REACTIVE POWER LOSS OPTIMIZATION FOR AN IEEE 14-BUS POWER SYSTEM USING VARIOUS ALGORITHMS

Deepa Subramanıam NACHIMUTHU[1], Rızwana John BASHA[2]

[1,2]Department of Electrical and Electronics Engineering,
Anna University Regional Centre, Coimbatore, Tamilnadu, India
E-Mail Id: [1]deepapsg@gmail.com, [2]rizwana.scholar@outlook.com

Abstract: *Optimization techniques play vital role in many problems. Many optimization algorithms have been proposed in past decades. In this paper various optimization techniques like Particle Swarm Optimization, Modified Particle Swarm Optimization, Artificial Bee Colony Algorithm, Fish School Search Algorithm and Gravitational Search Algorithm are compared and tested in an IEEE-14 Bus power system. By using these five optimization techniques, the power losses like active power and reactive power are reduced drastically. By reducing the losses, the voltage profile improvement and voltage stability enhancement can be maintained. Also this paper shows the better optimization technique which suits the power system to maintain stability by reducing power losses. The results are tabulated to show how much losses have been reduced from the determined actual losses in the power system.*

Keywords: *Particle Swarm Optimization (PSO), Modified Particle Swarm Optimization (MPSO), Artificial Bee Colony (ABC) Algorithm, Fish School Search (FSS) Algorithm and Gravitational Search Algorithm (GSA)*

1. Introduction

The required reactive power optimization at various locations determines the objective function. The problem on reactive power influences on operation of power systems significantly. Due to this power generated in power system, it results in transmission losses. This is one of the most important problems which are present in this field [1-2]. The problem that has to be solved in a reactive power optimization is to determine the required reactive generation at various locations so as to determine the required reactive generation at various locations so as to optimize the objective function. Decrease in power factor can also be due to the loss of reactive power [3].

PSO is a stochastic, population based technique and it is well adapted to multidimensional space [4-5]. MPSO has a faster convergence compared to PSO due to its acceleration weight and inertia factor introduced in MPSO [6]. In ABC, the best solution is observed by a scout due to randomly produced solution. When the target nectar is found the bees will interact and the intensity of the bee interactions gives the solution of the problem [7-8].

Fish School Search was inspired by some fishes in gregarious behavior. FSS is to improve the survivability of the entire group. The success of FSS search space is mostly due to the members of the population [9]. In GSA, the particles move towards the best particles. Due to the heavy mass inertia, the particles search the space locally. Higher attraction of agents is due to the more gravitational mass. The performance of GSA is better when compared to all other algorithms [10].

In this paper, GSA algorithm shows better performance compared to other algorithms. Since GSA is inspired by physical phenomenon. Due to this algorithm the losses are reduced drastically compared to other algorithms. Hence by GSA the voltage stability enhancement is improved due to the reduced losses. GSA algorithm is tested in IEEE 14-Bus power system. The results are tabu-

Received on: 22.12.2012
Accepted on: 03.09.2013

lated and shown clearly to prove the advantages of this algorithm.

2. Proposed Methodology

2.1 PSO Algorithm

2.1.1 Overview

Dr. Russell Eberhart and Dr. James Kennedy developed particle swarm optimization algorithm in 1995. This algorithm is inspired by behaviour of bird flocking or fish schooling. This technique uses the number of particles (agents) to search the solution present in the population.

The basic definitions used in PSO algorithm are:

- *Particle:* The elements which are interacting in problem space.
- *Swarm:* Movement of group of 'n' number of particles in problem space.
- *Optimization:* The opportunity of availing best ones perfectly.
- *PSO:* In total, it is the opportunity of finding the optimal solutions.
- *Pbest:* The best value achieved so far by the individual particle.
- *Lbest:* The best value achieved so far by a particle in competitions of its neighbourhood particles.
- *Gbest:* The best fitness value which is achieved from the individuals.

2.1.2 Algorithm

Step 1: Generate 'n' number of population randomly.
Step 1.1: Initialize particles position and velocity randomly.
Step 2: For each particles position, evaluate fitness.
Step 2.1: Initialize Pbest (Particle best), Lbest (Local best) and Gbest (Global best) for each particle and iteration count.
Step 3: Update iteration count.
Step 4: Update particles position, Velocity, Pbest, Lbest and Gbest.
Step 4.1: Update velocity of the component using;

$$v(k,j,i+1) = v(k,j,i) + C1 * rand * \left(pbestx(j,k) - x(k,j,i)\right) + C2 * rand * \left(gbestx(k) - x(k,j,i)\right) \quad (1)$$

where,
v = velocity of the particle at the i^{th} iteration.
rand = random number should be between 0 and 1.
C1,C2 is taken as 1.

Step 4.2: Update the Position Component by using;

$$x(k,j,i+1) = x(k,j-1,i) + v(k,j,i) \quad (2)$$

Step 5: Check for stopping condition;
Step 5.1: If 100 iterations are reached, stop it.
Step 5.2: Else update iteration count.

2.2 MPSO Algorithm:

2.2.1 Overview

MPSO is a technique which is used for fast convergence compared to PSO algorithm by updating inertia weight and acceleration factor. The ability of breaking away the local optimum is greatly improved. It also avoids the premature convergence problem effectively.

2.2.2 Algorithm

Step 1: Generate 'n' number of population randomly.
Step 1.1: Initialize particles position and velocity randomly.
Step 2: For each particles position, evaluate fitness.
Step 2.1: Initialize Pbest (Particle best), Lbest (Local best) and Gbest (Global best) for each particle and iteration count.
Step 3: Update iteration count.
Step 4: The population can be updated by using MPSO technique:

i) *Inertia Weight Factor:*

The inertia weight factor is calculated by using:

$$W^k{}_i = W_{min} + \frac{F^{k-1}{}_{pbest} \times \left|F^{k-1}{}_1 - F^{k-1}{}_{pbest}\right|}{F^{k-1}{}_1 \times \left|F^{k-1}{}_1 - F^{k-1}{}_{gbest}\right|} \quad (3)$$

ii) *Modified Acceleration Factor*
The acceleration factor can be calculated to speed up the convergence.

$$C^k{}_{1,i} = \sqrt{\frac{F^{k-1}{}_i}{F^{k-1}{}_{pbest}}} \quad (4)$$

$$C^k{}_{2,i} = \sqrt{\frac{F^{k-1}{}_i}{F^{k-1}{}_{gbest}}} \quad (5)$$

Step 5: Update particles position, Velocity, Pbest, Lbest and Gbest.
Step 6: Check for stopping condition;
Step 6.1: If 100 iterations are reached, stop it.
Step 6.2: Else update iteration count.

2.3 ABC Algorithm:

2.3.1 Overview

ABC is a stochastic, population based method developed by Dervis Karaboga in 2005. This algorithm is motivated by the intelligent behaviour of honey bees. Here the bee colony comprises of three groups of bees: employed bees, onlookers and scouts. Each employed bee is assumed for each source of food. After coming back to hive, the bees dance on their area. Onlookers choose the food sources depending on the dances of the employed bees. If the food sources of the employed bees are abandoned, it becomes stouts.

2.3.2 Algorithm

Step 1: Generate 'n' number of population randomly.
Step 2: Each solution vector is generated by using;

$$X_{ij} = X_{minj} + rand(0,1)\times\left(X_{maxj} - X_{minj}\right) \quad (6)$$

where,
X_{maxj} and X_{minj} are the upper and lower bounds of j dimension.

Step 3: Fitness of each food source is evaluated.
Step 4: To determine new food source, each employed bee searches its current food in the neighbourhood;

$$V_{ij} = X_{ij} + \varphi_{ij}\left(X_{ij} - X_{kj}\right) \quad (7)$$

where,
k,j : randomly chosen number, taken as 1.
Φ_{ij} : random number chosen between -1 and +1.

Step 5: If the fitness of the new food source is equal to or better than that of X_i; the new food source takes the place of X_i in the population and becomes a new member. Else it (employed bee) leaves the position and moves to a new food source.

Step 6: After evaluating the information received from the employed bees, the onlooker bee selects a food source. To select the food source (i), the probability (P_i) is determined by;

$$P_i = \frac{f_i}{\sum_{i=1}^{n} f_i} \quad (8)$$

where,
f_i : Fitness value of the food source X_i.
Step 7: The onlooker bee generates a new food source and fitness evaluation is determined.
Step 8: If the food source cannot be further improved by upcoming iterations, it will be abandoned and the employed bee of the concerned food source becomes a stout. Te stout generates a new food source by using;

$$V_{ij} = X_{minj} + rand(0,1)\times\left(X_{maxj} - X_{minj}\right) \quad (9)$$

Step 9: Checking for stopping criteria.
Step 9.1: If termination condition is met, best food source is reported.
Step 9.2: Else the iteration is repeated from step 4.

2.4 FSS Algorithm

2.4.1 Overview

FSS is a computational intelligent technique developed by Bastos-Filho and Lima-Neto in 2007. It is a stochastic, bio-inspired, population based technique. This search process is carried out by fish having limited memory individuals. The FSS algorithm can be grouped into two classes: feeding and swimming. The operator eating determines the quality of the solution. There are three swimming operators which drive the fish movements.

2.4.2 Algorithm

Step 1: Generate 'n' number of population (fishes) randomly.
Step 2: Initialize iteration count.
Step 3: Initialize individual movement of each fish.
Step 4: Evaluate the fitness of each fish's new position.
Step 5: Each fish weight increases depending on the feeding process. To calculate new weight;

Feeding Process:

$$W_i(t+1) = W_i(t) + \frac{\Delta f_i}{max\{|f[X_i(t+1)] - f[X_i(t)]|\}'} \quad (10)$$

where,
W_i (t+1) : Weight of the fish.
X_i : Position of ith fish.
f[X_i(t)] : Fitness function.

Step 6 : The position of each fish can be determined by;

Individual Movement:

$$new_i(t) = X_i(t) + rand(-1,1) \times (step_{index}) \quad (11)$$

where,
$step_{index}$: Percentage of search space amplitude bounded by parameters ($step_{index\_min}$ and $step_{index\_max}$).
X_i : Current position of fish.
rand(-1,1) : random number between -1 and +1.

Step 7 : If new position is better than the previous, the movement will occur.

Step 8 : After the movement of fishes, the positions are updated.

Collective-Instinctive Movement:

$$X_i(t+1) = X_i(t) + \frac{\sum_{i=1}^{n} \Delta X_{indexi}\{f[X_i(t+1)] - f[X_i(t)]\}}{\sum_{i=1}^{n}\{f[X_i(t+1)] - f[X_i(t)]\}} \quad (12)$$

where,
ΔX_{indexi} : Position alteration of the ith fish during its individual movement.
Whole school move towards set of the most successful fishes.

Step 9 : The school will be successful, if the collective-volitive movement contracts. If the weight increases then the fish school is successful and new positions are:

$$X_i(t+1) = X_i(t) - step_{vol} \times rand \times [X_i(t) - Bari(t)] \quad (13)$$

Step 10: If the weight decreases, then the fish school is not successful and new positions are:

$$X_i(t+1) = X_i(t) + step_{vol} \times rand \times [X_i(t) - Bari(t)] \quad (14)$$

where,

$$Bari(t) = \frac{\sum_{i=1}^{n} X_i(t) \times W_i(t)}{\sum_{i=1}^{n} W_i(t)} \quad (15)$$

rand : Random number between 0 and 1.

Step 11 : Stopping condition criteria.

Step 11.1 : If definite amount of cycles are reached, stop it.

Step 11.2 : Else update the iteration count.

2.5 GSA Algorithm

2.5.1 Overview

GSA is a meta-heuristic algorithm. It is based on law of gravity and law of motion. It has a good convergence rate compared to PSO and MPSO. The particles in search space are the collection of masses. Here objects are the particles and each of the particles performance is measured by masses. Each object attracts the others by a gravity force and this force causes a movement of all objects towards the heavier mass objects.

2.5.2 Algorithm

Step 1 : Generate 'n' number of particles randomly.

Step 2 : Evaluate the fitness for each particle.

Step 3 : Initialize gravitational constant G(t) and iteration count.

Step 4 : Update:

i. Gravitational Constant:

$$G(t) = G(t_0) \times \left(\frac{t_0}{t}\right)^{\beta}, \beta < 1 \quad (16)$$

where,
G(t_0) : Gravitational Constant at time interval (t_0).

ii. best(t) and worst(t):

$$best\ (t) = \min_{j \epsilon \{1,\ldots N\}} fit_j(t) \quad (17)$$

$$worst\ (t) = \max_{j \epsilon \{1,\ldots N\}} fit_j(t) \quad (18)$$

where,
fit_j(t) : Fitness value of the particle i at time t.

iii. Inertia Mass:

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad (19)$$

where,

$$m_i(t) = \frac{fit_i(t) - worst\ (t)}{best\ (t) - worst\ (t)} \quad (20)$$

Step 5 : Calculate Total Force.

$$F_i^d(t) = \sum_{\substack{j=1 \\ \neq i}}^{N} rand_j F_{ij}^d(t) \quad (21)$$

$$F_{ij}^d(t) = G(t) \times \frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \epsilon} \times \left(x_j^d(t) - x_i^d(t)\right) \quad (22)$$

$$R_{ij}(t) = \left\| X_i(t), X_j(t) \right\|_2 \quad (23)$$

where,
$rand_j$: Random number between 0 and 1
M_{aj} : Active gravitational mass
M_{pi} : Passive gravitational mass
$R_{ij}(t)$: euclidean distance between two particles i and j.

Step 6 : Calculate Acceleration.

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \quad (24)$$

where,
$M_{ii}(t)$: Inertial mass of the i^{th} particle.

Step 7 : Update velocity and position.

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \quad (25)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (26)$$

Step 8 : Check for stopping condition.
Step 8.1 : If 100 iterations are reached, stop it.
Step 8.2 : Else repeat the steps from 3-7.

3. Problem Definition

The IEEE 14-bus power system is used to test the power losses using various optimization techniques. The one line diagram of an IEEE-14 bus power system is shown in the figure 1. The line data, reactive power limit, shunt capacitor data, bus data and transformer tap setting data are given in table 1-5. The data used for this power system are:

- Bus 1 and 2 are considered as generator bus.
- Bus 3-14 is considered as load bus.
- Bus 3, 6 and 8 are considered as synchronous compensators.

By using conventional method like Newton Raphson method, the losses are calculated manually. The various optimization techniques are used to reduce the power losses in the system.

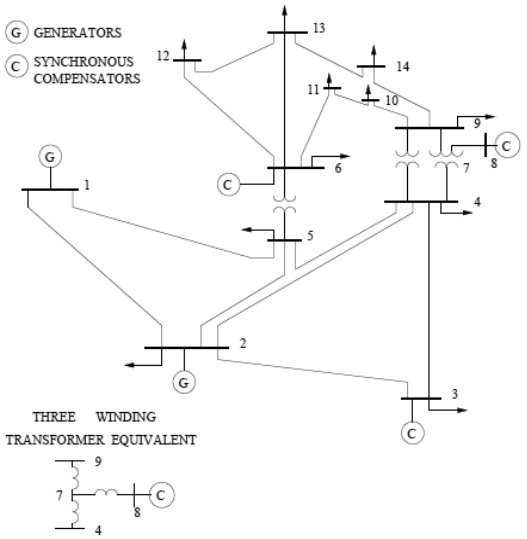

Figure 1. IEEE 14-Bus Power System

The one line diagram of IEEE 14-Bus system is shown in figure 1.

Table 1. Line Data of IEEE 14-Bus Power System

Line No.	From Bus	To Bus	Line Impedance		Half Line charging B (p.u.)
			R (p.u.)	X (p.u.)	
1	1	2	0.0193	0.0591	0.0264
2	2	3	0.0469	0.1979	0.0219
3	2	4	0.0581	0.1763	0.0187
4	1	5	0.0540	0.2230	0.0246
5	2	5	0.0569	0.1738	0.0170
6	3	4	0.0670	0.1710	0.0173
7	4	5	0.0133	0.0421	0.0064
8	5	6	0.0000	0.2520	0
9	4	7	0.0000	0.2091	0
10	7	8	0.0000	0.1761	0
11	4	9	0.0000	0.5561	0
12	7	9	0.0000	0.1100	0
13	9	10	0.0318	0.0845	0
14	6	11	0.0949	0.1989	0
15	6	12	0.1229	0.2558	0
16	6	13	0.0661	0.1302	0
17	9	14	0.1271	0.2703	0
18	10	11	0.0820	0.1920	0
19	12	13	0.2209	0.1998	0
20	13	14	0.1709	0.3480	0

Table 2. Reactive Power Limits

Bus No.	Q_{min} (p.u)	Q_{max} (p.u)
2	-0.40	0.50
3	0	0.40
6	-0.06	0.24
8	-0.06	0.24

Table 3. Shunt Capacitor Data

Bus No.	Susceptance (p.u.)
9	0.19

Table 4. Bus Data of IEEE-14 Bus Power System

Bus No.	Bus Voltage	Generation		Load	
	V (p.u)	P (p.u)	Q (p.u)	P (p.u)	Q (p.u.)
1	1.060	2.32	0.16	0.000	0.000
2	1.045	0.40	0.00	0.217	0.127
3	1.010	0.00	0.00	0.942	0.191
4	1.000	0.00	0.00	0.478	0.039
5	1.000	0.00	0.00	0.076	0.016
6	1.070	0.00	0.00	0.112	0.075
7	1.000	0.00	0.00	0.000	0.000
8	1.090	0.00	0.00	0.000	0.000
9	1.000	0.00	0.00	0.295	0.166
10	1.000	0.00	0.00	0.090	0.058
11	1.000	0.00	0.00	0.035	0.018
12	1.000	0.00	0.00	0.061	0.016
13	1.000	0.00	0.00	0.035	0.058
14	1.000	0.00	0.00	0.149	0.050

Table 5. Transformer Tap Setting Data of an IEEE 14-Bus Power System

From Bus	To Bus	Tap Setting Value (p.u.)
4	7	0.978
4	9	0.969
5	6	0.932

4. Results and Discussions

The results are shown in table 6-7. The GSA algorithm is compared with the various algorithms. The reactive power is minimized much by using GSA algorithm. The active and reactive power losses are minimized much which is tabulated in table 7. By reducing the losses, the voltage profile can be improved. This makes the stable power system. The voltage profile improvement is shown in figure 2. Here the control variables are taken as generator voltages and the rest are taken as dependent variables.

Table 6. System Status after Optimization

Variables	PSO Voltage (p.u.)	MPSO Voltage (p.u.)	ABC Voltage (p.u.)	FSS Voltage (p.u.)	GSA Voltage (p.u.)
Control Variables					
V1	1.0600	1.0600	1.0600	1.0600	1.0600
V2	1.0450	1.0450	1.0450	1.0450	1.0450
V3	1.0100	1.0100	1.0100	1.0100	1.0100
V6	1.0700	1.0700	1.0700	1.0700	1.0700
V8	1.0900	1.0900	1.0900	1.0900	1.0900
Dependent Variables					
V4	0.9796	0.9561	0.9964	0.9777	0.9265
V5	0.9653	1.0041	0.9714	0.9324	0.9801
V7	0.9692	0.9821	0.9905	0.9507	0.9749
V9	0.9504	0.9121	0.9414	0.9767	1.0073
V10	0.9534	0.9410	0.9254	0.9697	1.0017
V11	0.9526	0.9541	1.0033	0.9301	1.0046
V12	0.9819	0.9405	0.9175	0.9389	0.9735
V13	0.9194	0.9338	0.9121	0.9373	0.9646
V14	0.9606	0.9458	0.9853	0.9952	1.0061

Table 7. Result Comparisons for an IEEE 14-Bus Power System

Algorithm	14-Bus Power System
Conventional Method	13.593 MW + j56.910 MVAR
PSO Algorithm	9.9159 MW + j37.1336 MVAR
MPSO Algorithm	8.5053 MW + j39.8343 MVAR
ABC Algorithm	6.4611 MW + j35.6122 MVAR
FSS Algorithm	7.8458 MW + j48.1797 MVAR
GSA Algorithm	3.2764 MW + j31.8845 MVAR

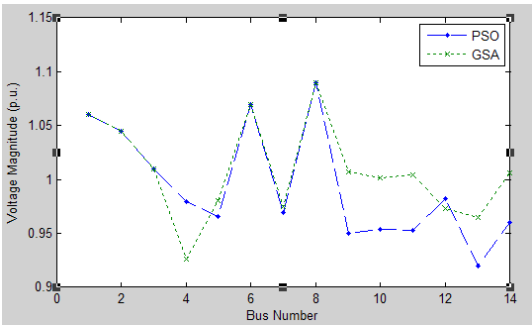

Figure 2. Voltage Profile Improvement of IEEE 14-Bus Power System

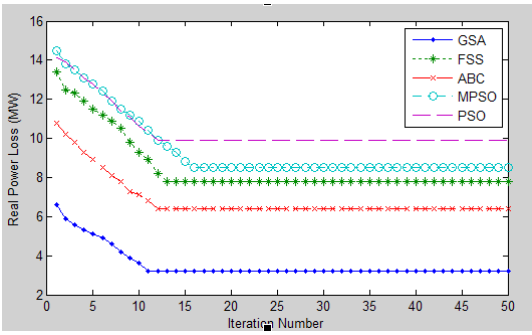

Figure 3. Real Power Loss of an IEEE 14-Bus Power System

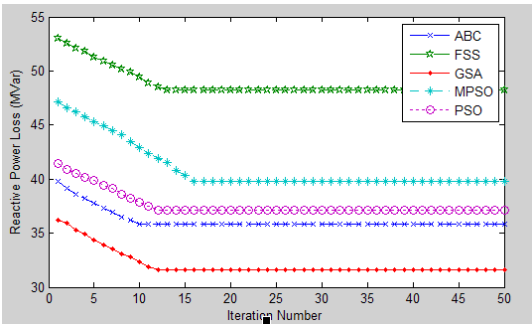

Figure 4. Reactive Power Loss of an IEEE 14-Bus Power System

Figure 3 and figure 4 depicts the real and reactive power loss of the tested system of table 7. Here the dependent variables depend on control variables. The voltage values are improved by using optimization techniques. By improving this voltage profile, the losses like real and reactive power losses are minimized. As losses are minimized, the power system will remain stable. Hence from all the simulation results shown, GSA optimization technique proves well with reduced losses.

5. Conclusion

In this paper, PSO, MPSO, ABC, FSS and GSA algorithm are proposed. The IEEE 14-Bus power system is tested using these algorithms. The GSA algorithm minimizes the reactive power drastically compared to the other algorithms. It has a good convergence compared to other algorithms. Hence by minimizing the losses like active power and reactive power, the voltage profile is improved and makes the power system much stable. Simulation results prove the effectiveness of GSA algorithm.

6. References

[1] S.N. Deepa, J. Rizwana, "Power System Stability by Reducing Power Losses using Optimization Techniques", Proceedings of IEEE on Computational Intelligence and Computing Research Conference, Dec. 2013, pp. 178-181.

[2] P.R. Sujin, T. Ruban Deva Prakash and M. Mary Linda, "Particle Swarm Optimization based Reactive Power Optimization", Journal of Computing, Vol.2, Issue.1, pp. 73-78, Jan. 2010.

[3] O. Carl John Salaan and R. Noel Estoperez, "An Artificial Neural Network based Real-Time Reactive Power Controller", Proceedings of the World Congress on Engineering and Computer Science, Oct. 2011.

[4] Vivek Kumar Jain, Himmet Singh and Laxmi Srivastava, "Minimization of Reactive Power using Particle Swarm Optimization", Int. Journal of Computational Engineering Research, Vol.2, Issue 3, pp.686-691, June 2012.

[5] Joong-Rin Shin, Byung Seop Kim, Jong Bae Park and Kwang Y. Lee, "A New Optimal Routing Algorithm for Loss Minimization and Voltage Stability Improvement in Radial Power Systems", IEEE Transactions on Power Systems, Vol.22, Issue 2, pp.648-657, May 2007.

[6] V.P. Sakthivel and S. Subramanian, "Using MPSO Algorithm to Optimize Three-Phase Squirrel Cage Induction Motor Design", Proc. of IEEE Conference, pp. 261-267, 2011.

[7] Kursat Ayan, Ulas Kilic, "Artificial Bee Colony Algorithm Solution for Optimal Reactive Power Flow", Applied Soft Computing, Vol.12, Issue 5, pp.1477-1482, May 2012.

[8] Ali Ozturk, Serkan Cobanli, Pakize Erdogmus and Salik Tosun, "Reactive Power Optimization with Artificial Bee Colony Algorithm", Academic Journals, vol.5, No.19, pp.2848-2857, Oct.2010.

[9] J.C.C Anthony Lins, J.A. Carmelo Bastos-Filho, N.O. Debora Nascimento, A.C. Marcos Oliveira Junior and Fernando B De Lima-Neto, "Analysis of the Performance of the Fish School Search Algorithm Running in Graphic Processing Units", Theory and New Application of Swarm Intelligence, pp. 17-32, Mar. 2012.

[10] Esmat Rashedi, Hossein Nezamabadi-pour and Saeid Sryazdi, "GSA: A Gravitational Search Algorithm", Information Sciences, Elseveir, pp. 2232-2248, Mar. 2009.

[11] K. Lenin, M.R. Mohan, "Ant Colony Search Algorithm for Optimal Reactive Power Optimization", Serbian Journal of Electrical Engg, Vol.3, No.1, pp.77-88, June 2006.

[12] Hirotaka Yoshida, Kenichi Kaowatza, Yoshikazu Fukuyama, Yosuke Nakanishi, "Particle Swarm Optimization for Reactive Power and Voltage Control considering Voltage Stability", Proc. Of Int. Conf. on Intelligent System Applications to Power Systems, Apr.1999, pp.1-6.

[13] A.R. Bhowmik, A.K. Chakraborty, and P. Das, "Optimal Location of UPFC based on PSO Algorithm considering Active Power Loss Minimization", Proc. of IEEE Conference, 2012, pp.1-5.

[14] A.H. Manatawy and M.S. Al-Ghandi, "A New Reactive Power Optimization Algorithm", Proceedings of IEEE Bolonga Power Tech Conference, June 2003.

[15] P.K. Roy, S.P. Ghoshal, and S.S Thakur, "Optimal VAR Control for Improvements in Voltage Profiles and for Real Power Loss Minimization using Biogeography based Optimiation", Electrical Power and Energy Systems, Vol.43, Issue 1, pp.830-838, Dec. 2012.

[16] Abbas Rabiee, Maziar Vanouni, Mostafa Parniani, "Optimal Reactive Power Dispatch for Improving Voltage Stability Margin using a Local Voltage Stability Index", Energy Conversion and Management, Vol.59, pp.66-73, July 2012.

[17] M. Vardarajan and K.S. Swarup, "Differential Evolutionary Algorithm for Optimal Reactive Power Dispatch", Electrical Power and Energy Systems, Vol.30, Issue 8, pp.435-441, Oct. 2008.

[18] Davoud Sedighizadeh and Ellips Masehian, "Particle Swarm Optimization: Methods, Taxonomy and Applications", Int. Journal of Computer Theory and Engg, Vol.1, No.5, pp. 486 – 502, Dec.2009.