteration in the same range. It is observed that by taking t=2 (decrement factor) for a minimum value of 5 particles (i.e. q=5), the kount is being decreased in the range of 20 to 320 particles (i.e. 'p'). Although the function value is not as accurate as that in basic PSO but it is optimum. There is a gradual decrease in iteration in the same range. For a minimum value of 10 particles (q=10), for initial size of 80, 160, 320 and 640 on increasing decrement factor to 4 & 8, kounts decreases for 80 & 160 particles compared to the result shown at S.No. 1 for Basic PSO(BPSO). It is to be noted that BPSO didn't converge when particle size is chosen to '5'. Similarly, for a minimum value of 5 particles (q=5), for initial size of 80, 160, 320 and 640 on increasing decrement factor to 4 &8 , kounts decreases for all size of particles and iterations also decreases.

Table 3.3 Results of SBPSO to Booth's Function by varying p, t and q

	1	2	3	4	5	6
S.No.	**p**	**t**	**q**	**F**	**Kount**	**Iteration**
1	10	1	10	-19	1300	129
2	20	2	10	-17	1020	99
3	20	2	5	-12	505	93
4	40	2	10	-19	950	87
5	40	2	5	-12	495	80
6	80	2	10	-18	1130	94
7	80	2	5	-11	640	86
8	160	2	10	-18	1280	86
9	160	2	5	-10	855	82
10	320	2	10	-19	1740	85
11	320	2	5	-13	1205	57
12	80	4	10	-15	1130	97
13	80	4	5	-8	695	105
14	160	4	10	-14	1230	89
15	160	4	5	-8	755	80
16	320	4	10	-16	1540	88
17	320	4	5	-11	1110	79
18	160	8	10	-16	1180	86
19	160	8	5	-5	870	108
20	320	8	10	-18	1480	82
21	320	8	5	-16	1070	80
22	640	8	10	-12	2220	88
23	640	8	5	-4	1835	96

3.6.3 Beale's Function

$$F(X_1,X_2) = (1.5-X_1+X_1*X_2)^2 + (2.25-X_1+X_1*X_2^2)^2 + (2.625-X_1+X_1*X_2^3)^2$$

Minimum value and range for the function are as follows:

- Optimal values of $X_1=3, X_2=0.5$

Table 3.4 shows the application of SBPSO for the Beale's function. Column (1)of this table shows the numbers of particles 'p'. Column (2) shows decrement factor 't'. Column (3) of this table shows 'q' which is minimum numbers of particles required to optimize a function. Column (4), (5) and (6) show the function value, kount and number of iterations respectively.

S.No. 1 of this table shows the result of Basic PSO (BPSO) for minimum particle size of '10' required to optimize the Beale's function. The remaining rows show the result of SBPSO.

For S.No. 2 to 11, the decrement factor 't' is fixed to 2 for various values of initial particle size varying from 20 to 640 in multiples of 2. It is observed that for particle size 20 to 160, the number of iterations and the value of kount is less compared to BPSO. It is observed that by taking t=2 (decrement factor) for a minimum value of 10 particles (i.e. q=10), the kount is being decreased in the range of 20 to 160 particles (i.e. 'p'). There is a gradual decrease in iteration in the same range. It is observed that by taking t=2 (decrement factor) for a minimum value of 5 particles (i.e. q=5), the kount is being decreased in the range of 20 to 320 particles (i.e. 'p'). Although the function value is not as accurate as that in basic PSO but it is optimum. There is a gradual decrease in iteration in the same range. For a minimum value of 10 particles (q=10), for initial size of 80, 160, 320 and 640 on increasing decrement factor to 4 & 8, kounts decreases for 80 & 160 particles compared to the result shown at S.No. 1 for Basic PSO(BPSO). It is to be noted that BPSO didn't converge when particle size is chosen to '5'. Similarly, for a minimum value of 5 particles (q=5), for initial size of 80, 160, 320 and 640 on increasing decrement factor to 4 &8 , kounts decreases for all size of particles and iterations also decreases.

Table 3.4 Results of SBPSO to Beale's Function by varying p, t and q

	1	2	3	4	5	6
S.No.	p	t	q	F	Kount	Iteration
1	10	1	10	-12	1250	124
2	20	2	10	-10	980	94
3	20	2	5	-9	450	81
4	40	2	10	-12	1090	100
5	40	2	5	-17	490	75
6	80	2	10	-10	1150	95
7	80	2	5	-8	655	88
8	160	2	10	-19	1300	87
9	160	2	5	-9	865	83
10	320	2	10	-11	1860	96
11	320	2	5	-10	1270	69
12	80	4	10	-20	940	77
13	80	4	5	-10	575	80
14	160	4	10	-20	1210	86
15	160	4	5	-7	775	83
16	320	4	10	-10	1760	104
17	320	4	5	-8	1100	74
18	160	8	10	-15	1190	86
19	160	8	5	-10	740	81
20	320	8	10	-14	1530	86
21	320	8	5	-14	485	62
22	640	8	10	-18	2240	89
23	640	8	5	-15	1790	86

3.6.4 Sphere's Function

$$F(X_1, X_2, X_3) = X_1^2 + X_2^2 + X_3^2$$

Minimum value and range for the function are as follows:

- Optimal values of $X_1=0, X_2=0, X_3=0$

Table 3.5 shows the application of SBPSO for the Sphere's function. Column (1)of this table shows the numbers of particles 'p'. Column (2) shows decrement factor 't'. Column (3) of this table shows 'q' which is minimum numbers of particles required to optimize a function. Column (4), (5) and (6) show the function value, kount and number of iterations respectively.

S.No. 1 of this table shows the result of Basic PSO (BPSO) for minimum particle size of '10' required to optimize the Sphere's function. The remaining rows show the result of SBPSO.

For S.No. 2 to 11, the decrement factor 't' is fixed to 2 for various values of initial particle size varying from 20 to 640 in multiples of 2. It is observed that for particle size 20 to 160, the number of iterations and the value of kount is less compared to BPSO. It is observed that by taking t=2 (decrement factor) for a minimum value of 10 particles (i.e. q=10), the kount is being decreased in the range of 20 to 160 particles (i.e. 'p'). There is a gradual decrease in iteration in the same range. It is observed that by taking t=2 (decrement factor) for a minimum value of 5 particles (i.e. q=5), the kount is being decreased in the range of 20 to 320 particles (i.e. 'p'). Although the function value is not as accurate as that in basic PSO but it is optimum. There is a gradual decrease in iteration in the same range. For a minimum value of 10 particles (q=10), for initial size of 80, 160, 320 and 640 on increasing decrement factor to 4 & 8, kounts decreases for 80 & 160 particles compared to the result shown at S.No. 1 for Basic PSO(BPSO). It is to be noted that BPSO didn't converge when particle size is chosen to '5'. Similarly, for a minimum value of 5 particles (q=5), for initial size of 80, 160, 320 and 640 on increasing decrement factor to 4 &8 , kounts decreases for all size of particles and iterations also decreases.

Table 3.5 Results of SBPSO to Sphere Function by varying p, t and q

	1	2	3	4	5	6
S.No.	P	t	q	F	Kount	Iteration
1	10	1	10	-9	1260	125
2	20	2	10	-7	890	85
3	20	2	5	-7	600	111
4	40	2	10	-10	1000	91
5	40	2	5	-6	530	86
6	80	2	10	-11	1040	84
7	80	2	5	-8	600	77
8	160	2	10	-10	1370	94
9	160	2	5	-8	840	78
10	320	2	10	-10	1620	72
11	320	2	5	-7	1330	81
12	80	4	10	-9	1140	97
13	80	4	5	-7	575	80
14	160	4	10	-9	1326	97
15	160	4	5	-7	750	78
16	320	4	10	-9	1630	97
17	320	4	5	-6	1100	74
18	160	8	10	-9	1260	93
19	160	8	5	-8	775	88
20	320	8	10	-15	1500	83
21	320	8	5	-7	1095	84
22	640	8	10	-11	2400	105
23	640	8	5	-5	1765	81

3.6.5 Rastrigin's Function

$$f(x) = \sum_{i=1}^{n} (x_i^2 - 10 * \cos(2* \text{pi} * x_i))$$

$$\mathbf{F(X_1,X_2) = 20 + (X_1^2) - (10 * \cos (2*pi*X_1)) + (X_2^2) - (10 * \cos (2*pi*X_2))}$$

Minimum value and range for the function are as follows:

- Optimal values of $X_1=0, X_2=0$

Table 3.6 shows the application of SBPSO for the Rastrigin's function. Column (1)of this table shows the numbers of particles 'p'. Column (2) shows decrement factor 't'. Column (3) of this table shows 'q' which is minimum numbers of particles required to optimize a function. Column (4), (5) and (6) show the function value, kount and number of iterations respectively. S.No. 1 of this table shows the result of Basic PSO (BPSO) for minimum particle size of '10' required to optimize the Rastrigin's function. The remaining rows show the result of SBPSO.

For S.No. 2 to 11, the decrement factor 't' is fixed to 2 for various values of initial particle size varying from 20 to 640 in multiples of 2. It is observed that for particle size 20 to 160, the number of iterations and the value of kount is less compared to BPSO. It is observed that by taking t=2 (decrement factor) for a minimum value of 10 particles (i.e. q=10), the kount is being decreased in the range of 20 to 160 particles (i.e. 'p'). There is a gradual decrease in iteration in the same range. It is observed that by taking t=2 (decrement factor) for a minimum value of 5 particles (i.e. q=5), the kount is being decreased in the range of 20 to 320 particles (i.e. 'p'). Although the function value is not as accurate as that in basic PSO but it is optimum. There is a gradual decrease in iteration in the same range. For a minimum value of 10 particles (q=10), for initial size of 80, 160, 320 and 640 on increasing decrement factor to 4 & 8, kounts decreases for 80 & 160 particles compared to the result shown at S.No. 1 for Basic PSO(BPSO). It is to be noted that BPSO didn't converge when particle size is chosen to '5'. Similarly, for a minimum value of 5 particles (q=5), for initial size of 80, 160, 320 and 640 on increasing decrement factor to 4 &8 , kounts decreases for all size of particles and iterations also decreases.

Table 3.6 Results of SBPSO to Rastrigin's Function by varying p, t and q

	1	2	3	4	5	6
S.No.	p	t	q	F	Kount	Iteration
1	10	1	10	-12	1300	129
2	20	2	10	-8	1030	99
3	20	2	5	-9	550	101
4	40	2	10	-19	1080	99
5	40	2	5	-7	595	92
6	80	2	10	-20	1280	108
7	80	2	5	-14	685	94
8	160	2	10	-22	1340	91
9	160	2	5	-9	900	92
10	320	2	10	-18	1730	83
11	320	2	5	-13	1240	63
12	80	4	10	-18	1040	87
13	80	4	5	-10	585	82
14	160	4	10	-8	1460	111
15	160	4	5	-7	645	89
16	320	4	10	-18	1680	96
17	320	4	5	-12	990	52
18	160	8	10	-17	1340	101
19	160	8	5	-6	900	113
20	320	8	10	-13	1450	78
21	320	8	5	-12	1030	71
22	640	8	10	-17	2250	90
23	640	8	5	-14	1760	80

3.7 DISCUSSION

SBPSO has been successfully applied to Mathematical Benchmark functions – Rosenbrock, Beale, Sphere, Booth and Rastrigin function. The value of random numbers Rp and Rg have been kept to fix value 0.5 and 0.6 respectively and the values of constriction factor Cp and Cg have been fixed to 2 in velocity modification equation. In SBPSO, size of the particles changes in every iteration based on function value. The size of particles decreases by the factor 't' in every iteration. The various values of 't' tried are 2, 4 & 8 for various initial size of particles. The various initial sizes of particles considered are 20, 40, 80, 160, 320 & 640. The present considered value of particle size is double of its previous size.

SBPSO has been compared with Basic PSO (BPSO) based on kount and number of iteration required to converge the function. It has been observed that the minimum particles size required for optimizing a function in BPSO is '10'. SBPSO could optimize the function for particle size less than the minimum particle size of BPSO i.e. SBPSO optimize the function for '5' particles.

Kount and number of iterations decreased for all the function for all initial size of particle considered.

CHAPTER 4:

APPLICATION OF SBPSO TO ECONOMIC LOAD DISPATCH

4.1 Introduction To Economic Load Dispatch

The economic load dispatch (ELD) problem is one of the important optimization problems in the electric power system. The objective of the ELD of electric power generation is to schedule the committed generating unit outputs so as to meet the required load demand at minimum operating cost while satisfying all unit and system equality and inequality constraints. This makes the ELD problem a large-scale highly nonlinear constrained optimization problem. Improvements in scheduling the unit outputs can lead to significant cost savings.

4.2 LIST OF SYMBOLS

a_i , b_i , c_i are the cost coefficient of i^{th} generator

P_{gi} : the active power generation of the i^{th} generator

$P_{gmax,i}$ $P_{gmin,i}$: are the maximum and minimum power generation limits of i^{th} thermal unit

Ng : Total number of generators committed.

P_d : Total system demand

P_L : Total system losses

Pg : Total power generated.

B_{mn}, B_{0m}, B_{00} : Transmission losses B co-efficient

Z : penalty factor

P_{gm}, P_{gn} is the active power at the m^{th} and n^{th} generator.

NG is the total number of generators in the system.

B_{mn}, B_{om}, B_{oo} are loss coefficients.

4.3 MATHEMATICAL FUNCTION

Objective function being used to minimize the cost of generation is given as :

$$F_C = \sum_{i=1}^{NG} F\left[C_i\left(P_{gi}\right)\right] \quad (4.1)$$

Where cost function is defined as:

$$C_i\left(P_{gi}\right) = \sum_{i=1}^{NG} (a_i P_{gi}^2 + b_i P_{gi} + c_i) \quad (4.2)$$

The objective function used to find the system transmission losses is given as:

$$P_L = \sum_{m=1}^{NG} \sum_{n=1}^{NG} P_{gm} B_{mn} P_{gn} + \sum_{m=1}^{NG} B_{om} P_{gm} + B_{oo} \quad (4.3)$$

Subject to the constraints:

Equality constraint

$$\sum_{i=1}^{NG} P_{gi} = P_D + P_L \quad (4.4)$$

Inequality constraint

$$P_{gimin} \leq P_{gi} \leq P_{gimax} \quad i = 1, 2 \ldots NG \quad (4.5)$$

Here:

F	objective function to be optimized
F_C	cost of the generation
P_L	system transmission losses

4.4 COMPUTATIONAL PROCEDURE:

Selection Based Particle Swarm Optimization (SBPSO) has been used to perform the optimization of ELD function. To consider the equality constraint of the problem, the function has been modified by inclusion of a parameter Z. The objective function becomes as follows:

$$F=F_C+Z(P_D+P_L-P_G) \quad (4.6)$$

Where:

Parameter Z is fixed at 500 for all three IEEE 5, 14 and 30 bus systems. Different values of Z were considered and it was observed that ELD problem converged when it was fixed to 500 for all the systems.

Inequality constraints have been considered in the PSO programming which is done in the MATLAB. The program checks the power output of each particle for each generator in each iterations and the power is tied to the corresponding limit violated. Logic to implement the inequality constraint is as shown below:

for i=1: N_G

for m=1: p

if $P_{gi} < P_{gimin}$

$P_{gi} = P_{gimin}$

end

if $P_{gi} > P_{gimax}$

$P_{gi} = P_{gimax}$

end

end

end

The optimum solution is obtained when the

i. Change in the value of Economic Load Dispatch function during successive iterations is less than the limit specified which is $T=10^{-6}$ and
ii. The equality constraint is satisfied such that the absolute value of difference between generation, demand and losses is less than $T=10^{-6}$.

With the assistance of MATLAB, we generate randomly the initial position and velocity of particles. To build the convergence rate, breaking points are forced on position of particles. Here positions i.e. the generations are decision variables. The most extreme and least points of confinement on the velocity have been allocated as Vmin = –Pgimin/2 and Vmax = Pgimax/2 individually. The velocities are altered to the values of corresponding limits if violated during the iterations. Initial estimations of personal best and global best have been taken as the initial esteem randomly generated by MATLAB.

The sequence for the solution of Economic Load Dispatch problem using Particle Swarm Optimization technique is explained as follows:

1. Fix the no. of particles 'p' in swarm and set the no. of maximum iterations it_{max} and tolerance value T.
2. Fix the factor of decrement 't' and a last size of particle is to be chosen say 'q'.
3. Fix the cost coefficients, loss coefficients, and load demand and generator limits of all the generators.
4. Generate P_i^k and V_i^k, the initial random positions (i.e. generations) and velocity (i.e. updation factor) respectively.
5. Set iteration kount K = 0.
6. Calculate the losses for each particle, using the eq. (4.3).
7. Calculate the value of ELD function using eq. (4.6).
8. At 0^{th} iteration the personal and global best positions (i.e. generations) are same as the initial random positions (i.e. generations).
9. Increase the iteration kount k by 1 using k=k+1 in the every run of ELD function.
10. Calculate the velocity (i.e. positions updating factor) of each particle using eq.(2.1).
11. Check if velocity is within the limits. Fix the velocity to the limit violated.
12. Calculate the new positions (i.e. generations) of the particles by evaluating eq.(2.2).
13. Check if generations (i.e. positions) of each particle are within the generator limits, if not fix the generation to the limit violated.

14. Sort the particles with accordance to their function values in increasing order.
15. Size of the particles is updated by the factor p/t say new ‘P’, and the size of particles updated is ‘P’.
16. Save new P to p.
17. Select the top ‘p’ no. of particles from the sorted particles with accordance to their function values in increasing order.
18. If p < q, make ‘p’ equal to ‘q’ and initialize the value of ‘t’ equals to 1 and go to next step.
19. Calculate ELD function for the new size of particles (P or p) and new positions (i.e. generations) that has generated.
20. Update Xpbest and Xgbest values by comparing ELD function values.
21. Check if both the stopping criteria are satisfied, if not then go to step 10, else stop.
22. Output the values of cost of generation and system transmission losses.

4.5 Flowchart

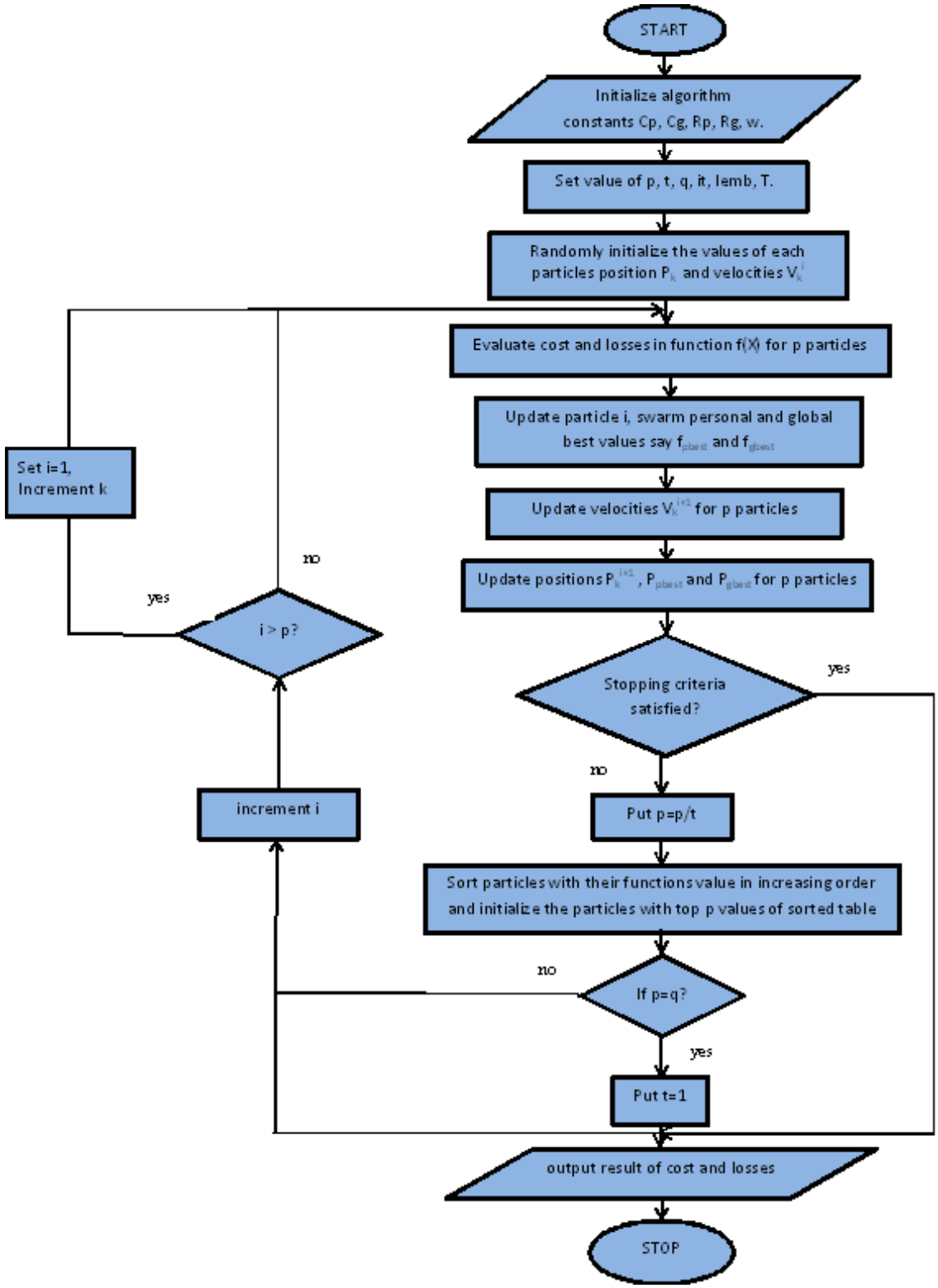

Fig 7 ELD to SBPSO evaluation flowchart

4.6 COMPUTATIONAL RESULTS:

SBPSO has been applied to IEEE 5, 14 and 30 bus system and results are shown in tables 4.1, 4.2 and 4.3 respectively.

TABLE 4.1 Results of IEEE 5-bus system by SBPSO varying p, t and q

	1	2	3	4	5	6	7
S.NO.	p	t	q	Cost	Losses	Kount	Iteration
1.	10	1	10	765.25	5.11	1180	117
2.	40	2	10	762.33	5.11	950	87
3.	40	4	10	762.79	5.10	1150	108
4.	80	2	10	763.91	5.10	1180	99
5.	80	4	10	761.30	5.16	1070	91
6.	80	8	10	763.24	5.10	1140	99
7.	160	2	10	761.23	5.18	1350	93
8.	160	4	10	762.25	5.11	1320	98
9.	160	8	10	761.58	5.13	1200	88
10.	160	16	10	764.63	5.10	1200	89
11.	40	2	5	762.64	5.10	555	92
12.	40	4	5	762.34	5.11	580	100
13.	80	2	5	766.06	5.13	640	86
14.	80	4	5	762.44	5.10	600	86
15.	80	8	5	762.25	5.11	595	87
16.	160	2	5	761.79	5.12	930	97
17.	160	4	5	761.35	5.16	785	86
18.	160	8	5	764.90	5.11	750	84
19.	160	16	5	765.27	5.11	820	100

TABLE 4.2 Results of IEEE 14-bus system by SBPSO varying p, t and q

	1	2	3	4	5	6	7
S.NO.	p	t	q	Cost	Losses	Kount	Iteration
1.	10	1	10	1192.97	7.47	1030	102
2.	40	2	10	1069.87	7.21	1010	93
3.	40	4	10	1177.33	6.99	920	85
4.	80	2	10	1162.69	7.5	1120	93
5.	80	4	10	1174.48	6.9317	980	82
6.	80	8	10	1168.00	7.14	1060	91
7.	160	2	10	1157.65	7.97	1290	87
8.	160	4	10	1162.01	743	1180	84
9.	160	8	10	1160.27	7.52	1270	95
10.	160	16	10	1163.27	7.23	1150	84
11.	40	2	5	1182.18	7.05	565	94
12.	40	4	5	1194.78	6.56	690	122
13.	80	2	5	1188.13	6.66	660	90
14.	80	4	5	1206.34	6.538	645	95
15.	80	8	5	1211.45	6.500	635	95
16.	160	2	5	1191.47	6.59	910	93
17.	160	4	5	1183.2	6.72	800	89
18.	160	8	5	1211.6	6.49	785	91
19.	160	16	5	1208.0	6.52	770	90

TABLE 4.3 Results of IEEE 30-bus system by SBPSO varying p, t and q

	1	2	3	4	5	6	7
S.NO.	p	t	q	Cost	Losses	Kount	Iteration
1.	10	1	10	1286.33	8.46	1490	148
2.	40	2	10	1280.93	8.80	1200	112
3.	40	4	10	1279.3	8.86	1040	97
4.	80	2	10	1295.94	8.09	1250	106
5.	80	4	10	1294.46	8.14	1240	108
6.	80	8	10	1294.46	8.14	1100	95
7.	160	2	10	1288.96	8.34	1390	97
8.	160	4	10	1284	9.39	1410	107
9.	160	8	10	1292.83	9.78	1330	101
10.	160	16	10	1282.89	8.64	1480	117
11.	40	2	5	1279.5	8.85	825	146
12.	40	4	5	1275.7	9.18	575	99
13.	80	2	5	1287.4	8.41	695	97
14.	80	4	5	1285.3	8.5	645	95
15.	80	8	5	1280.13	8.5	690	106
16.	160	2	5	1280.13	8.8	915	94
17.	160	4	5	1286.5	8.45	1000	129
18.	160	8	5	1292.94	8.19	825	99
19.	160	16	5	1294.53	8.14	765	89

3

For IEEE 5 bus system, the number kounts is less than BPSO sizes 40 and 80 for the decrement factor ‘t’ as 2, 4 and 8 for the minimum number of 10 particles. Similarly, the number kounts is decreasing for the particles 40, 80 and 160 for the decrement factor as 2, 4, 8 and 16 for the minimum number of 5 particles. This is seen from the result shown at S.No. 5 and 7 the cost being optimized and the losses are increased as compared to the Basic PSO. These two quantities are inverse to each other with increase in cost of generation, losses of transmission line decrease.

For IEEE 14 bus system, the number kounts is less than BPSO sizes 40 and 80 for the decrement factor ‘t’ as 2, 4 and 8 for the minimum number of 10 particles. Similarly, the number kounts is decreasing for the particles 40, 80 and 160 for the decrement factor as 2, 4, 8 and 16 for the minimum number of 5 particles. This is seen from the result shown at S.No. 8 and 15 the cost being optimized and the losses are increased as compared to the Basic PSO. These two quantities are inverse to each other with increase in cost of generation, losses of transmission line decrease.

For IEEE 30 bus system, the number kounts is less than BPSO sizes 40 and 80 for the decrement factor ‘t’ as 2, 4 and 8 for the minimum number of 10 particles. Similarly, the number kounts is decreasing for the particles 40, 80 and 160 for the decrement factor as 2, 4, 8 and 16 for the minimum number of 5 particles. This is seen from the result shown at S.No. 5, 9 and 11 the cost being optimized and the losses are increased as compared to the Basic PSO. These two quantities are inverse to each other with increase in cost of generation, losses of transmission line decrease.

It is observed that the result of SBPSO are better than BPSO in terms of kount and number of iterations for IEEE 5, 14 & 30 bus systems.

CHAPTER 5
CONCLUSION AND FUTURE DIRECTIONS

5.1CONCLUSION

Following are the definite contributions of the work:

1. Basic PSO has been implemented on mathematical benchmark function and minimum number of particles required to optimise the function has been determined. This size is 10 for all functions.
2. Selection Based PSO (SBPSO) has been developed in which a better population of particle is selected in each iteration based on function value. Population in each iteration is decreased by decrement factor 't'.
3. SBPSO has been implemented to mathematical benchmark functions and Economic Load Dispatch problem for IEEE 5, 14 and 30 bus system.
4. SBPSO is found to converge for minimum '5' particles for which BPSO fails to converge.
5. For SBPSO kount value, a formula has been developed by considering computational result of various function.
6. SBPSO is found to be computationally faster than BPSO.
7. With large increase in size of particles and decrement factor results are better i.e. both cost of generation and transmission line losses decreases in the system but the number of kounts increases.
8. The technique is applied to the particle size lying in the range 40 to 160 with decrement factor of 2, 4, 8 and 16 for the minimum number of particle size 5 and 10. For the combination of parameters, the result are found to be better interms of function value and kount value than that of BPSO for 10 particles for both mathematical benchmark functions as well as Economic Load Dispatch problem for IEEE 5, 14 and 30 bus system.
9. For p=80, t=4 and q=5 the function converge faster and are most accurate.
10. The ELD problem found to be converge faster and is more accurate for the following combination of parameters 1) p=160, t=4 and q=5 for 5 bus system, 2) p=80, t=4 and q=10 for 14 bus system and 3) p=40, t=4 and q=5 for 30 bus system.

5.2 FUTURE DIRECTION

1. The effect of variation of all the parameters such as random numbers (Rp & Rg) and acceleration coefficient (Cp & Cg) and inertia weight. By varying one parameter at a time on convergence of SBPSO should be studied.
2. Considering other objective of power system security, environmental degradation due to pollution, stability, reliability, etc.
3. The convergence and accuracy of PSO should be increased

REFERENCES

[1]. J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", IEEE International Conference on Neural Networks, vol. 4, pp. 1942-1948, Perth, Australia, 1995.

[2]. Ajith Abraham,He Guo, and Hongbo Liu , "Swarm Intelligence:Foundations,Perspectives and Applications" Studies in Computational intelligence (SCI)26,3-25 (2006)

[3]. Karin Zielinski and ainer taur "Stopping Criteria for a Constrained Single –Objective Particle Swarm Optimization Algorithm Informatica 31 (2007) 51-54.

[4]. W.B.Langdon,Riccardo Poli,Chirstopher R Stephens, "Kernel methods for PSOs"20 December 2005.

[5]. Mark S. Voss. "Principal Component Particle Swarm Optimization (PCPSO) 0-7803-8916-6/05 IEEE.

[6]. JACO F. SCHUTTE and ALBERT A.GROENWOLD, "A study of Global Optimization Using Particle Swarms"Journal of Global Optimization Springer 2005.

[7]. Wei-Bing Liu and Xian -Jia Wang, "An evolutionary game based particle swarm optimization algorithm."journal of computational and Applied Mathematics214(2008)30-35.

[8]. Wei Zu and Yan-Ling Hao et al.(2008), "Enhancing the Particle Swarm Optimization based on Equilibrium of Distribution." 978-1-4244-1734-6/08 IEEE. 2008 Chinese Control and Decision Conference (CCDC 2008).

[9]. Serkan Kiranyaz, Jenni Pulkkinen and Moncef Gabbouj, "Multi-dimensional Paricle Swarm Optimization for Dynamic Environments"978-1-4244-3397-1/08/2008 IEEE.

[10]. Bilal Benmessahel, Mohamed Touahria, An improved Combinatorial Particle Swarm Optimization Algorithm to Database Verticle Partition "Journal of Emerging Trends in Computing and Information Sciences Vol 2 No.3 ISSN 2079-8407.

[11]. Huanhuan Ji,Jing Jie ,Junji Li and Ying Tan, "A Bi-swarm particle swarm optimization with cooperative Co-evolution" International Conference on Computational Aspects of Social Networks 978-0-7695-4202-7/10 2010IEEE.

[12]. Zhi Li and Yong Chen , "Design and implementation for parallel Particle Swarm Optimization Color Quantization Algorithm"International Conference on Computer and Information Application 978-1-4244-8598-7/10 IEEE

[13]. Weidong Jii and Keqi Wangi, "An Improved Particle Swarm Optimization Algorithm" International Conference on Computer Science and Network Technology,2011

[14]. Ismail Ibrahim, Zulkifli Md. Yusof, Sophan, Hamzah Ahmad, Zuwairie Ibrahim, "A Novel Multi-State Particle Swarm Optimization for Discrete Combinatorial Optimization Problems" Fourth International Conference on Computational Intelligence, Modelling and Simulation, 2012

[15]. Kyle Robert Harrison, "GP- Based adaptable Evolutionary Particle Swarm Optimization " November 23,2012

[16]. Nikhil Padhye , Kalyanmoy Deb, and Pulkit Mittal, "Boundary Handling Approaches in Particle Swarm Optimization", KanGAL Report Number 2012014 July 28, 2012.

[17]. Zahra Beheshti, Siti Mariyam Shamsuddin ,and Siti Sophiayati Yuhaniz**,** "Binary Accelerated Particle Swarm Algorithm (BAPSA)for discrete optimization problems.

[18]. Luis Miguel Rios · Nikolaos V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations" J Glob Optim (2013) 56:1247–1293 DOI 10.1007/s10898-012-9951-y

[19]. Zhimin Chen1, Yuming Bo1, Panlong Wu1, Weijun Zhou1, "A new particle filter based on organizational adjustment particle swarm optimization" Appl. Math. Inf. Sci. **7**, No. 1, 179-186 (2013)

[20]. Lin Lu, Qi Luo, Jun-yong Liu, and Chuan Long "An Improved Particle Swarm Optimization Algorithm" Sichuan University, Chengdu, Sichuan, China, 610065.

[21]. Kyle Robert Harrison, “GP- Based adaptable Evolutionary Particle Swarm Optimization “ November 23,2012.

[22]. YAN Zhe-ping, DENG Chao, Zhou Jia-jia, Chi Dong-nan , “A Novel Two-subpopulation Particle Swarm Optimization”Proceedings of the 10thWorld Congress on Intelligent Control and Automation July 6-8, 2012, Beijing, China.

[23]. Nikhil Padhye , Kalyanmoy Deb, and Pulkit Mittal, “Boundary Handling Approaches in Particle Swarm Optimization”, KanGAL Report Number 2012014 July 28, 2012.

[24]. Zahra Beheshti, Siti Mariyam Shamsuddin ,and Siti Sophiayati Yuhaniz**,** “Binary Accelerated Particle Swarm Algorithm (BAPSA)for discrete optimization problems.

[25]. Luis Miguel Rios · Nikolaos V. Sahinidis, “Derivative-free optimization: a review of algorithms and comparison of software implementations” J Glob Optim (2013) 56:1247–1293 DOI 10.1007/s10898-012-9951-y

[26]. Zhimin Chen1, Yuming Bo1, Panlong Wu1, Weijun Zhou1, “A new particle filter based on organizational adjustment particle swarm optimization” Appl. Math. Inf. Sci. **7**, No. 1, 179-186 (2013)

[27]. Lin Lu, Qi Luo, Jun-Yong Liu, And Chuan Long “An Improved Particle Swarm Optimization Algorithm” Sichuan University, Chengdu, Sichuan, China, 610065

[28]. Megahed I., Abou-Taleb N. and Iskandrani M. , “A modified method for solving the economic dispatching problem”, IEEEE Transactions on Power Apparatus and Systems, Vol. PAS-96 (I), pp. 124-133, January/February, 1977.

[29]. Happ H. H., “Optimal power dispatch - a comprehensive survey”, IEEE Transactions on Power Apparatus an8 Systems, Vol. PAS-96, no. 3, May / june1977.

[30]. Kwatny H. G. and Athay T. A., "Coordination of economic dispatch and load frequency control in electric power systems," Proceedings, 18th IEEE Conference on Decision and Control, 1979.

[31]. Aoki K. and Satoh T., " Economic dispatch with network security constraints using parametric quadratic programming", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 12 December 1982

[32]. Lin C. E. and Viviani G. L., "Hierarchical Economic Dispatch for Piecewise Quadratic Cost Functions", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-103, No. 6, June 1984.

[33]. Ramanathan R., "Fast economic dispatch based on the penalty factors from Newton' s method", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 7, July 1985.

[34]. Walters David C. and Sheble Gerald B., "Genetic algorithm solution of economic dispatch with valve point loading", IEEE Transactions on Power Systems, Vol. 8, No. 3, August 1993.

[35]. Abdullah and Bakar ,"Implementation of Hybrid Particle Swarm Optimization for Combined Economic-Emission Load Dispatch Problem" IEEE 8th International Power Engineering and Optimization Conference 24-25 March 2014, pp 402-409.

[36]. Bhattacharya and Chattopadhyay, "A Modified Particle Swarm Optimization for Solving the Non-Convex Economic Dispatch" Proc. of the IEEE Conference Evolutionary Computation, 2009.

[37]. Dasgupta and Banerjee, "An Analysis of Economic Load Dispatch using Different Algorithms" 1st International Conference on Non Conventional Energy (ICONCE 2014), pp 216-219

[38]. N.K.Jain and Uma Nangia, "Analysis of Economic Load Dispatch Using Improved Partical Swarm Optimization" IEEE 2014.

[39]. Liu, Han and Zhou, "Hierarchical Economic Load Dispatch Based on Chaotic-particle Swarm Optimization", Ninth International Conference on Natural Computation (ICNC) 2013, pp 517-521.

[40]. Mark M. Millonas, "Swarms, Phase Transitions, and Collective Intelligence", *Complex Systems Group, Theoretical Division and Center for Nonlinear Studies*, MS B258 Los Alamos National Laboratory, Los Alamos, NM 87545 & Santa Fe Institute, Santa Fe, NM, pp. 1-32.

[41]. R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," *IEEE International Conference on Evolutionary Computation*, pp. 611-616, May 1998.

APPENDIX- I

1) IEEE 5 BUS SYSTEM

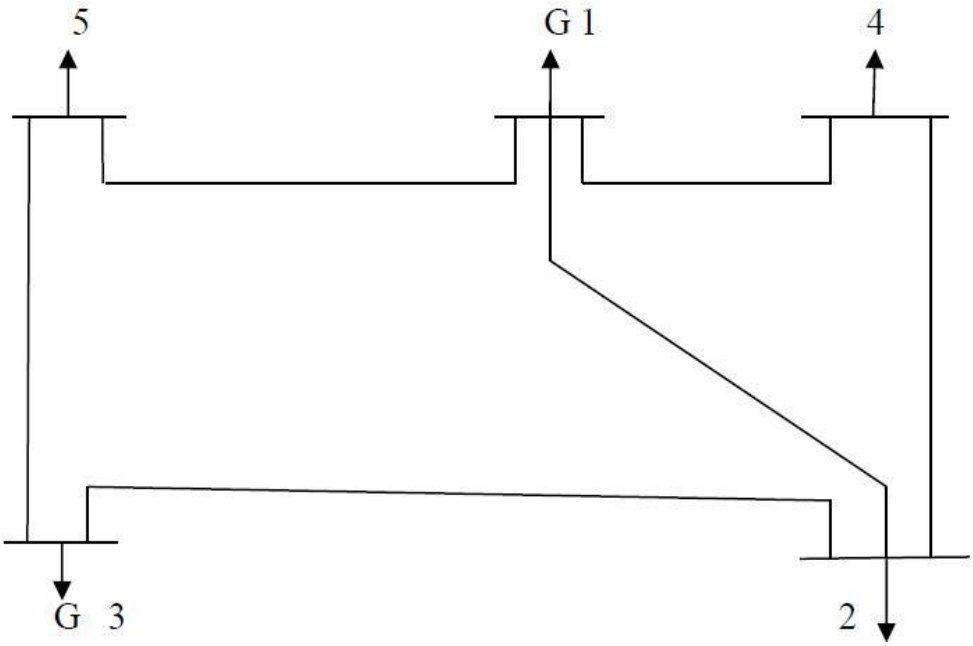

Fig. 8: BUS-CODE DIAGRAM OF 5 BUS SYSTEM

TABLE I-A: LINE DATA OR IMPEDANCE DATA (5 BUS SYSTEM)

LINE DESIGNATION	*R(p.u.)	*X(p.u.)	LINE CHARGING
1-2	0.10	0.4	0.0
1-4	0.15	0.6	0.0
1-5	0.05	0.2	0.0
2-3	0.05	0.2	0.0
2-4	0.10	0.4	0.0
3-5	0.05	0.2	0.0

*The impedance are based on MVA as 100

TABLE I-B: BUS DATA or OPERATING CONDITIONS (5 BUS SYSTEM)

	GENERATION		LOAD	
BUS NO.	MW	VOLTAGE MAGNITUDE	MW	MVAR
1*	- - -	1.02	- - -	- - -
2	- - -	- - -	60	30
3	100	1.04	- - -	- - -
4	- - -	- - -	40	10
5	- - -	- - -	60	20

*Slack Bus

TABLE I-C: REGULATED BUS DATA (5 BUS SYSTEM)

BUS NO.	VOLTAGE MAGNITUDE	MVAR CAPACITY		MW CAPACITY	
		MINIMUM	MAXIMUM	MINIMUM	MAXIMUM
1	1.02	0.0	60	30	120
3	1.04	0.0	60	30	120

The nodal load voltage inequality constraints are $0.9 \leq Vi \leq 1.05$

Cost characteristics of IEEE 5 bus system

The cost characteristics of the IEEE 5 Bus System are as follows: $C_1=50p_1^2+351p_1+44.4$ \$/hr.

$C_3=50p_3^2+389p_3+40.6$ \$/hr.

Here, the total load demand of the system is 160 MW. Maximum and minimum active power constraint on the generator bus for the given system is 120 MW and 30 MW respectively. Voltage magnitude constraint for generator at bus 3 is 1.04 pu.

M-file For Calculating B- Coefficients:

```
Clear
basemva=100
accuracy=0.0001
maxiter=10
busdata=[1 1 1.02 0 0 0 0 0 0 60 0;2 0 1 0 60 30 0 0 0 0 0;3 2 1.04 0 0 0 82 0 0 60
0;4 0 1 0 40 10 0 0 0 0 0;5 0 1 0 60 20 0 0 0 0 0];
Linedata=[1 2 0.10 0.4 0 1;1 4 0.15 0.6 0 1; 1 5 0.05 0.2 0 1;2 3 0.05 0.2 0 1;2 4 0.10
0.4 0 1;3 5 0.05 0.2 0 1];
disp(busdata)
disp(linedata)
mwlimit=[30 120;30
120]; Ifybus
Ifnewton
busout
bloss
```

B-Coefficient Calculated is as:

B11 = 0.00035336 B12 = 0.0000103196

B21 = 0.0000103196 B22 = 0.000368992

2) IEEE 14 BUS SYSTEM

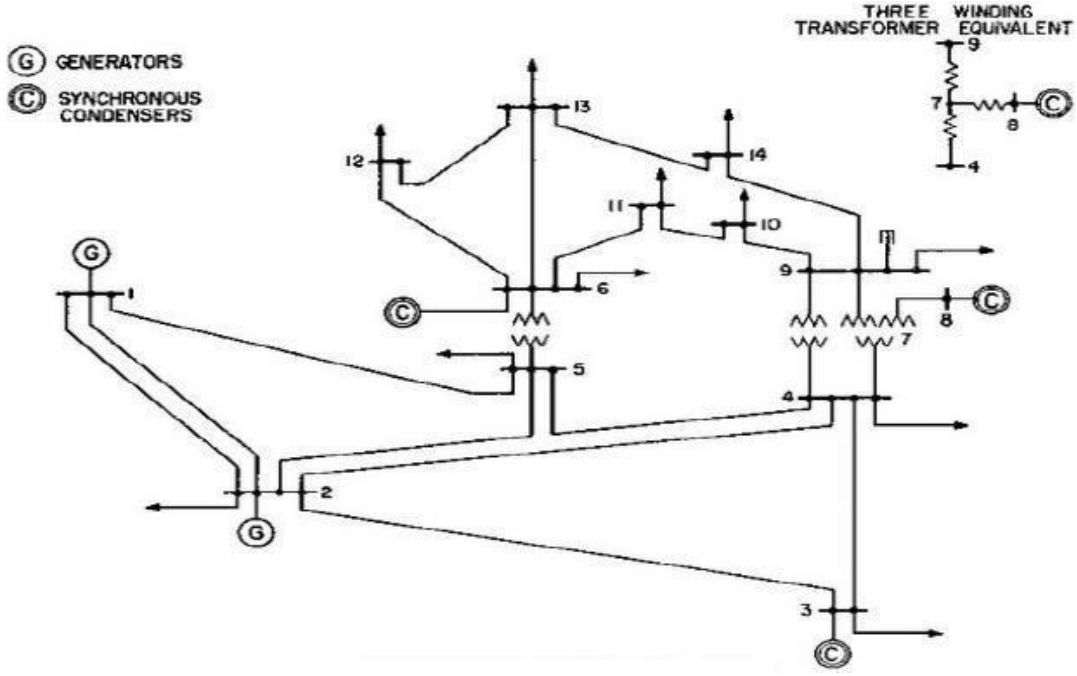

Fig. 9: BUS-CODE DIAGRAM OF 14 BUS SYSTEM

TABLE I-D: IMPEDANCE & LINE-CHARGING DATA (14 BUS SYSTEM)

Line Designation	Resistance p.u. *	Reactance p.u. *	Line Charging	Tap Setting
1-2	0.019379	0.059170	0.0264	1
1-5	0.054029	0.223040	0.0264	1
2-3	0.046980	0.197970	0.0219	1
2-4	0.058110	0.176320	0.0187	1
2-5	0.056950	0.173880	0.0170	1
3-4	0.067010	0.171030	0.0173	1
4-5	0.013350	0.042110	0.0064	1
4-7	0	0.20912	0	1
4-9	0	0.55618	0	1
5-6	0	0.25202	0	1
6-11	0.09498	0.19890	0	1
6-12	0.12291	0.25581	0	1
6-13	0.06615	0.13027	0	1
7-8	0	0.17615	0	1
7-9	0	0.11001	0	1
9-10	0.03181	0.08450	0	1
9-14	0.12711	0.27038	0	1
10-11	0.08205	0.19207	0	1
12-13	0.22092	0.19988	0	1
13-14	0.17093	0.34802	0	1

* Impedance and line-charging susceptance in p.u. on a 100 MVA base.

TABLE I-E: BUS DATA or OPERATING CONDITIONS (14 BUSSYSTEM)

Bus No.	Voltage		Generation		Load	
	Magnitude (in pu)	Phase angle (deg.)	MW	MVAR	MW	MVAR
1*	1.06	0	0	0	0	0
2	1	0	40	0	21.7	12.7
3	1	0	0	0	94.2	19.0
4	1	0	0	0	47.8	-3.9
5	1	0	0	0	7.6	1.6
6	1	0	0	0	11.2	7.5
7	1	0	0	0	0	0
8	1	0	0	0	0	0
9	1	0	0	0	29.5	16.6
10	1	0	0	0	9.0	5.8
11	1	0	0	0	3.5	1.8
12	1	0	0	0	6.1	1.6
13	1	0	0	0	13.5	5.8
14	1	0	0	0	14.9	5.0

*Slack Bus

TABLE I-F: REGULATED BUS DATA (14 BUS SYSTEM)

Bus no.	Voltage magnitude (in pu)	Minimum MVAR capability	Maximum MVAR capability
2	1.05	-40	50
3	1.010	0	40
6	1.070	-6	24
8	1.090	-6	24

Cost characteristics of IEEE 14 bus system

The cost characteristics of the IEEE 14 Bus System are as follows: $C_1 = 50p_1^2+245p_1+105$ \$/hr.

$C_2 = 50p_2^2+351p_2+44.4$ \$/hr.

$C_6 = 50p_6^2+389p_6+40.6$ \$/hr.

Here, the total load demand of the system is 259 MW. The maximum active power constraint is 200 MW, 100MW and 100 MW for the generators of bus no. 1, 2 and 6 respectively. The minimum active power constraint is 50 MW, 20MW and 20 MW for the generators of bus no. 1, 2 and 6 respectively. Voltage magnitude

constraint for generator at bus 2 is 1.045, for bus no. 6 is 1.070, for bus no. 3 is 1.010 & for bus no. 8 is 1.090.

M-file For Calculating B- Coefficients:

```
Clear
basemva=100
accuracy=0.0001
maxiter=10
busdata=[1 1 1.06 0 0 150 0 0 0 0 0;2 2 1.045 0 21.7 12.7 63.11 0 -40 50 0;3 0 1.01 0
94.2 19 0 0 0 40 0;4 0 1 0 47.8 -3.9 0 0 0 0 0;5 0 1 0 7.6 1.6 0 0 0 0 0;6 2 1.07 0 11.2
7.5 77.12 0 -6 24 0;7 0 1 0 0 0 0 0 0 0 0;8 0 1.09 0 0 0 0 0 -6 24 0 ;9 0 1 0 29.5 16.6 0
0 0 0 0; 10 0 1 0 9 5.8 0 0 0 0 0;11 0 1 0 3.5 1.8 0 0 0 0 0;12 0 1 0 6.1 1.6 0 0 0 0
0;13 0 1 0 13.5 5.8 0 0 0 0 0;14 0 1 0 14.9 5 0 0 0 0 0];
linedata=[1 2 0.01938 0.05917 0.0264 1;1 5 0.05403 0.22304 0.0246 1; 2 3 0.04699
0.19797 0.0219 1; 2 4 0.05811 0.17632 0.0170 1; 2 5 0.05695 0.17388 0.0173 1; 3 4
0.06701 0.17103 0.0064 1; 4 5 0.01335 0.04211 0.0 1; 4 7 0.0 0.20912 0.0 0.978; 4 9
0.0 0.55618 0.0 0.969;5 6 0.0 0.25202 0.0 0.932; 6 11 0.09498 0.19890 0.0 1;6 12
0.12291 0.25581 0.0 1;6 13 0.06615 0.13027 0.0 1;7 8 0.0 0.17615 0.0 1; 7 9 0.0
0.11001 0.0 1; 9 10 0.03181 0.08450 0.0 1;9 14 0.12711 0.27038 0.0 1; 10 11
0.08205 0.19207 0.0 1;12 13 0.22092 0.19988 0.0 1;13 14 0.17093 0.34802 0.0 1];
disp(busdata)
disp(linedata)
mwlimit=[50 200;20 100;20
100] Ifybus
Ifnewton
busout
bloss
```

B-Coefficient Calculated is as:

B11 = 0.0231 B12 = 0.0078 B13 = -0.0007

B21 = 0.0078 B22=0.0182 B23= 0.0022

B31=-0.0007 B32= 0.0022 B33= 0.0329

C) IEEE 30 BUS SYSTEM

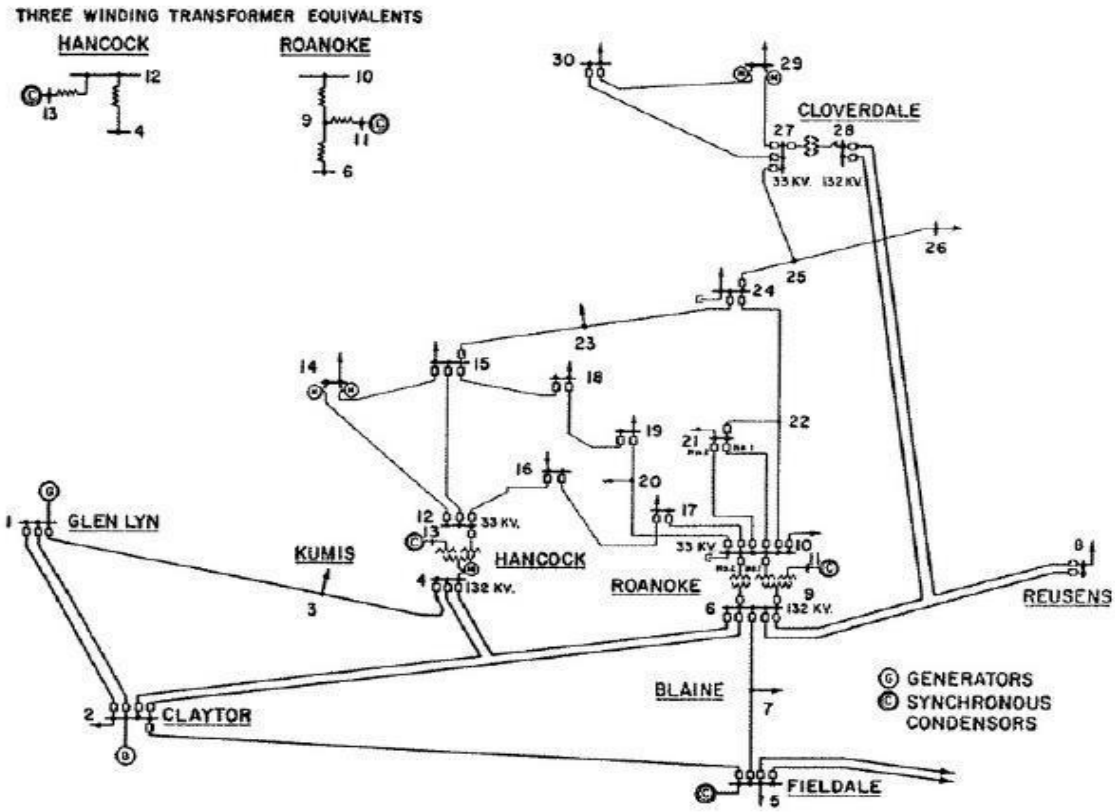

Fig. 10: BUS-CODE DIAGRAM OF 30 BUS SYSTEM

TABLE I-G: IMPEDANCE & LINE-CHARGING DATA (30 BUS SYSTEM)

Line Designation	Resistance p.u.*	Reactance p.u.*	Line Charging	Tap Setting
1-2	0.0192	0.0575	0.0264	1
1-3	0.0452	0.1852	0.0204	1
2-4	0.0570	0.1737	0.0184	1
3-4	0.0132	0.0379	0.0042	1
2-5	0.0472	0.1983	0.0209	1
2-6	0.0581	0.1763	0.0187	1
4-6	0.0119	0.0414	0.0045	1
5-7	0.0460	0.1160	0.0102	1
6-7	0.0267	0.0820	0.0085	1
6-8	0.0120	0.0420	0.0045	1
6-9	0	0.2080	0	0.978
6-10	0	0.5560	0	0.969
9-11	0	0.2080	0	1
9-10	0	0.1100	0	1
4-12	0	0.2560	0	0.932
12-13	0	0.1400	0	1
12-14	0.1231	0.2559	0	1
12-15	0.0662	0.1304	0	1

12-16	0.0945	0.1987	0	1
14-15	0.2210	0.1997	0	1
16-17	0.0824	0.1923	0	1
15-18	0.1070	0.2185	0	1
18-19	0.0639	0.1292	0	1
19-20	0.0340	0.0680	0	1
10-20	0.0936	0.2090	0	1
10-17	0.0324	0.0845	0	1
10-21	0.0348	0.0749	0	1
10-22	0.0727	0.1499	0	1
21-22	0.0116	0.0236	0	1
15-23	0.1000	0.2020	0	1
22-24	0.1150	0.1790	0	1
23-24	0.1320	0.2700	0	1
24-25	0.1885	0.3292	0	1
25-26	0.2544	0.3800	0	1
25-27	0.1093	0.2087	0	1
27-28	0	0.3960	0	0.968
27-29	0.2198	0.4153	0	1
27-30	0.3202	0.6027	0	1
29-30	0.2399	0.4533	0	1
8-28	0.0636	0.2000	0.0214	1
6-28	0.0169	0.0599	0.0065	1

*Impedance and line-charging susceptance in p.u. on a 100 MVA base.

TABLE I-H: BUS DATA or OPERATING CONDITIONS (30 BUS SYSTEM)

Bus No.	Voltage		Generation		Load	
	Magnitude (in pu)	Phase angle (deg.)	MW	MVAR	MW	MVAR
1*	1.06	0	0	0	0	0
2	1	0	40	0	21.7	12.7
3	1	0	0	0	2.4	
4	1	0	0	0	7.6	
5	1	0	0	0	94.2	
6	1	0	0	0	0	0
7	1	0	0	0	22.8	10.9
8	1	0	0	0	30.0	30.0
9	1	0	0	0	0	0
10	1	0	0	0	5.8	2.0
11	1	0	0	0	0	0
12	1	0	0	0	11.2	7.5
13	1	0	0	0	0	0
14	1	0	0	0	6.2	1.6
15	1	0	0	0	8.2	2.5
16	1	0	0	0	3.5	1.8
17	1	0	0	0	9.0	5.8
18	1	0	0	0	3.2	0.9

19	1	0	0	0	9.5	3.4
20	1	0	0	0	2.2	0.7
21	1	0	0	0	17.5	11.2
22	1	0	0	0	0	0
23	1	0	0	0	3.2	1.6
24	1	0	0	0	8.7	6.7
25	1	0	0	0	0	0
26	1	0	0	0	3.5	2.3
27	1	0	0	0	0	0
28	1	0	0	0	0	0
29	1	0	0	0	2.4	0.9
30	1	0	0	0	10.6	1.9

*Slack Bus

TABLE I-I: REGULATED BUS DATA (30 BUS SYSTEM)

Bus no.	Voltage magnitude (in pu)	Minimum MVAR capability	Maximum MVAR capability
2	1.045	-40	50
5	1.01	-40	40
8	1.01	-10	40
11	1.082	-6	24
13	1.071	-6	24

TABLE I-J: TRANSFORMER DATA (30 BUS SYSTEM)

Transformer designation	Tap setting*
4-12	0.932
6-9	0.978
6-10	0.969
28-27	0.968

*****Off nominal turns ratio, as determined by the actual transformer-tap position and the voltage bases. In the case of nominal turns ratio, this would equal to 1.

TABLE I-K: STATIC CAPACITOR DATA (30 BUS SYSTEM)

Bus no	Susceptance*p.u.
10	0.19
24	0.043

*Susceptance in p.u. on 100 MVA base.

Cost characteristics of IEEE 30 bus system:

The cost characteristics of the IEEE 30 Bus System are as follows:

$C_1 = 50p_1^2+245p_1+105$ \$/hr

$C_2 = 50p_2^2+351p_2+44.4$ \$/hr

$C_8 = 50p_8^2+389p_8+40.6$ \$/hr

The total load demand of the IEEE 30 bus system is 283.4 MW. The maximum active power constraint is 250 MW, 100MW and 100 MW for the generators of bus no. 1, 2 and 8 respectively. The minimum active power constraint is 50 MW, 30MW and 30 MW for the generators of bus no. 1, 2 and 8 respectively. Voltage magnitude constraint for generator at bus 2 is 1.045, for bus no. 5 is 1.01, for bus no. 8 is 1.010, for bus no. 11 is 1.082 &for bus no. 13 is 1.071.

M-file For Calculating B- Coefficients:

```
Clear basemva=100
accuracy=0.0001
maxiter=10
busdata=[1 1 1.06 0 0 0 0 0 0 0 0;2 2 1.045 0 21.7 12.7 90 0 -40 50 0; 3 0 1 0 2.4 1.2 0 0 0 0
0;4 0 1 0 7.6 1.6 0 0 0 0 0;5 0 1.01 0 94.2 19 0 0 -40 40 0; 6 0 1 0 0 0 0 0 0 0 0; 7 0 1 0 22.8
10.9 0 0 0 0 0;8 2 1.010 30 30150 0 -10 40 0; 9 0 1 0 0 0 0 0 0 0 0; 10 0 1 0 5.8 2 0 0 0 0
0.19; 11 0 1.082 0 0 0 0 0 -6 24 0; 12 0 1 0 11.2 7.5 0 0 0 0 0; 13 0 1.071 0 0 0 0 0 -6 24 0; 14
0 1 0 6.2 1.6 0 0 0 0 0;15 0 1 0 8.2 2.5 0 0 0 0 0;16 0 1 0 3.5 1.8 0 0 0 0 0; 17 0 1 0 9 5.8 0 0 0
0 0; 18 0 1 0 3.2 0.9 0 0 0 0 0; 19 0 1 0 9.5 3.4 0 0 0 0 0; 20 0 1 0 2.2 0.7 0 0 0 0 0;21 0 1 0
17.5 11.2 0 0 0 0 0;22 0 1 0 0 0 0 0 0 0 0;23 1 0 3.2 1.6 0 0 0 0 0; 24 0 1 0 8.7 6.7 0 0 0 0
0.043; 25 0 1 0 0 0 0 0 0 0 0;26 0 1 0 3.5 2.3 0 0 0 0 0; 27 0 1 0 0 0 0 0 0 0 0; 28 0 1 0 0 0 0 0
0 0 0;29 0 1 0 2.4 0.9 0 0 0 0 0; 30 0 1 0 10.6 1.9 0 0 0 0 0]; linedata=[1 2 0.0192 0.0575
0.0264 1;1 3 0.0452 0.1852 0.0204 1; 2 4 0.0570 0.19797 0.0219 1; 2 4 0.05811 0.17632
0.0170 1; 2 5 0.05695 0.17388 0.0173 1; 3 4 0.06701 0.17103 0.0064 1; 4 5 0.01335 0.04211
0.0 1; 4 7 0.0 0.20912 0.0 0.978; 4 9 0.0 0.55618 0.0 0.969;5 6 0.0 0.25202 0.0 0.932; 6 11
0.09498 0.19890 0.0 1;6 12 0.12291 0.25581 0.0 1;6 13 0.06615 0.13027 0.0 1;7 8 0.0
0.17615 0.0 1; 7 9 0.0 0.11001 0.0 1; 9 10 0.03181 0.08450 0.0 1;9 14 0.12711 0.27038 0.0
1; 10 11 0.08205 0.19207 0.0 1;12 13 0.22092 0.19988 0.0 1;13 14 0.17093 0.34802 0.0 1];
disp(busdata)
disp(linedata)
mwlimit=[50 150;50 150;50 150]
Ifybus
Ifnewton
busout bloss
```

B-Coefficient Calculated is as:

B11 = 0.0307 B12 = 0.0129 B13 = -0.0002

B21 = 0.0129 B22=0.0152 B23= - 0.0011

B31= -0.0002 B32=- 0.0011 B33= 0.0190

APPENDIX II

MATLAB Program for optimization of benchmark functions using PSO

ROSENBROCK FUNCTION:

```
clc;
clear all;
disp(' we have to minimize f = 100(x1^2-x2)^2+(1-x1)^2 i.e. rosenbrock…
function')
p=input('Enter the no. of particles in a swarm:');       %no. of particles
t=input('Enter the no. to divide in each iteration:');    %no. of particles
pn=p;
ss=t;
%it=input('Enter the no. of iterations:');
it=500;
x1=zeros(p,it+1);
x2=zeros(p,it+1);
v1=zeros(p,it+1);
v2=zeros(p,it+1);
f=zeros(p,it+1);
fp=zeros(1,p);
df=zeros(1,(it));
rp=1;
rg=1;
cp=1;
cg=1;
kount=0;
lemb=1;
T=10;
% T=input('Enter the tolerance value');
tt = input('min no of particles');
% lemb=input('Enter the value of retardationfactor');
x1(:,1)=unifrnd(0,2,1,p);
x2(:,1)=unifrnd(0,2,1,p);
v1(:,1)=unifrnd(0,0.5,1,p);
v2(:,1)=unifrnd(0,0.5,1,p);

for j=1:p
    f(j,1)= 100*(x1(j,1)^2-x2(j,1))^2+(1-x1(j,1))^2 ;
    kount=kount+1;
end
%Initial personal besst values
x1p=x1(:,1);
x2p=x2(:,1);

%for Initial Global best values updation
fmin=min(f(:,1));
for k=1:p
    if f(k,1)==fmin
        gb=k;
    else
    end
end
%Initial global best value
x1g=zeros(p,it+1);
x2g=zeros(p,it+1);
for k=1:p
x1g(k,1) = x1(gb,1);
x2g(k,1) = x2(gb,1);
end
```

```
fgm = min(f(:,1));

% fig=zeros(1,485);
%t=zeros(1,485);
disp(sprintf('This is %d no. of iteration',0));
print = [x1(1:p,1) x2(1:p,1) v1(1:p,1) v2(1:p,1) f(1:p,1)];
    disp('     x1        x2        v1        v2        f')
    disp(print)
fp=f;
for g=1:it
    disp(sprintf('This is %d no. of iteration',g));

w=0.60;
    for j=1:p
        v1(j,(g+1)) = w*v1(j,g) + rp*cp*(x1p(j)-x1(j,g)) + rg*cg*(x1g(j,g)-
x1(j,g));
        v2(j,(g+1)) = w*v2(j,g) + rp*cp*(x2p(j)-x2(j,g)) + rg*cg*(x2g(j,g)-
x2(j,g));
        x1(j,(g+1)) = x1(j,g) + lemb*v1(j,(g+1));
        x2(j,(g+1)) = x2(j,g) + lemb*v2(j,(g+1));
        f(j,(g+1))= 100*(x1(j,g+1)^2-x2(j,g+1))^2+(1-x1(j,g+1))^2 ;
        kount=kount+1;
    end

%To find change in the values of f
    for j=1:p
        df(j,g)= abs(f(j,(g+1))-f(j,g)) ;
    end

%personal best values updation
    for k=1:p
        if f(k,g+1)< fp(k)
            x1p(k)=x1(k,g+1);
            x2p(k)=x2(k,g+1);
            fp(k,1)= f(k,g+1);
        else
        end
    end

%for Global best values updation
    if min(f(1:p,(g+1)))<fgm
        fgm=min(f(1:p,(g+1)));
        X=f(1:p,(g+1));
        k=find(X==min(X));
        x1g(1:p,g+1) = x1(k,g+1);     %global best values
        x2g(1:p,g+1) = x2(k,g+1);
    else
        x1g(1:p,g+1) = x1g(1:p,g);     %global best values
        x2g(1:p,g+1) = x2g(1:p,g);
    end
    print = [x1(1:p,g+1) x2(1:p,g+1) v1(1:p,g+1) v2(1:p,g+1) f(1:p,g+1)];
    disp('     x1        x2        v1        v2        f')
    disp(print)

% Stoping criterion
    ki=0;
    for j=1:p
        if (df(j,g)<=10^(-T))
            ki=ki+1;
        end
    end
```

```
    if ki >= p
        break
    end
      D=[x1(1:p,g+1) x2(1:p,g+1) v1(1:p,g+1) v2(1:p,g+1) x1p(1:p)…
      x2p(1:p) x1g(1:p,g+1) x2g(1:p,g+1) f(1:p,g+1)];
        B=sortrows(D,9);

        p=p/t;

        if p<=tt
            p=tt;
            t=1;
        end

        C=B(1:p,1:9);
       x1(1:p,g+1)=C(:,1);
       x2(1:p,g+1)=C(:,2);
       v1(1:p,g+1)=C(:,3);
       v2(1:p,g+1)=C(:,4);
       x1p(1:p)=C(:,5);
       x2p(1:p)=C(:,6);
       x1g(1:p,g+1)=C(:,7);
       x2g(1:p,g+1)=C(:,8);

      f(1:p,g+1)=C(:,9);
      sn=g;

end
minf=100*(x1g(1,sn).^2-x2g(1,sn)).^2+(1-x1g(1,sn)).^2;
disp (sprintf (' i=%d    kount=%d     p=%d    q=%d   T=%d   it=%d',g,
kount, pn, ss, T, it))
disp(sprintf('F = %d   x1=%d   x2=%d ',minf,x1g(1,sn),x2g(1,sn)))
```

BEALE FUNCTION:

```
clc;
clear all;
disp(' we have to minimize f=((1.5-x1+x1*x2)^2)+((2.25-
x1+x1*(x2^2))^2)+((2.625-x1+x1*(x2^3))^2) i.e. Beales function')
p=input('Enter the no. of particles in a swarm:');       %no. of particles
t=input('Enter the no. of particles to be eliminated in each iteration:');
%no. of particles
pn=p;
ss=t;
it=500;
tt=input('enter the no of particle to select :');
%it=input('Enter the no. of iterations:');
x1=zeros(p,it+1);
x2=zeros(p,it+1);
v1=zeros(p,it+1);
v2=zeros(p,it+1);
f=zeros(p,it+1);
fp=zeros(1,p);
df=zeros(1,(it));
rp=1;
rg=1;
cp=1;
cg=1;
kount=0;
```

```
T=10;
lemb=1;
x1(:,1)=unifrnd(0,2,1,p);
x2(:,1)=unifrnd(0,2,1,p);
v1(:,1)=unifrnd(0,0.5,1,p);
v2(:,1)=unifrnd(0,0.5,1,p);

for j=1:p
    f(j,1)=((1.5-x1(j,1)+x1(j,1)*x2(j,1))^2)+((2.25-
x1(j,1)+x1(j,1)*(x2(j,1).^2))^2)+((2.625-x1(j,1)+x1(j,1)*(x2(j,1).^3))^2);
    kount=kount+1;
end
%Initial personal besst values
x1p=x1(:,1);
x2p=x2(:,1);

%for Initial Global best values updation
fmin=min(f(:,1));
for k=1:p
    if f(k,1)==fmin
        gb=k;
    else
    end
end
%Initial global best value
x1g=zeros(p,it+1);
x2g=zeros(p,it+1);
for k=1:p
x1g(k,1) = x1(gb,1);
x2g(k,1) = x2(gb,1);
end
fgm = min(f(:,1));

disp(sprintf('This is %d no. of iteration',0));
print = [x1(1:p,1) x2(1:p,1) v1(1:p,1) v2(1:p,1) f(1:p,1)];
    disp('     x1        x2        v1        v2        f')
    disp(print)
fp=f;
for g=1:it
    disp(sprintf('This is %d no. of iteration',g));


w=0.60;
    for j=1:p
        v1(j,(g+1)) = w*v1(j,g) + rp*cp*(x1p(j)-x1(j,g)) + rg*cg*(x1g(j,g)-
x1(j,g));
        v2(j,(g+1)) = w*v2(j,g) + rp*cp*(x2p(j)-x2(j,g)) + rg*cg*(x2g(j,g)-
x2(j,g));
        x1(j,(g+1)) = x1(j,g) + lemb*v1(j,(g+1));
        x2(j,(g+1)) = x2(j,g) + lemb*v2(j,(g+1));
        f(j,g+1)=((1.5-x1(j,g+1)+x1(j,g+1)*x2(j,g+1))^2)+((2.25-
x1(j,g+1)+x1(j,g+1)*(x2(j,g+1).^2))^2)+((2.625-
x1(j,g+1)+x1(j,g+1)*(x2(j,g+1).^3))^2);
        kount=kount+1;
    end

%To find change in the values of f
    for j=1:p
        df(j,g)= abs(f(j,(g+1))-f(j,g)) ;
    end
```

```
%personal best values updation

    for k=1:p
        if f(k,g+1)< fp(k)
            x1p(k)=x1(k,g+1);
            x2p(k)=x2(k,g+1);
            fp(k,1)= f(k,g+1);
        else
        end
    end

%for Global best values updation
    if min(f(1:p,(g+1)))<fgm
        fgm=min(f(1:p,(g+1)));
        X=f(1:p,(g+1));
        k=find(X==min(X));
        x1g(1:p,g+1) = x1(k,g+1);     %global best values
        x2g(1:p,g+1) = x2(k,g+1);
    else
        x1g(1:p,g+1) = x1g(1:p,g);     %global best values
        x2g(1:p,g+1) = x2g(1:p,g);
    end
    print = [x1(1:p,g+1) x2(1:p,g+1) v1(1:p,g+1) v2(1:p,g+1) f(1:p,g+1)];
    disp('     x1         x2         v1         v2         f')
    disp(print)

% Stoping criterion
    ki=0;
    for j=1:p
        if (df(j,g)<=10^(-T))
            ki=ki+1;
        end
    end
    if ki >= p
        break
    end
        D=[x1(1:p,g+1) x2(1:p,g+1) v1(1:p,g+1) v2(1:p,g+1) x1p(1:p)
x2p(1:p) x1g(1:p,g+1) x2g(1:p,g+1) f(1:p,g+1)];
        B=sortrows(D,9);
        p=p/t;
        if p<=tt
            p=tt;
            t=1;
        end

        C=B(1:p,1:9);
       x1(1:p,g+1)=C(:,1);
       x2(1:p,g+1)=C(:,2);
       v1(1:p,g+1)=C(:,3);
       v2(1:p,g+1)=C(:,4);
       x1p(1:p)=C(:,5);
       x2p(1:p)=C(:,6);
       x1g(1:p,g+1)=C(:,7);
       x2g(1:p,g+1)=C(:,8);

      f(1:p,g+1)=C(:,9);
      sn=g;

end
```

```
minf=((1.5-x1g(1,sn)+x1g(1,sn)*x2g(1,sn))^2)+((2.25-
x1g(1,sn)+x1g(1,sn)*(x2g(1,sn).^2))^2)+((2.625-
x1g(1,sn)+x1g(1,sn)*(x2g(1,sn).^3))^2);
disp (sprintf (' i=%d    kount=%d     p=%d    t=%d   q=%d   T=%d
it=%d',g, kount, pn, ss, tt, T, it))
disp(sprintf('F = %d   x1=%d   x2=%d ',minf,x1g(1,sn),x2g(1,sn)))
```

MATLAB Program for the solution of IEEE 30-bus system using PSO

```
clc;
clear all;
disp(' we have to minimize ELD 30 BUS SYSTEM ')
p=input('Enter the no. of particles in a swarm:');       %no. of particles
t=input('Enter the no. to divide in each iteration:')    %no. of particles
a=10^(-4)*[50 50 50];
b=10^(-2)*[245 351 389];
c=[105 44.4 40.6];
B=10^(-2)*[0.0307 0.0129 -0.0002; 0.0129 0.0152 -0.0011; -0.0002 -0.0011
0.0190];
pn=p;
ss=t;
%it=input('Enter the no. of iterations:');
it=1000;
p1=zeros(p,it);
p2=zeros(p,it);
p3=zeros(p,it);
v1=zeros(p,it);
v2=zeros(p,it);
v3=zeros(p,it);
f=zeros(p,it);
fp=zeros(1,p);
df=zeros(1,(it));
sp=zeros(p,it);
csp=zeros(p,it);
pl=zeros(p,it);
c1=zeros(p,it);
c2=zeros(p,it);
c3=zeros(p,it);
C=zeros(p,it);

rp=1;
rg=1;
cp=1;
cg=1;
kount=0;
lemb=1;
T=6;
pd=283.4;

    plp=zeros(1,p);
    z=500;
% T=input('Enter the tolerance value');
tt = input('min no of particles');
% lemb=input('Enter the value of retardationfactor');
n=1;
while n==1
    for j=1:p
```

```
    p1(j,1)=unifrnd(50,250,1);
    p2(j,1)=unifrnd(30,100,1);
    p3(j,1)=pd-p1(j,1)-p2(j,1);
    if p3(j,1)<30&&p3(j,1)>100
        n=1;
        break;
    else
        n=0;
    end
    end
end
v1(:,1)=unifrnd(0,0.5,1,p);
v2(:,1)=unifrnd(0,0.5,1,p);
v3(:,1)=unifrnd(0,0.5,1,p);

    for j=1:p
         c1(j,1) = a(1)*(p1(j,1))^2 + b(1)*p1(j,1) + c(1);
         c2(j,1) = a(2)*(p2(j,1))^2 + b(2)*p2(j,1) + c(2);
         c3(j,1) = a(3)*(p3(j,1))^2 + b(3)*p3(j,1) + c(3);
         C(j,1) = c1(j,1) + c2(j,1) + c3(j,1);
    end
%To calculate initial value of cost function we need PL
for j=1:p
    pl(j,1)= [p1(j,1) p2(j,1) p3(j,1)]*B*[p1(j,1) p2(j,1) p3(j,1)]';
%    kount=kount+1;
end
%To calculate initial value of cost function
for j=1:p
    f(j,1)=((a(1)*(p1(j,1))^2 + b(1)*p1(j,1) + c(1)) + (a(2)*(p2(j,1))^2 +
b(2)*p2(j,1) + c(2)) ...
           + (a(3)*(p3(j,1))^2 + b(3)*p3(j,1) + c(3))) + z*abs(pd+pl(j,1)-
p1(j,1)-p2(j,1)-p3(j,1));
        kount=kount+1;
end
%0th iteration data display
    disp('this is the 0th iteration')
    print0 = [p1(:,1) p2(:,1) p3(:,1) v1(:,1) v2(:,1) v3(:,1) f(:,1)
c1(:,1) c2(:,1) c3(:,1) C(:,1)];
    disp('      P1          P2         P3         V1          v2         V3
f         c1         c2         c3         C ')
    disp(print0)
%Initial personal besst values
p1p=p1(:,1);
p2p=p2(:,1);
p3p=p3(:,1);
%for Initial Global best values updation
fmin=min(f(:,1));
for k=1:p
    if f(k,1)==fmin
        gb=k;
    else
    end
end
%Initial global best value
p1g=zeros(p,it);
p2g=zeros(p,it);
p3g=zeros(p,it);

for k=1:p
p1g(k,1) = p1(gb,1);
p2g(k,1) = p2(gb,1);
p3g(k,1) = p3(gb,1);
```

```
end
fgm = min(f(:,1));

fp=f;
for g=1:it
    disp(sprintf('This is %d no. of iteration',g));

w=0.60;
    for j=1:p
        v1(j,(g+1)) = w*v1(j,g) + rp*cp*(p1p(j)-p1(j,g)) + rg*cg*(p1g(j,g)-
p1(j,g));
        v2(j,(g+1)) = w*v2(j,g) + rp*cp*(p2p(j)-p2(j,g)) + rg*cg*(p2g(j,g)-
p2(j,g));
        v3(j,(g+1)) = w*v3(j,g) + rp*cp*(p3p(j)-p3(j,g)) + rg*cg*(p3g(j,g)-
p3(j,g));
    end
%V(min) and V(max) constraint
    for j=1:p
        if v1(j,(g+1))< -15
            v1(j,(g+1))= -15;
        end
        if v2(j,(g+1))< -15
            v2(j,(g+1))= -15;
        end
        if v3(j,(g+1))< -15
            v3(j,(g+1))= -15;
        end
        if v1(j,(g+1))> 60
            v1(j,(g+1))= 60;
        end
        if v2(j,(g+1))> 60
            v2(j,(g+1))= 60;
        end
        if v3(j,(g+1))> 60
            v3(j,(g+1))= 60;
        end
    end
for j=1:p
        p1(j,(g+1)) = p1(j,g) + lemb*v1(j,(g+1));
        p2(j,(g+1)) = p2(j,g) + lemb*v2(j,(g+1));
        p3(j,(g+1)) = p3(j,g) + lemb*v3(j,(g+1));
end
%Pmin and Pmax constraint
    for j=1:p
        if p1(j,(g+1))< 50
            p1(j,(g+1))= 50;
        end
        if p2(j,(g+1))< 20
            p2(j,(g+1))= 20;
        end
        if p3(j,(g+1))< 20
           p3(j,(g+1))= 20;
        end
        if p1(j,(g+1))> 250
            p1(j,(g+1))= 250;
        end
        if p2(j,(g+1))>100
            p2(j,(g+1))= 100;
        end
        if p3(j,(g+1))> 100
            p3(j,(g+1))= 100;
        end
```

```
    end
    %For losses formulation (PL)
    for j=1:p
        pl(j,(g+1))= [p1(j,(g+1)) p2(j,(g+1)) p3(j,(g+1))]*B*[p1(j,(g+1))
p2(j,(g+1)) p3(j,(g+1))]';
       % kount=kount+1;
    end

%Main objective function
   for j=1:p
        f(j,(g+1))= (a(1)*(p1(j,(g+1)))^2 + b(1)*p1(j,(g+1)) + c(1)) +...
                     (a(2)*(p2(j,(g+1)))^2 + b(2)*p2(j,(g+1)) + c(2))+...
                     (a(3)*(p3(j,(g+1)))^2 + b(3)*p3(j,(g+1)) + c(3)) +...
                     + z*abs(pd+pl(j,(g+1))-p1(j,(g+1))-p2(j,(g+1))-
p3(j,(g+1)));
        kount=kount+1;
   end

%personal best values updation
    for k=1:p
        if f(k,g+1)< fp(k)
            p1p(k)=p1(k,g+1);
            p2p(k)=p2(k,g+1);
            p3p(k)=p3(k,g+1);
            fp(k,1)= f(k,g+1);
        else
        end
    end

%for Global best values updation
    if min(f(1:p,(g+1)))<fgm
        fgm=min(f(1:p,(g+1)));
        X=f(1:p,(g+1));
        k=find(X==min(X));
        p1g(1:p,g+1) = p1(k,g+1);     %global best values
        p2g(1:p,g+1) = p2(k,g+1);
        p3g(1:p,g+1) = p3(k,g+1);
    else
        p1g(1:p,g+1) = p1g(1:p,g);     %global best values
        p2g(1:p,g+1) = p2g(1:p,g);
        p3g(1:p,g+1) = p3g(1:p,g);
    end
    for j=1:p
         c1(j,(g+1)) = a(1)*(p1(j,(g+1)))^2 + b(1)*p1(j,(g+1)) + c(1);
         c2(j,(g+1)) = a(2)*(p2(j,(g+1)))^2 + b(2)*p2(j,(g+1)) + c(2);
         c3(j,(g+1)) = a(3)*(p3(j,(g+1)))^2 + b(3)*p3(j,(g+1)) + c(3);
         C(j,(g+1)) = c1(j,(g+1)) + c2(j,(g+1))+c3(j,(g+1));



    end
    for j=1:p
        df(j,g)= abs(f(j,(g+1))-f(j,g));
        sp(j,g)= abs(pd+pl(j,(g+1))-p1(j,(g+1))-p2(j,(g+1)));
      csp(j,g)= abs(C(j,(g+1))-C(j,g));
     end

    print = [p1(1:p,(g+1)) p2(1:p,(g+1)) p3(1:p,(g+1)) v1(1:p,(g+1))…
v2(1:p,(g+1)) v3(1:p,(g+1)) f(1:p,(g+1)) c1(1:p,(g+1)) c2(1:p,(g+1))…
c3(1:p,(g+1))  C(1:p,(g+1))];
    disp('      P1          P2          P3         V1          V2         V3 …
f         c1          c2          c3          C ')
```

```
    disp(print)
   %Stoping criterion
    ki=0;
    for j=1:p
        if ((df(j,g)<=10^(-T)))
            ki=ki+1;
        end
    end
    if ki >= p
        break
    end

        D=[p1(1:p,g+1) p2(1:p,g+1) p3(1:p,g+1) v1(1:p,g+1) v2(1:p,g+1)…
v3(1:p,g+1) p1p(1:p) p2p(1:p) p3p(1:p) p1g(1:p,g+1) p2g(1:p,g+1)…
p3g(1:p,g+1) f(1:p,g+1)];
        G=sortrows(D,13);

        p=p/t;

        if p<=tt
            p=tt;
            t=1;
        end

        Y=G(1:p,1:13);
       p1(1:p,g+1)=Y(:,1);
       p2(1:p,g+1)=Y(:,2);
       p3(1:p,g+1)=Y(:,3);
       v1(1:p,g+1)=Y(:,4);
       v2(1:p,g+1)=Y(:,5);
       v3(1:p,g+1)=Y(:,6);
       p1p(1:p)=Y(:,7);
       p2p(1:p)=Y(:,8);
       p3p(1:p)=Y(:,9);
       p1g(1:p,g+1)=Y(:,10);
       p2g(1:p,g+1)=Y(:,11);
       p3g(1:p,g+1)=Y(:,12);
       f(1:p,g+1)=Y(:,13);
      sn=g;

end
disp(' we have to minimize the cost function of a 3 machine system')
disp (sprintf (' i=%d    kount=%d     p=%d    q=%d   it=%d',g, kount, pn,
ss, it))
disp(sprintf('Total demand of power Pd = %d \n',pd))
disp(sprintf('Total loses in the lines Pl = %d \n',pl(1,sn)))
disp(sprintf('Minimum cost incured = %d \n',C(1,sn)))

disp('Final values of generations of the three generators')
disp(sprintf('\nP1=%d',p1(1,sn)))
disp(sprintf('P2=%d',p2(1,sn)))
disp(sprintf('P3=%d',p3(1,sn)))
disp(sprintf('\nPD+Pl = %d',pd+pl(1,sn)))
disp(sprintf('\nP1+P2+P3=%d\n',p1(1,sn)+p2(1,sn)+p3(1,sn)))

disp(sprintf('Z taken = %d',z))
```