Performance analysis of Improving Transient Stability of a Plant Using Partical Swarm Optimization and Bacterial Foraging Optimization Technique

[1]Chitransh Tiwari, [2]Ram Kumar Denjare

[1]MTech Scholar, [2]Assistant Professor
Electrical and Electronics Engineering Department,
Kalinga University Raipur, India

***Abstract-* In our daily life the application of Control systems appear practically everywhere in our homes, in industry, in communications, information technologies, etc. PID controllers are the most widely-used type of controller for industrial applications. From the constructional point of view they are simple and exhibit robust performance over a wide range of operating conditions. There are three main parameters involved are Proportional (P), Integral (I) and Derivative (D). The proportional part is responsible for following the desired set-point, while the integral and derivative part account for the buildup of past errors and the rate of change of error in the process respectively. Particle Swarm Optimization (PSO) and Bacterial Foraging Optimization (BFO) are two bio-inspired optimization techniques which have gained popularity recently in field of optimum tuning. These two methods are discussed along with their detailed algorithm structure. BFO and PSO algorithms are implemented for optimum tuning**

***Index Terms*— Plant, PID controller, Ziegler-Nichols continuous cycling method. Particle Swarm Optimization, Bacterial Foraging Optimization**

I. INTRODUCTION

The Proportional-Integral-Derivative (PID) controller has been proved the most popular controller of this century for its remarkable effectiveness, easiness of implementation and vast applicability. But it is also hard to tune the PID controller. A number of tuning methods are done manually which are difficult and time consuming. For using PID controller efficiently, the optimal tuning of its parameters has become a significant research area. Optimization problems have been resolved with the aid of numerous techniques. Today's, an alternative approach to the traditional methods for operations research, meta-heuristic methods are implanted to simplify optimization difficulties..

Artificial Intelligence (AI) based techniques are the new entry in field of optimum tuning. AI based techniques provides a better approach to reach the terms of stability. The objective of thesis is to give a comparative analysis of performance of Particle Swarm Optimization and Bacteria Forging Optimization for finding out the stability criterion. Artificial Intelligence provides a better approach for optimum tuning of the system to limit the stability of the unstable system. Matlab is used to analyze the performance of PSO and BFO for stability enhancement of the system. The comparative study is based on the tuning parameters obtained for controller by the two techniques namely PSO and BFO; The introduction of Artificial Intelligence tuning based methods is popular now days over conventional tuning methods due to their higher performance and faster speed.Particle Swarm Optimization (PSO) and Bacterial Foraging Optimization (BFO) are two bio-inspired optimization techniques which have gained popularity recently in field of optimum tuning. These two methods are discussed along with their detailed algorithm structure. BFO and PSO algorithms are implemented for optimum tuning.

II.TUNING OF PID CONTROLLER

The PID controller calculation involves three separate control parameters, i.e. proportional, integral and derivative values .The proportional value determines the reaction of the current error, the integral value determines the reaction based on the sum of recent errors and derivative value determines the reaction based on the rate at which the error has been changing and the weighted sum of these three actions is used to adjust the process via the final control element.
The block diagram of a control system with unity feedback employing Soft computing PID control action is shown in Figure-1.

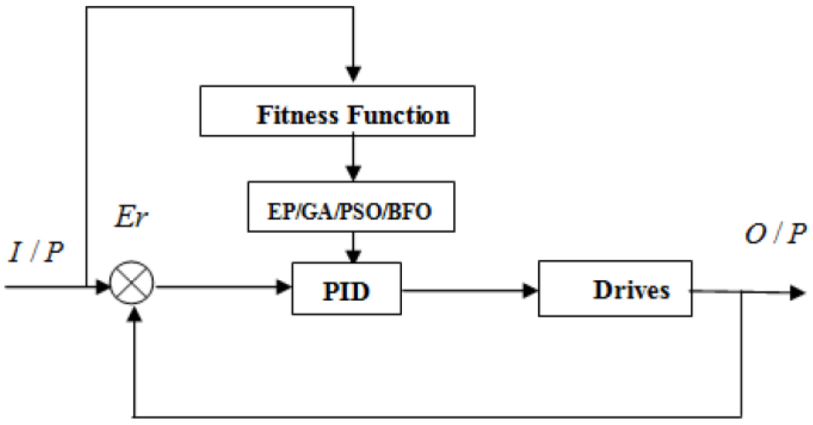


Fig 2: PID Tuning

The mathematical representation of PID control is given in (1).

$$u(t) = K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right] \qquad (1)$$

A. *Classical Techniques*

Classical techniques make certain assumptions about the plant and the desired output and try to obtain analytically, or graphically some feature of the process that is then used to decide the controller settings. These techniques are computationally very fast and simple to implement, and are good as a first iteration. But due to the assumptions made, the controller settings usually do not give the desired results directly and further tuning is required. A few classical techniques have been reviewed in this paper.

B. *Computational or Optimization Techniques*

These are techniques which are usually used for data modeling and optimization of a cost function, and have been used in PID tuning. Few examples are neural networks (computational models to simulate complex systems), genetic algorithm and differential evolution. The optimization techniques require a cost function they try to minimize. There are four types of cost functions used commonly,

- Integral Absolute Error

$$\mathrm{IAE} = \int_0^{\tau} |\mathrm{e(t)}|$$

- Integral Square Error

$$\mathrm{ISE} = \int_0^{\tau} |\mathrm{e(t)}|^2$$

- Integral Time Absolute Error

$$\mathrm{ITAE} = \int_0^{\tau} \mathrm{t}|\mathrm{e(t)}|$$

- Integral Time Square Error

$$\mathrm{ITSE} = \int_0^{\tau} \mathrm{t}|\mathrm{e(t)}|^2$$

Computational models are used for self tuning or auto tuning of PID controllers. Self tuning of PID controllers essentially sets the PID parameters and also models the process by using some computational model and compares the outputs to see if there are any process variations, in which case the PID parameters are reset to give the desired response. The existent types of adaptive techniques are classified based on the fact that if the process dynamics are varying [3], then the controller should compensate these variations by adapting its parameters. There are two types of process dynamics variations, predictable and unpredictable. The predictable ones are typically caused by nonlinearities and can be handled using a gain schedule, which means that the controller parameters are found for different operating conditions with an auto-tuning procedure that is employed thereafter to build a schedule. Different techniques have been used to replace the gain schedule mentioned above.

Computational models are used for self tuning or auto tuning of PID controllers. Self tuning of PID controllers essentially sets the PID parameters and also models the process by using some computational model and compares the outputs to see if there are any process variations, in which case the PID parameters are reset to give the desired response. The existent types of adaptive

techniques are classified based on the fact that if the process dynamics are varying [3], then the controller should compensate these variations by adapting its parameters. There are two types of process dynamics variations, predictable and unpredictable. The predictable ones are typically caused by nonlinearities and can be handled using a gain schedule, which means that the controller parameters are found for different operating conditions with an auto-tuning procedure that is employed thereafter to build a schedule. Different techniques have been used to replace the gain schedule mentioned above. In the discussion of various

III.REASON FOR SELECTING SOFT COMPUTING TECHNIQUES

Optimization techniques like Genetic Algorithm (GA), Simulated Annealing (SA), Particle Swarm Optimization (PSO) and Bacterial Foraging Optimization (BFO) belonging to the family of evolutionary computational algorithms have been widely used in many control engineering applications. These are powerful soft computing techniques which create a set of potential solutions called as populations. EP, GA, PSO and BFO found the optimal solution through cooperation and competition among potential solutions. These algorithms are highly relevant for industrial applications, because they are capable of handling problems with non linear constraints, multiple objectives and dynamic properties of the components that frequently appear in real-world problem.
The advantages of using heuristic techniques for PID are listed below.
a) Heuristic Techniques can be applied for higher order systems without model reduction.
b) These methods can also optimize the design criteria such as gain margin, Phase margin, closed loop bandwidth when the system is subjected to step and load change.

3.1. CLASSICAL TECHNIQUES

Most classical techniques make assumptions of the plant model and try to derive the controller settings for these general models. To determine the dynamics of these systems, the step response of the systems are obtained. This response is characterized by different equations, using which different classical methods have been developed. Ziegler and Nichols [5] proposes that many industrial processes have step response as given in the Fig. 1. Where K being the static gain, θ the dead time and τ1 the time constant. The parameter a is determined by the intercept of the line (tangent to the graph) with the y-axis and θ the x intercept.

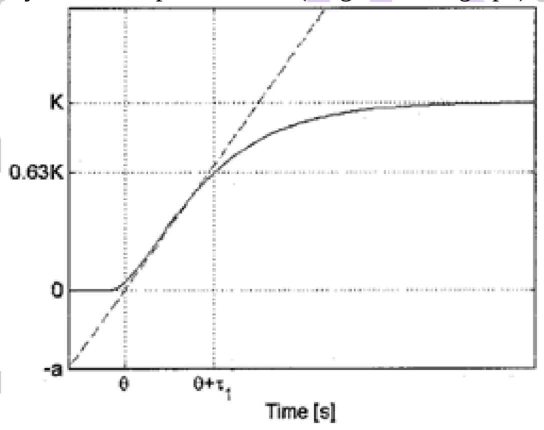


Fig. a shows Step response of first order system

A. Ziegler Nichols Method

This is by far the most popular tuning method in use. It was proposed by John Ziegler and Nathaniel Nichols [5] in 1942 and is still a simple, fairly effective PID tuning method. There are two methods proposed by Ziegler and Nichols.
The second method is based on knowledge of the response to specific frequencies. The idea is that the controller settings can be based on the most critical frequency points for stability. This method is based on experimentally determining the point of marginal stability. This frequency can be found by increasing the proportional gain of
the controller, until the process becomes marginally stable. These two parameters define one point in the Nyquist plot. The gain is called ultimate gain Ku and the time period Tp. The PID parameter setting is given in [5]. The Ziegler and Nichols method is the first PID tuning techniques made and they are made based on certain controller assumptions. Hence, there is always a requirement of further tuning; because the controller settings derived are rather aggressive and thus result in excessive overshoot and oscillatory response. Also for the first method the parameters are rather difficult to estimate in noisy environment. In the second method, as the system is driven towards instability for determining the parameters, practically this can be quite detrimental to the system.

B. Cohen Coon Method

Cohen and Coon [6] design a method with the PID controller parameters decided based on a FOLPD model. The main design requirement is the rejection of load disturbances. The controller parameter settings are given in [6].Despite a better model, the results of the Cohen Coon method are not much better than the Ziegler Nichols method.

IV. PARTICLE SWARM OPTIMIZATION

The concept of Particle Swarm Optimization is by the flocking and schooling patterns of birds and fish, Particle Swarm Optimization (PSO) was invented by Russell Eberhart and James Kennedy in 1995. This approach came to in existence by developing computer software simulations of birds flocking around food sources, and then later realized how well their algorithms worked on optimization problems.Particle Swarm Optimization is very complicated but it is a reliable technique. After a number of iteration it gives the desired value of gain parameter. Let us take an example a group of fish were roaming in the sea in the search of food in alternate direction. If any of the fish found the food it will gives information to the other fish and after that the fish gives the information to the other. The cycle of giving information will continue till the last fish found the food. Hence the concept shows that if one of the fish found the food it is the best area where it found the food. If it is found in the minimum distance covered then it said to be the best position of that fish and it is known as the individual best position of the fish or in terms of particle it is the best position of the particle known as particle best (pbest). If in the group of fish every fish find the found in all direction and they all are having their best position but according to the rule the minimum distance covered by the fish is the best so for that from the swarm of best position if any of the fish covers minimum distance from all the fish, then it is said to be the global best (gbest).

The algorithm keeps track of three global variables:

Condition or target values

Global best (gBest) value indicates that which particle's data is currently closest to the Target

Stopping value indicates that the particle is found the food and all the particles are in the same direction and after that it stops the criterion.

Each particle consists of:

Each and every data signifying a possible solution by adapting such type of algorithm.

After applying the algorithm how much the data should be changed can maintain by a factor that is velocity, each particles may have their own velocity.

One of the factors is the personal best indicating that each and every particle may experience their best position during the roaming period.

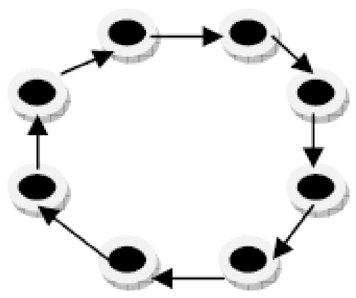
The value changes according to the random values. In the flocking birds example above, providing data in each coordinate which is a three dimensional coordinate roaming according to their velocity. The individual coordinates of each bird would try to come closer to the coordinates of the bird which is closer to the food's coordinates (gBest). If it follows the pattern or sequence then it will provide a better data.The velocity of each bird or particles is calculated according to their random natures of position. If any bird or particle is much far from the food or say target then it will have to fly much more effort than the closer one so it is having high velocity then the others. It tries to come closer towards the food or target with a high velocity. In the birds example, the individuals furthest from the food would make an effort to keep up with the others by flying faster toward the gBest bird. If the data is a pattern or sequence, then the velocity would describe how different the pattern is from the target, and thus, how much it needs to be changed to match the target. The gBest factor is totally depends upon the pBest value, if any particle came closer to the target than ultimately there is variation in the value of gBest, gBest tries to move closer to the target until it reached the position of the target.

In the programming part three columns are initialized and named it as kp ki and kd which are the gain parameters of proportional, integral and derivatives. After several iterations the value of the gain parameters are find their best position. Let us assume that i = 50,100,150,200 (iterations of the PSO) in which the gain parameters are find their best positions, the best gain value is the minimum value or minimum approach by the controller to reach the stability limit.

It is also common to see PSO algorithms using population topologies, or "neighbourhoods", which can be smaller, localized subsets of the global best value. These neighbourhoods can involve two or more particles which are predetermined to act together, or subsets of the search space that particles happen into during testing. The use of neighbourhoods often helps the algorithm to avoid getting stuck in local minima.Exploration and exploitation are the two phenomena which a plays an important role in the field of PSO algorithm. If the iteration is small than it can evaluate in a small form or in a small area then that is known as exploitation which is covering the whole area. if the area is large or say iteration is very large then evaluate the value in the big platform this is known to be exploration.The possibility of getting more accurate value is in exploration type rather than exploitation because more and more iteration may give accurate and proper data. But sometime it is time taking to evaluate each and every iteration.

V.SWARM TOPOLOGY

Each particle i (total number of particles) has its neighbourhood Ni (a subset of P). The arrangement of the neighbourhoods is called the swarm topology, which can be represented by a graph. Usual topologies are: Fully connected topology, Circle topology and single sighted topology.



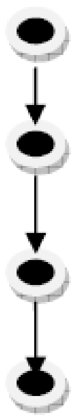


Fig (a). Ring topology, where each individual compares only to those to the left and right.
Fig (b). Single-sighted, where individuals only compare themselves to the next best.

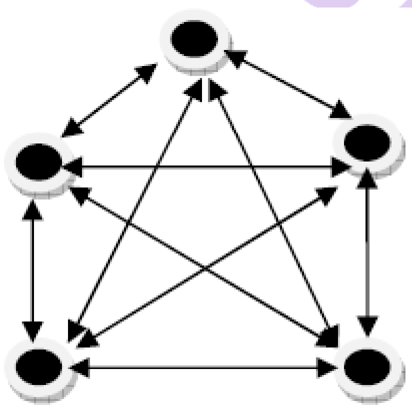


Fig (c). fully connected topology, where everyone is compared together.

Characteristics of particle i at iteration t:

- $x_i^{(t)}$- the position
- $p_i^{(t)}$-the individual best position
- $l_i^{(t)}$- the local best position of the neighbouring particles
- $v_i^{(t)}$- the velocity of the particle

At the beginning of the algorithm, the particle positions are randomly initialized, and the velocities are set to 0, or to small random values.

5.1 Algorithm parameters:

w(t) - Inertia weight; a damping factor, usually decreasing from around 0.9 to around 0.4 during the computation
$\emptyset 1$, $\emptyset 2$ - Acceleration coefficients; usually between 0 and 4.

5.2 Manipulating its velocity:

Many versions of the particle speed update exist, for example:

$$v_i^{(t+1)} = \omega^{(t)} v_i^{(t)} + \emptyset_1 c\left(p_i^{(t)} - x_i^{(t)}\right) + \emptyset_2 c_2\left(l_i^{(t)} - x_i^{(t)}\right).$$

The symbols u1 and u2 represent random variables with the C(0,1) distribution. The first part of the velocity formula is called “inertia”, the se cond one “the cognitive (personal) component”, the third one is “the social (neighbour hood) component”. Position of particle i changes according to

$$x_i^{(t+1)} = x_i + v_i^{(t+1)}$$

5.3 Stopping criteria

After applying all the acceleration factors and velocity constant, it will provide better gain objectives. The algorithm is terminated after a given number of iterations, or once the fitness values of the particles (or the particles themselves) are close enough in some sense.

5.4 Basic Fundamentals of PSO Algorithm

The basic fundamentals of the PSO algorithm technique are stated and defined as follows:

Particle X(i): A candidate solution represented by a k-dimensional real-valued vector, where k is the number of optimized parameters; at iteration i, the jth particle X(i,j) can be described as

Xj(i) = [xj,1(i); xj,2(i); . . . .;xj,k(i). . . . .; xj,d(i)]
Where: xs are the optimized parameters
Xk(i,j) is the kth optimized parameter in the jth candidate solution d represents the number of control variables.

Population: This is the set or say population of n particles at iteration i.

$$\text{pop}(i) = \left[X_{i(j)}, X_{2(i)} \cdots\cdots\cdots X_{n(i)}\right]^T$$

Where n represent the number of individual solutions.

Swarm: As the name swarm defines the group of particles, it is a disorganized population of moving particles that tends to cluster together and each particle seems to moving in a random direction with their velocity.

Particle velocity V(i): The velocity of the moving particles represented by a d-dimensional real-valued vector; at iteration i, the jth particle Vj(i) can be described as:

Vj(i) = [vj,1(i); vj,2(i); . . . . . . vj,k(i); . . . . . ; vj,d(i)]

Weight factor w(i): This is a control parameter and sometimes it is called as inertia weight factor. This is controllable parameter, used to control the contact of the previous velocity on the current velocity. Hence, it influences the trade off between the exploration and local exploitation of the particles. For initial stages of the search process, a large weight factor to enhance exploration is recommended whereas it should be reduced at the last stages for better exploitation. Therefore, the inertia factor decreases linearly from about 0.9 to 0.4 during a continuous run.

In general, this factor is set according to following equation:

$$W = \text{wmax} - (\text{wmax} - \text{wmin})/\text{itermax} * \text{iter}$$

Individual best X*(i): During the movement of a particle through the search space, it compare its fitness value at the current position to the best fitness value it has ever

reached at any iteration up to the current iteration. The best position that is associated with the best fitness encountered thus far is called individual best X*(i). For each particle in the swarm, X*(i) can be determined and updated during the search. For the jth particle, individual best can be expressed as

Xj*(i) = [xj,1(i), xj,2*(i),. . . . . xj,d*(i)]T

In a minimization problem with only one objective function f, the individual best of the jth particle Xj*(i) is updated. Otherwise individual best solution of the jth particle will be kept as in the previous iteration.

Global best X**(t): This is the best position among all of the individual best positions achieved so far.

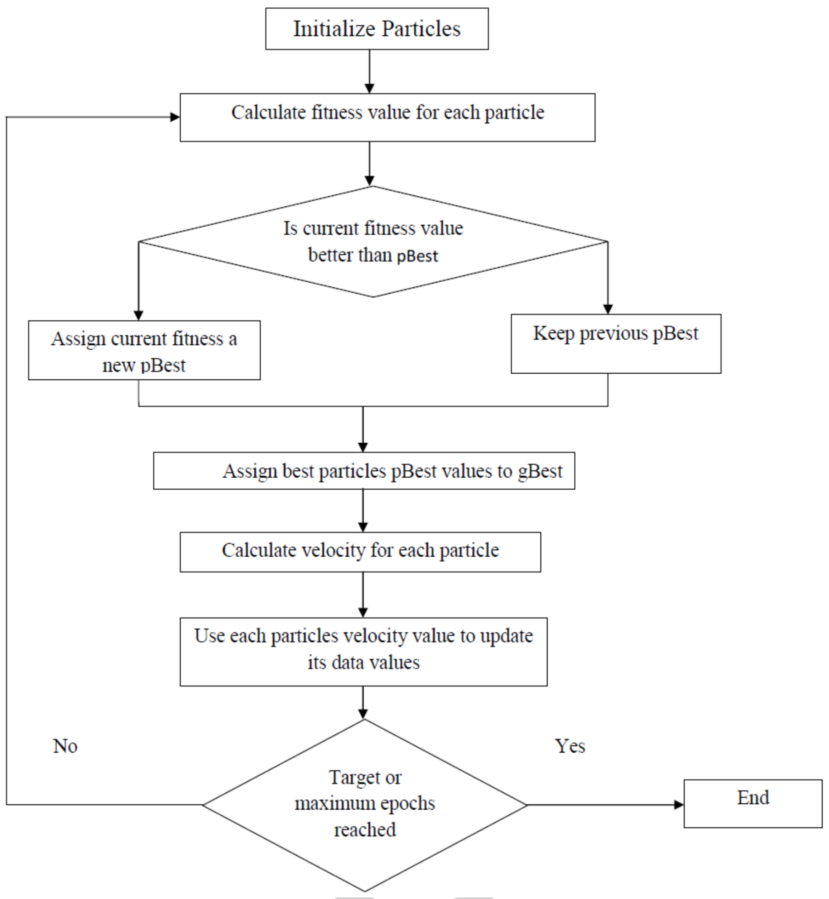
Stopping Criteria: The search process will be terminated whenever one of the following criteria is satisfied.

The number of iteration since the last change of the best solution is greater than a pre specified number.

The number of iterations reaches the maximum allowable number.

By using above all parameters one can design the PID controller model with the help of PSO PID techniques. For designing the PSO PID controller execution and stopping criteria is most important. For initializing any system it have to be considering several factors i.e. number of iteration from where the manipulation of the gain value is to be considered. Then the weighting factor provided in the initialized value with some acceleration as well as velocity to achieve the best position to get the accurate gain value, in other words the best position i.e. achieved by the system after execution is the required value of the gain.

5.6 Flow chart of PSO


Initialize Particles
Calculate fitness value for each particle
Is current fitness value better than pBest
Assign current fitness a new pBest
Keep previous pBest
Assign best particles pBest values to gBest
Calculate velocity for each particle
Use each particles velocity value to update its data values
No
Yes
Target or maximum epochs reached
End

Basic code for PSO

```
For every particle
{
Initialize each and every particle
}

Do until maximum iterations or minimum error criteria are achived
{
For each and every particle
{

Calculate Data and there fitness value If the fitness value is better than pBest
{
Set pBest = current fitness value
}
If pBest is better than gBest
{
```

```
Set gBest = pBest
}
}

For each and every particle
{
Calculate particle Velocity
Use gBest and Velocity to update particle Data(velocity as well as acceleration cofficient)
}
```

From the above code it is clear it will follow the concept of flow chart that during the first procedure the particle has to be providing some position in a 3-diamansional coordinate. And during each and every iteration the values of the gain parameters is to be evaluated. During that process the best values are came as an output. Suppose the best value is came after the 100th iteration than it will compared with the 50th iteration and by their comparison the algorithm provides the value of the required gain. The group of all the individual best position is called as population. And the best value came out from these population is known to be the gbest (global best), whereas the individual best known as pbest. As the theory states that the particle is changing its position after each iterations by the means of velocity and every particles have their different velocity when they are changing there position in the free space.

VI.BACTERIAL FORAGING ALGORITHM (BFA)

Recently, bacterial foraging algorithm (BFA) has emerged as a powerful technique for the solving optimization problems. BFA mimics the foraging strategy of *E. coli* bacteria which try to maximize the energy intake per unit time.

The E. coli bacterium has a plasma membrane, cell wall, and capsule that contains the cytoplasm and nucleoid (Figure 1). The pili (singular, pilus) are used for a type of gene transfer to other E. coli bacteria, and flagella (singular, flagellum) are used for locomotion . The cell is about 1 μmin diameter and 2 μm in length. The E. coli cell only weighs about 1 picogram and is about 70% water. Salmonella typhimurium is a similar type of bacterium .

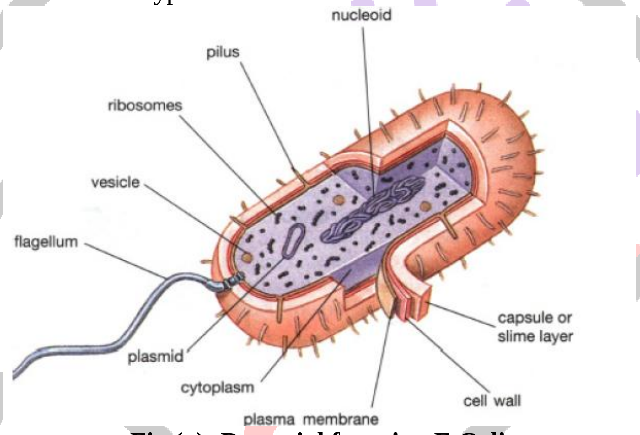


Fig (a): Bacterial foraging E.Coli

From the very early days it has drawn attention of researchers due to its effectiveness in the optimization domain. So as to improve its performance, a large number of modifications have already been undertaken. The bacterial foraging system consists of four principal namely chemotaxis, swarming, reproduction and elimination-dispersal. A brief description of each of these processes along with the pseudo-code of the complete algorithm is described below.

Chemotaxis: This process simulates the movement of an *E.coli* cell through swimming and tumbling via flagella. Biologically an *E.coli* bacterium can move in two different ways. It can swim for a period of time in the same direction or it may tumble, and alternate between these two modes of operation for the entire lifetime. Suppose $\theta^i(j,k,l)$represents i^{th} bacterium at j^{th}chemotactic, k^{th} reproductive and 1^{th} elimination-dispersal step. C(i) is the size of the step taken in the random direction specified by the tumble (run length unit).Then in computational chemotaxis the movement of the bacterium may be represented by

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + c(i)\frac{\Delta(i)}{\sqrt{\Delta^t(i)+\Delta(i)}} \qquad (1)$$

Where Δ indicates a vector in the random direction whose elements lie in [-1, 1].

Swarming: An interesting group behavior has been observed where a group of *E.coli* cells arrange themselves in a traveling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemoeffecter. The cells, when stimulated by a high level of *succinate*, release an attractant *aspertate*, which helps them to aggregate into groups and thus move as concentric patterns of swarms with high bacterial density. The cell-to-cell signaling in *E. coli* swarm may be represented by the following function.

$$J_{CC}(\theta, P(j,k,l)) = \sum_{i=1}^{s} J_{CC}\left(\theta, \theta^{i}(j,k,l)\right)$$

$$J_{CC} = \sum_{i=1}^{S}\left[-d_{attrac\ \tan t}\, e^{\left(-w_{attrac\ \tan t}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)}\right] + \sum_{i=1}^{S}\left[h_{repellant}\, e^{\left(-w_{repellant}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)}\right] \quad (2)$$

where $J_{CC}\left(\theta, P(j,k,l)\right)$ is the objective function value to be added to the actual objective function (to be minimized)to present a time varying objective function, *S* is the total number of bacteria, *p* is the number of variables to be optimized, which are present in each bacterium and $\theta = \left(\theta_1, \theta_2, \cdots \theta_p\right)^t$ is a point in the *p* dimensional search domain.

Reproduction: The least healthy bacteria eventually die while each of the healthier bacteria (those yielding lower value of the objective function) asexually split into two bacteria, which are then placed in the same location. This keeps the swarm size constant.

Elimination and Dispersal: Gradual or sudden changes in the local environment where a bacterium population lives may occur due to various reasons e.g. a significant local rise of temperature may kill a group of bacteria that are currently in a region with a high concentration of nutrient gradients. Events can take place in such a fashion that all the bacteria in a region are killed or a group is dispersed into a new location. Some guidelines for BFA Parameter Choices are

Size of population 'S': Increasing *S* can significantly increase the computational complexity of the algorithm. However, for
larger values of *S*, it is more likely at least some bacteria near an optimum point should be started, and over time, it is then more likely that many bacterium will be in that region, due to either chemotaxis or reproduction.

Length of chemotactic step 'C(i)'**:** If C(i) are too large, then if the optimum value lies in a valley with steep edges, the search will tend to jump out of the valley, or it may simply miss possible local minima by swimming through them without stopping. On the other hand, if C(i) are too small, convergence can be slow, but if the search finds a local minimum it will typically not deviate too far from it. *c*(*i*) is a sort of a "step size" for the algorithm

Chemotactic step '*Nc***':** If the size of *Nc* is chosen to be too short, the algorithm will generally rely more on luck and reproduction, and in some cases, it could more easily get trapped in a local minimum (premature convergence). *Ns* creates a bias in the random walk (which would not occur if Ns = 0), with large values tending to bias the walk more in the direction of climbing down the hill.

Reproduction number '*Nre***':** If *Nre* is too small, the algorithm may converge prematurely; however, larger values of *Nre* clearly increase computational complexity.

Elimination and dispersal number '*Ned***':** A low value for *Ned* dictates that the algorithm will not rely on random elimination-dispersal events to try to find favorable regions. A high value increases computational complexity but allows the bacteria to look in more regions to find good nutrient concentrations. Clearly, if *ped* is large, the algorithm can degrade to random exhaustive search. If, however, it is chosen appropriately, it can help the algorithm jump out of local optima and into a global optimum.

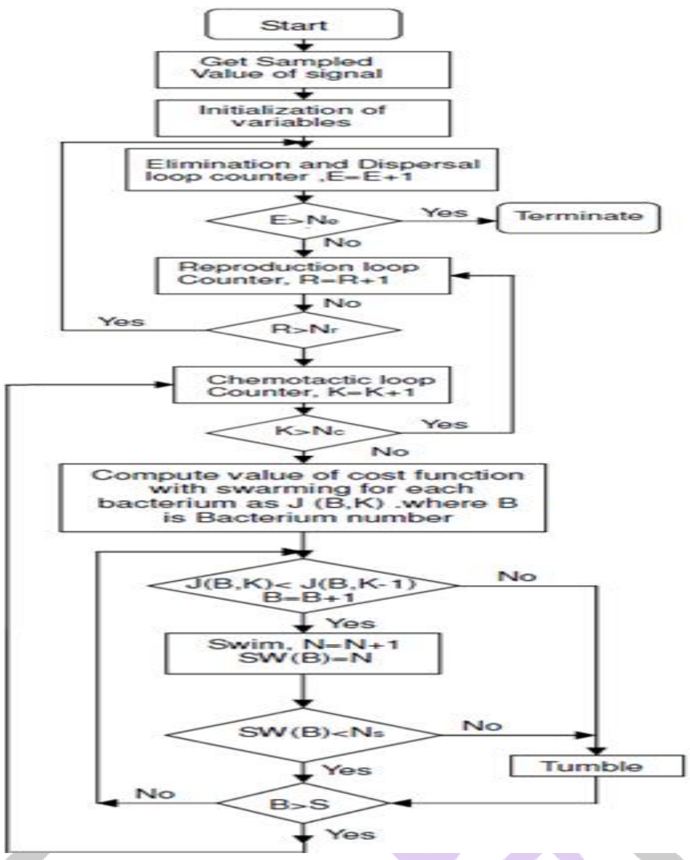

Start
Get Sampled Value of signal
Initialization of variables
Elimination and Dispersal loop counter ,E=E+1
E>Ne
Yes
Terminate
No
Reproduction loop Counter, R=R+1
No
Yes
R>Nr
Chemotactic loop Counter, K=K+1
K>Nc
Yes
No
Compute value of cost function with swarming for each bacterium as J (B,K) ,where B is Bacterium number
J(B,K)< J(B,K-1) B=B+1
No
Yes
Swim, N=N+1 SW(B)=N
SW(B)<Ns
No
Tumble
Yes
No
B>S
Yes

Fig. 3 Flowchart of BFA

Parameters defining cell-to-cell attractant functions 'J_{cc}':
If the attractant width is high and very deep, the cells will have a strong tendency to swarm (they may even avoid going after nutrients and favor swarming). On the other hand, if the attractant width is small and the depth shallow, there will be little tendency to swarm and each cell will search on its own Social versus independent foraging is then dictated by the balance between the strengths of the cell-to-cell attractant signals and nutrient concentrations.

6.1 PROPOSED ALGORITHM FOR THE IMPLEMENTATION OF THE BFOA

We have used bacterial forging optimization algorithm for segmention of the image. This algorithm shows how the bacterial based theory is used to detect the image. Natural selection tends to eliminate animals with poor "foraging strategies" (methods for locating, handling, and ingesting food) and favor the propagation of genes of those animals that have successful foraging strategies since they are more likely to enjoy reproductive success (they obtain enough food to enable them to reproduce). After many generations, poor foraging strategies are either eliminated or shaped into good ones (redesigned). Logically, such evolutionary principles have led scientists in the field of "foraging theory" to hypothesize that it is appropriate to model the activity of foraging as an optimization process:

Steps:

initialize Parameters: p, S, Nc, Ns, Nre and C (i), i = 1, 2… S
Where,
p = Dimension of search space
S = Number of bacteria in the population
Nc = Number of chemotaxis steps
Ns= Number of swimming steps
Nre = Number of reproduction steps
C (i) = Step size taken in the random direction specified by the tumble
J (i, j, k) = Fitness value or cost of i-th bacteria in the j-th chemotaxis and k-th reproduction steps
θ (i, j, k)= Position vector of i-th bacterium in j-th chemotactic step and k-th reproduction steps
Jbest (j, k) = Fitness of best position in the j-th chemotaxis and k-th reproduction steps
Jglobal= Fitness value or cost of the global best position in the entire search space

Step 1: Update the following parameters: J (i, j, k), Jbest (j, k) and Jglobal= Jbest (j, k)

Step 2: Reproduction Loop: k = k+1

Step 3: Chemotaxis loop: j = j+1

a) Compute fitness function J (i, j, k) for i= 1, 2, 3…S
b) Update Jbest (j, k).
c) Tumble: Generate a random vector Δ(i) ∈Rp with each
element Δm (i) m = 1, 2…p, a random
d) Compute q for i = 1, 2 ……….S
e) Swim
i) Let m = 0 (counter for swim length)
ii) While m < Ns (if have not climbed down too long.

- Let m = m+1
- Compute fitness function J (i, j+1, k) for i = 1, 2… S
- Update Jbest (j+1, k)
- If Jbest (j+1,k)< Jbest (j, k) (if doing better), Jbest (j, k) = Jbest (j+1, k) Compute θ for i = 1, 2…S [Synchronous position updation] Use this θ (i, j +1, k) to compute the new J (i, j+1, k)
- Else, let m = Ns. This is the end of the while statement Sub Step f) Mutation Operator Compute θ for i = 1, 2….S [Synchronous position updation by mutation operator]

Step 4: If j < Nc, go to step 3. In this case, continue chemotaxis since the life of bacteria is not over.

Step 5: Reproductions: The Sr=S/2 bacteria with the highest cost function values die and other Sr bacteria with the best values split. Update Jglobal from Jbest (j, k).

Step 6: If k < Nre, go to Step 2 otherwise end.

and rise time.

VII RESULT AND DISCUSSION

Table7.1 shows the comparison of PSO and BFO for 50 iterations with alpha = 10 and beta = 5.

S.No	Generation	Alpha	Beta	Kp	KI	Kd	Mp%	Ts	Tr
1	50	10	5	0.9682	0	0.854	22.5329	9.2757	1.4842
2	50	10	5	0.8809	0	0.6724	8.1781	8.3174	1.6991

Table 7.1 For PSO and BFO when n = 50 alpha = 10 and Beta = 5

The response in Fig.(a) shows that the comparison of PSO and BFO when the number of iteration is 50 and alpha =10 and beta=5 in such a condition that the output response of the system gives the value of maximum overshoot, rise time, settling time, and also the value of the gain i.e. the proportional gain and the integral gain.

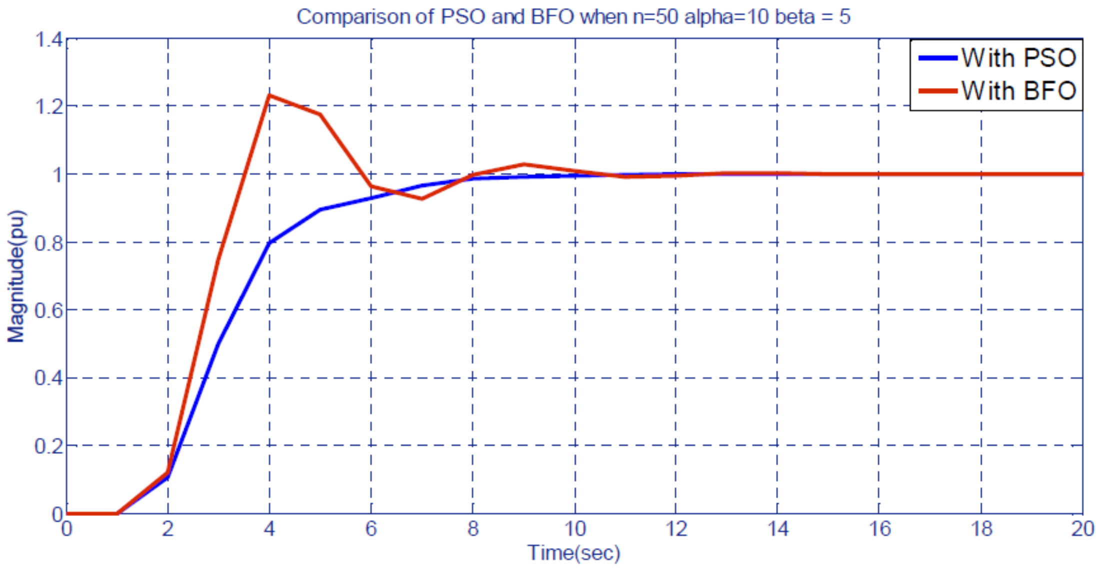


Fig.(a) Output response for comparison of BFO and PSO when n=50

When n=50 (i.e. number of iteration denoted by generation) in this case considering the value of Beta to be 5 and alpha to be 10, which will give the value of Kp = 0.9682 and Kd = 0.854 for PSO algorithm and similarly for BFO the value of Kp and Kd is 0.8809 and 0.6724 respectively. From the figure it is also clear that the maximum overshoot and settling time of PSO is greater than that of BFO, and the rise time is smaller than that of BFO.

The Table 7.2 shows the comparison of PSO and BFO for 50 iterations with alpha = 10 and beta = 5

S.No	Generation	Alpha	Beta	Kp	KI	Kd	Mp%	Ts	Tr
1	50	10	10	0.7463	0	0.8605	0.6630	7.0386	2.0542
2	50	10	10	0.8403	0	0.7790	2.3438	8.1187	1.8221

Table 7.2 For PSO and BFO when n = 50 alpha = 10 and Beta = 10

The response in Fig (b) shows that the comparison of PSO and BFO when the number of iteration is 50 and alpha =10 and beta =10 in such a condition the output response of the system gives the value of maximum overshoot, rise time, settling time, and also the value of the gain i.e. the proportional gain and the integral gain.

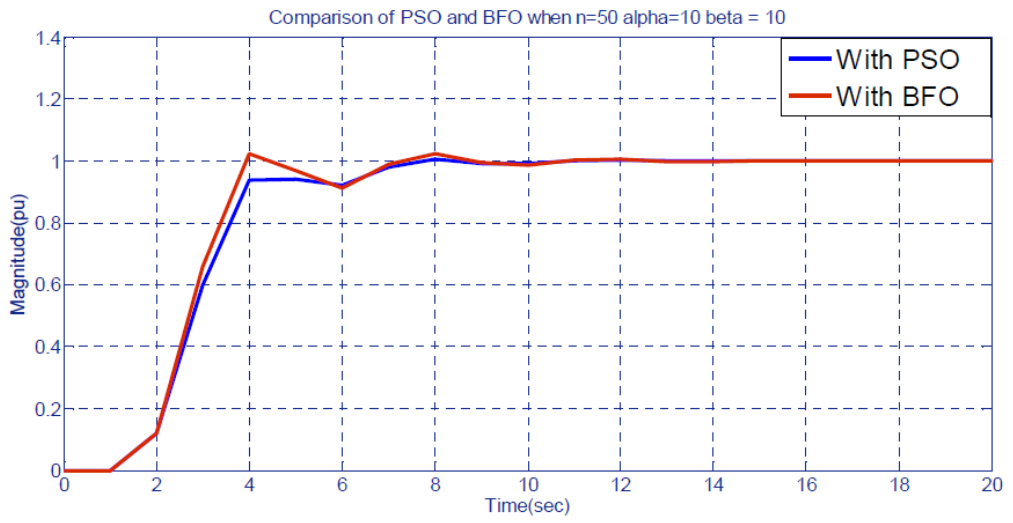


Fig.(b) Output response for comparison of BFO and PSO when n=50

When n=50 (i.e. number of iteration denoted by generation) in this case considering the value of Alpha to be 10 and Beta to be 10, which will give the value of Kp = 0.7463 and Kd = 0.8605 for PSO algorithm and similarly for BFO the value of Kp and Kd is 0.8403 and 0.7790 respectively. From the figure it is also clear that the maximum overshoot and settling time of BFO is greater than that of PSO, and the rise time is smaller than that of PSO.

References

[1] Zafer Bingul, Oguzhan Karahan; ‘ Tuning of Fractional PID Controllers Using PSO Algorithm for Robot Trajectory Control’;International Conference on Mechatronics, 2011, pp.955-960.
[2] V. K. Kadiyala; R. K. Jatoth; S. Pathalaiah; ‘ Design and implementation of fractional order PID controller for Aero fin control system’; IEEE; NaBIC 2009; pp. 696-701.
[3] C. H. Liu; Y. Y. Hsu; ‘Design of a Self-Tuning PI Controller for a STATCOM Using Particle Swarm Optimization’; IEEE 2010; pp. 702-715.
[4] D. Maiti, A. Acharya, M. Chakraborty, A. Konar, R. Janarthanan; ‘Tuning PID and PIlDd Controllers using the Integral Time Absolute Error Criterion’; IEEE 2008; pp. 457- 462.
[5] C. M. Lin, M. H. Lin, C. H. Chen, D. S. Yeung; ‘Robust PID control system design for chaotic systems using particle swarm optimization algorithm’; IEEE 2009; pp. 3294-3299.
[6]W.W. Cai, L. X. Jia, Y. B. Zhang, N. Ni; ‘Design and simulation of intelligent PID controller based on particle swarm optimization’; IEEE 2010
[7] Y. Bo, L. W. Zhou, Y. Feng; ‘A New PSO-PID Tuning Method for Time-delay Processes’; IEEE 2008
[8] A. A. A. El-Gammal, A. A. El- Samahy; ‘A Modified Design of PID Controller For DC Motor Drives Using Particle Swarm Optimization PSO’; IEEE 2009; pp. 419-424.
[9] E. Salim Ali, S. M. Abd-Elazim; ‘ Optimal PID Tuning for Load Frequency Control Using Bacteria Foraging Optimization Algorithm’, 14th International Middle East Power System Conference (MEPCON’10), December 2010, pp. 410-415.
[10] B. Sumanbabu, S. Mishra, B.K. Panigrahi, and G.K. Venayagamoorthy, “Robust tuning of modern power system stabilizers using bacterial foraging algorithm,” IEEE Congress on Evolutionary Computation, CEC 2007.
[11 T. Jain, M. J. Nigam; ‘Optimization of PD-PI Controller Using Swarm Intelligence’ ,International Journal of Computational Cognition, December 2008, Vol. 6, No. 4, pp. 55-59.
[12] S. Dasgupta, S. Das, A. Abraham, and A. Biswas, “Adaptive computational chemotaxis in bacterial foraging optimization: an

analysis," IEEE Transactions on Evolutionary Computing, (in press), 2008.
[13] R. Allamneni , "Bacterial foraging based channel equalizers," Master of Technology, Dept. Electronics and Communication Eng., National Institute of Technology, Rourkela, Orissa, Spain, May-2006.
[14] Mangraj B B, Misra IS and Barisal AK, "Optimizing Included Angle of symmetrical V-Dipoles for Higher Directivity Using Bacteria Foraging Optimization Algorithm", PIER B, Vol. 3, 2008, pp. , 295-314

[15] Sastri GSVR, Pattnaik SS, Bajpai OP, Devi S, Sagar CV, Patra PK and Bakwad KM, "Bacterial Foraging Optimization Technique to Calculate Resonant Frequency of Rectangular Microstrip Antenna", Int. J.

RF Microwave Computer Aided Eng., Vol. 18, 2008, pp. , 383-388.

[16] Sastri GSVR, Pattnaik SS, Bajpai OP, Devi S and Bakwad KM , "Velocity Modulated Bacterial Foraging Optimization Technique (VMBFO)", Applied Soft Computing Journal, Vol. 11 (1), January 2011, pp. 154-165.

[17] Sastri GSVR, Pattnaik SS, Bajpai OP, Devi S, Bakwad KM and Patra PK "Intelligent Bacterial Foraging Optimization Technique to Calculate Resonant Frequency of RMA", IJMOT, Vol 4(2), March 2009, pp.67-75.

[18] Chen H, Zu Y, and Hu K, "Adaptive Bacterial Foraging Optimization", Abstracts and Applied Analysis, Vol. 2011, 2011, pp. 1-27.

[19] Mahmoud KR, "Design Optimization of A Bow-Tie Antenna For 2.45 GHz RFID Readers Using A Hybrid BSO-NM Algorithm", PIER 100, 2010, pp. 105-117

[20] Bakwad KM, Pattnaik SS, Sohi BS, Devi S, Panigrahi BK, Sastri GSVR, "Bacterial Foraging Optimization Technique Cascaded with Adaptive Filter to Enhance Peak Signal to Noise Ratio from Single Image", IETE Journal of Research, Vol 55 (4), Jul-Aug 2009.

[21] Passino KM, "Biomimicry of Bacterial Foraging", IEEE Control System Magzine, Vol. 22, 2002, pp. 52-67.

[22] Datta T, Misra IS, Mangraj BB, Imtiaj S, "Improved Adaptive Bacteria Foraging algorithm in Optimization of Antenna Array for Faster Convergence", PIER C, Vol. 1, 2008, pp. 143-157.

[23] Liu W, Chen H, Chen H, and Chen M, "RFID Network Scheduling Using an Adaptive Bacteria Foraging Algorithm", Journal of Computer Information Systems (JCIS), Vol. 7(4), 2011, pp. 1238-1245.

[24] Korani W, " Bacterial Foraging Oriented By Particle Swarm Optimization Strategy for PID Tuning", GECOO 2008, 12-16 July, 2008, Atlanta, USA, pp. 1823-1826.

[25] Zhang Y, Wu L and Wang S, " Bacterial Foraging Optimization Based Neural Network for Short term Load Forecasting.

[26] Singh, Mahesh, R. N. Patel, and Rajkumar Jhapte. "Performance comparison of optimized controller tuning techniques for voltage stability."2016 IEEE First International Conference on Control, Measurement and Instrumentation (CMI). IEEE, 2016.