Comparative Study of Population Based Techniques for Power System Stabilizer Design

Pinaki Mitra, *Student Member, IEEE*, Chuan Yan, *Student Member, IEEE*, Lisa Grant, *Student Member, IEEE*,
Ganesh K. Venayagamoorthy, *Senior Member, IEEE* and Komla Folly, *Member*, IEEE
Real-Time Power and Intelligent Systems Laboratory
Missouri University of Science and Technology
Rolla, MO, USA
gkumar@ieee.org

Abstract— **Power System Stabilizers (PSSs) are used in interconnected power systems in order to mitigate low frequency oscillations. The comparison of the performances of four advanced population based techniques in tuning the parameters of PSS in a three-machine-nine-bus test system is presented in this paper. The algorithms considered in this paper are: a) Differential Evolution based Particle Swarm Optimization (DEPSO), b) Modified Clonal Selection Algorithm (MCSA), c) Small Population Based Particle Swarm Optimization (SPPSO) and d) Population Based Incremental Learning (PBIL). The comparative study is focused on the frequency domain performances. It is observed that MCSA is performing better than the other three algorithms. The performances of DEPSO and PBIL are quite similar whereas the SPPSO algorithm is not showing very good results compared to the other three.**

Keywords- Differential Evolution based Particle Swarm Optimization; Modified Clonal Selection Algorithm; Population Based Incremental Learning; Small Population Based Particle Swarm Optimization; Small Signal Stability.

I. INTRODUCTION

The generators in a large interconnected power system are generally equipped with Power System Stabilizers (PSSs) that provide supplementary feedback stabilizing signals in the excitation systems [1]. PSSs increase the power system stability limit and thereby extend the power transfer capability by mitigating the low frequency oscillations associated with the electromechanical modes. The PSSs are supposed to compensate for the phase lag between the reference voltage and the generated electrical power. For single machine infinite bus system, this phase lag angle can be calculated by simple mathematics and the PSS parameters can be designed in such a way that it can compensate for that exact phase lag and can damp out the specific mode of oscillation. In multi-machine system, it is hard to find out the exact parameters. It is generally done by frequency domain analysis of the system and PSS parameters are tuned by trial and error method to compensate for the phase lag generated in a particular operating point.

But, the problem with this conventional method is that the parameters found by trial and error method may not be the optimal parameters. Also, the PSS which is performing well for a particular operating condition may not perform that good in other operating condition. That means there is a chance that its performance will degrade with the change in operation condition. For these problems computational intelligence methods are used in tuning the PSS parameters. Generally the tuning is done for different operating conditions for example under different loading conditions. So the parameters found in this way are optimal for a wide range of operating conditions and can give better performance within that range.

Population based methods are very popular as global optimization techniques. PSO, GA, etc. are very commonly used for PSS tuning problems [2]-[3]. In a multi-machine environment, the PSS tuning problem has to deal with a large number of unknown variables. Therefore, it is a very common experience that different combinations of those parameters give rise to very close or even almost equal fitness values. This indicates that the solution hyperplane is non-smooth and has several closely placed local minima. Due to this characteristic of the problem, the conventional algorithms like PSO and GA are trapped to the local minima in most cases. In order to increase the exploration and exploitation capability of these conventional algorithms, several modified versions have been evolved. Small Population based Particle Swarm Optimization (SPPSO) [4] and Differential Evolution based Particle Swarm Optimization (DEPSO) [5] are two such advancements in PSO algorithm which have shown their potential in global optimization problems. Apart from them, Clonal Selection Algorithm (CSA) [6], which is inspired by human immunology and Population Based Incremental Learning (PBIL) [7], which is based on Learning Vector Quantization and Genetic Algorithm (GA), have also shown their promise among the other population based optimization methods. In this paper, a comparative study of the four above mentioned algorithms are carried out towards solving the PSS tuning problem in a multi-machine environment. The rest of the paper is organized as follows: Section II briefly describes the working principles of the four algorithms. Section III presents the test system considered in the paper. Section IV describes the procedure how the optimal tuning of PSS is achieved. Section V presents the results and the discussions. Finally, the conclusion is drawn in Section VI.

II. Population Based Algorithms

A. DEPSO

Conventional PSO is a swarm intelligence based algorithm developed by James Kennedy and Russell Eberhart in 1995 [8]-[10]. This algorithm is based on the behavior of a school of fish or flock of birds. In conventional PSO, better solutions are evolved through the collective interactions of the individuals in the swarm. The particles use memory to keep track of the position that resulted in the highest fitness, known as the 'p_{best}' value. The best value of all the 'p_{best}' values is defined as the global best position, 'g_{best}' with respect to the desired output. The core of the PSO algorithm is the velocity and position update equations given by (1) and (2) respectively. The velocity of the ith particle of d dimension is given by:

$$v_{id}(k+1) = w \cdot v_{id}(k) + c_1 \cdot rand_1 \cdot (p_{best\_id}(k) - x_{id}(k)) \\ + c_2 \cdot rand_2 \cdot (g_{best\_id}(k) - x_{id}(k)) \tag{1}$$

The position vector of the ith particle of d dimension is updated as follows:

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \tag{2}$$

Differential Evolution (DE) is a population-based search strategy developed by Price and Storn in 1995 [11]. DE is similar to other evolution based algorithms since it consists of an initial randomly-generated population, which is improved using selection, reproduction, and mutation processes repeated through generations until problem convergence occurs. In DE the mutation operator consists of an arithmetic combination of randomly selected individuals in the swarm [12]. Selection of parents does not depend on fitness values so all individuals have an equal chance of being chosen.

DE combined with PSO creates a hybrid DEPSO algorithm [5]. In DEPSO, the regular PSO equations, (1) and (2), are used to update the position of each particle and then a mutation is applied to each particle's 'p_{best}' position to obtain a new set of particles or offspring. The DEPSO equations to produce the offspring are given in (3) – (4), where U is the offspring, P_p is the parent 'p_{best}' particle, P_{2-5} are 4 randomly selected 'p_{best}' particles to create the difference vector, and δ is the mutation value. After a trial vector is created for each of the 'p_{best}' particles in the swarm, they are evaluated and the 'p_{best}' and the 'g_{best}' particles of the swarm are changed if the offspring has a better fitness.

$$\delta_{2,j} = \frac{(P_{2,j} - P_{3,j}) + (P_{4,j} - P_{5,j})}{2} \tag{3}$$

$$U_j = P_{p,j} + \gamma \times \delta_{2,j} \tag{4}$$

δ_2 - difference vector averaged over 2 differentials
U - trial vector
γ - scaling factor
P_p - p_{best} particle used as a parent to generate offspring
P_{2-5} - p_{best} particles selected to form the difference vector

B. SPPSO

As the number of particles in the swarm of a conventional PSO algorithm increases, the convergence to a global solution is more and more ensured. The reason is, higher the number of particles, the greater the exploration of the search space. But, as the number of particles increases, the memory requirement for the algorithm also increases which is often not permissible in the real world application of the algorithm with digital signal processors or microcontrollers, etc. Also, the speed of convergence reduces a lot. In order to overcome these problems, SPPSO algorithm was developed by Das and Venayagamoorthy [4]. The concept of SPPSO is to start with a small number of particles (generally around five) and after a few iterations, replace all the particles except the global best with same number of regenerated particles. In this method, since the PSO runs with a very small number of particles, the memory requirement is reduced a lot. Also, since after few iterations a new set of particles are introduced, the chance of fixation to a local minima decreases and convergence is achieved much faster than conventional PSO.

C. Modified CSA

CSA is an integrated part of Artificial Immune System (AIS), which explains how an immune response is mounted when a non-self antigen is recognized by the B cells [13]. It is an evolutionary algorithm, where, during evolution, the antibodies which can recognize the antigens proliferate by cloning [14]. The term 'fitness' is equivalent to 'affinity' in AIS. The general steps involved in CSA are as follows:

- To create a population P of random solutions to the given problem.

- To evaluate the fitness of each member.

- To rank the population by fitness.

- While termination condition not met:

- To take the fittest N population members.

- To create n clones from each member of N, where n is proportional to the fitness of the member of N.

- To evaluate the fitness of the cloned members of N.

- To mutate each clone inversely proportionally to its fitness.

Given P and the mutated clones, to choose the best P members and form a new population.

In order to further increase the exploration capability of CSA, a concept of regeneration is also introduced in this paper. This is termed as Modified CSA (MCSA), where, if the best

fitness does not change for consecutive N iterations, half of the population is replaced by the regenerated random population.

D. PBIL

PBIL is a combination of Learning Vector Quantization (LVQ) and Genetic Algorithm. Basically, PBIL combines the strength of GA and LVQ. In PBIL, the role of population is redefined. The population is replaced by a single probability vector (PV) in PBIL [15]. The PV is created by counting the numbers of 1s and 0s in each gene position. This is done so that there is a higher probability of producing solutions similar to the best individual in the following generations. A brief summary of PBIL taken from [7] is provided below:

Step 1.   Initialize all dimensions of probability vector *PV* to 0.5 to ensure uniform random bitstrings.

Step 2.   Generate a population of uniformly random bitstrings by comparing with PV. If PV is greater than a random number the individual's dimension value is '1' else the dimension value is '0'.

Step 3.   Evaluate each individual bitstring as a solution to the problem using the objective function and identify the 'best' solution.

Step 4.   Update *PV* using (5) in favor of "best".

Step 5.   Mutate *PV* according to (6) to ensure diversity and sufficient coverage of the problem space.

Step 6.   Generate a new population with the modified *PV* stopping once an optimal solution is found, otherwise go to Step 2.

$$PV_i(j) = PV_i(j)*(1.0 - LR) + LR*BestTrial(j) \tag{5}$$

$$PV_i(j) = PV_i(j) - FF*(PV_i(j) - 0.5) \tag{6}$$

Where,

LR   - Learning Rate

FF   - Forgetting Factor

Best Trial   - Best solution of the population

i   - Particle number

j   - Dimension

III. TEST SYSTEM

The power system used for this study is the three-machine nine-bus power system shown in Fig. 1 [7]. Each of the three machines is a fourth order two-axis model equipped with a second order AVR. For the design of the PSS, several operating conditions are considered as test cases. These operating conditions include the nominal operating condition (case 1), light load under nominal (case 2) and weak (case 4) transmission line system, and heavy load under nominal (case 3) and weak (case 5) transmission line system. Tables I-III contain the system data for the five test cases.
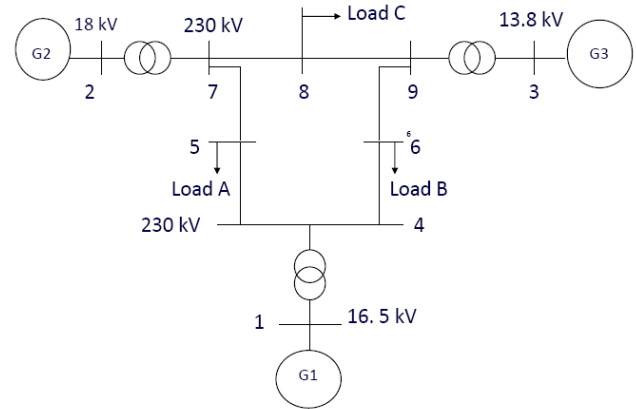

Figure 1.   Test Power System

TABLE I.   FIVE OPERATING CONDITIONS*

Gen #	G1		G2		G3	
Case	P_e	Q_e	P_e	Q_e	P_e	Q_e
1	0.716	0.321	1.630	-0.001	0.850	-0.118
2	0.505	-0.005	1.100	-0.241	0.300	-0.307
3	2.115	0.877	1.900	0.392	1.240	0.281
4	0.523	-0.674	1.100	-0.637	0.300	0.523
5	2.193	0.500	1.900	0.123	1.240	-0.086

*All values are in p.u.

TABLE II.   LOAD PARAMETERS* FOR THE FIVE CASES

Load	A		B		C	
Case	P_L	Q_L	P_L	Q_L	P_L	Q_L
1	1.250	0.500	0.900	0.300	1.000	0.350
2 & 4	0.750	0.300	0.540	0.180	0.600	0.210
3 & 5	1.750	0.700	1.620	0.540	1.800	0.630

*All values are in p.u.

TABLE III.   PARAMETERS OF THE TRANSMISSION LINES (NOMINAL)*

Line	Series Impedance (Z)	Shunt Admittance (B)
1-4	j0.0576	0.000
2-7	j0.0625	0.000
3-9	j0.0586	0.000
4-5	0.010 + j0.0085	0.176
4-6	0.017 + j0.0920	0.158
5-7	0.032 + j0.161	0.306
6-9	0.039 + j0.170	0.358
7-8	0.0085 + j0.072	0.149
8-9	0.0119 + j0.10008	0.209

*All values are in p.u.

IV. PROCEDURE FOR OPTIMAL TUNING OF PSS

The transfer function of a PSS is given by:

$$K(s) = K_p \left(\frac{T_w s}{1+T_w s} \right) \left(\frac{1+T_1 s}{1+T_2 s} \right) \left(\frac{1+T_3 s}{1+T_4 s} \right) \tag{7}$$

Where K_P is the PSS gain and T_1-T_4 represent the time constants and T_w is the washout time constant. Since T_w is generally constant, there are five parameters which are to be tuned. In this paper, there are three generators in the test system and each generator is equipped with a PSS. So, 15 parameters are to be optimized as a whole.

To obtain the optimum values for these parameters, the PSSs are tuned for five operating conditions as discussed earlier. The objective of this project is to maximize the smallest damping ratio from the five test cases. The Eigen values of the 3-machine power system are obtained for each of the five operating conditions. The damping ratios for each eigenvalue i and operating condition j are calculated using (8). The objective function J, given by (9), first determines the smallest damping ratio out of all cases and then tries to maximize that.

$$\zeta_{ij} = \frac{-\sigma_{ij}}{\sqrt{\sigma_{ij}^2 + \omega_{ij}^2}} \tag{8}$$

$$J = \max(\min(\zeta_{ij})) \tag{9}$$

Where $i = 1, 2,.....n$ and $j = 1, 2,.....m$

In order to have a fare comparison among all the four algorithms, same number of iterations/generations and same ranges of the parameters are used. Also, other than SPPSO, same population size is used for the remaining three algorithms. The values of different variables relevant for each algorithm are summarized in Table IV. The limits of the PSS parameters are given in Table V. For all the algorithms, the initialization is done randomly inside this limit of the PSS parameters. In case of SPPSO, the regeneration occurred after each set of ten iterations. In case of MCSA, the regeneration occurred if the best fitness remains unchanged for 20 consecutive iterations.

V. RESULTS

First, the system is run without PSS for the five cases mentioned before. It is found that for case 5, there is one pair of eigenvalues having positive real part (0.5324 ±12.3292i), which indicates that the system is unstable without PSS for the fifth operating condition. Now PSSs are connected to all the three generators and DEPSO, SPPSO, MCSA and the PBIL algorithms are used separately to find out the optimum parameters for the PSSs. Each algorithm is run for 10 times. Table VI presents the optimal parameters of the three PSSs found by the four algorithms. The values of the parameters

correspond to the best fitness achieved by each algorithm among the 10 trials.

As mentioned earlier, the objective of each algorithm is to maximize the minimum damping ratio obtained among the five cases. The minimum damping ratio among the five cases represents the fitness value for each algorithm. The best fitness value achieved by each algorithm is a measure of the strength of the algorithm. Higher the value better is the algorithm. Table VII compares the average best fitness achieved by the algorithms in 10 trials. It also presents the highest and the lowest values of the best fitness for the 10 trials.

TABLE IV. PARAMETER VALUES RELEVANT FOR THE ALGORITHMS

Parameters	DEPSO	SPPSO	MCSA	PBIL
Population size	20	5	20	20
Iteration	400	400	400	400
Learning rate	NA	NA	NA	0.1
Forgetting Factor	NA	NA	NA	0.005
Number of bits	NA	NA	NA	15
w	0.8	0.8	NA	NA
c_1	2.0	2.0	NA	NA
c_2	2.0	2.0	NA	NA
Velocity limit for K_p	±5.0	±5.0	NA	NA
Velocity limit for T_1-T_4	±0.5	±0.5	NA	NA
Cloning factor (β)	NA	NA	0.5	NA
Max. no. of antibodies	NA	NA	5	NA

TABLE V. PSS PARAMETER RANGES

$0 \leq K_p \leq 20$
$0.015 \leq T_1\text{-}T_4 \leq 1$

TABLE VI. OPTIMUM PSS PARAMETERS FOR THE FOUR ALGORITHMS

		DEPSO	SPPSO	MCSA	PBIL
PSS1	K_p	14.3510	19.9832	19.8666	13.1427
	T_1	0.0150	0.9782	0.3071	0.3416
	T_2	0.0150	0.5342	0.9334	0.3852
	T_3	0.1077	0.0150	0.3573	0.8743
	T_4	0.0150	0.0150	0.5064	0.3021
PSS2	K_p	5.4349	0.5000	16.7707	1.4905
	T_1	0.1661	1.0000	0.0582	0.2335
	T_2	0.0151	0.0150	0.3456	0.0150
	T_3	0.9901	0.8243	0.1607	0.9956
	T_4	1.0000	0.6723	0.5334	0.4006
PSS3	K_p	0.8497	0.5000	9.4243	1.1737
	T_1	0.3855	1.0000	0.3245	0.4374
	T_2	0.0294	0.9999	0.1732	0.0317
	T_3	0.0786	1.0000	0.0762	0.0683
	T_4	0.0150	0.0150	0.4953	0.0150

TABLE VII. AVERAGE, MAXIMUM AND MINIMUM FITNESS VALUES FOR THE FOUR ALGORITHMS

	DEPSO	SPPSO	MCSA	PBIL
Average Fitness	0.2013	0.1615	0.2110	0.2014
Maximum Fitness	0.2151	0.2019	0.2318	0.2347
Minimum Fitness	0.1710	0.0757	0.1832	0.1599

It can be observed from Table VII that the as a whole performance is best for MCSA. The average fitness achieved by it is 0.2110. This is the highest among the four algorithms. The maximum fitness achieved by MCSA is also very good (0.2318). The minimum value of the fitness for MCSA is greater than the other three algorithms. It is also observed that the performance of the DEPSO and PBIL are very close. Their average fitness values are almost same (0.2013 and 0.2014 respectively). But PBIL has got the highest value of the maximum fitness among the 10 trials (0.2347). The performance of SPPSO is not at all good compared to other algorithms. It has got the lowest values of maximum, minimum and average fitness.

The fitness vs. iteration characteristics for each algorithm is compared in Fig. 2. Here the fitness corresponds to the average fitness of the 10 trials. It is observed that the fitness characteristics are almost same for MCSA, DEPSO and PBIL up to 270 iterations. But after that, the fitness of MCSA is suddenly improved. It looks like it is suddenly coming out of the local minimum where the PBIL and DEPSO are trapped. The reason behind this is the regeneration concept which empowers the algorithm with more exploration capability. The regeneration is also there in SPPSO. In the figure, it looks like SPPSO has experienced the advantage of regeneration somewhere close to 330 iterations. Before that, it was trapped to a local minimum. Though it could come out of that local minimum, it could not reach the global minimum. A possible reason might be the very small population size, which is not sufficient for this particular PSS tuning problem.

In order to test the robustness of the tuned PSSs, the system is now run under another operating condition for which PSSs were not tuned. This operating condition is presented in Tables VIII and IX which is named as case 6. In this case, the lines 4-6, 4-5, 5-7, 6-9, 7-8 and 8-9 are doubled under nominal load condition. The parameters corresponding to the highest value of best fitness achieved by each algorithm among the 10 trials are used for each PSS. Table X compares the minimum damping ratio and the corresponding eigenvalues for the four algorithms under this operating condition.
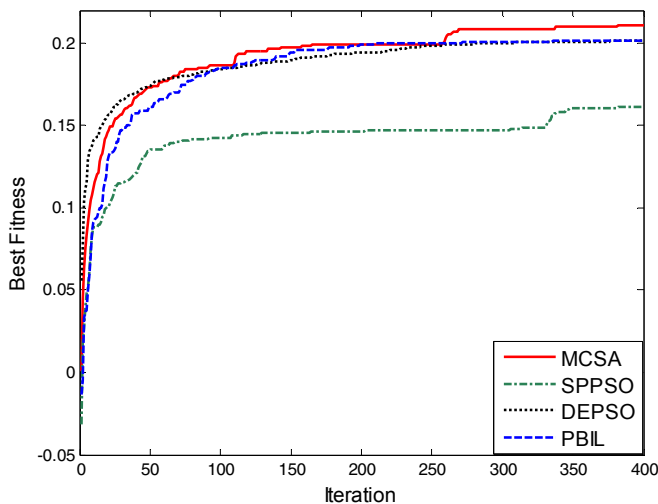
TABLE VIII.   Sixth Operating Conditions*

Gen #	G1		G2		G3	
Case	P_e	Q_e	P_e	Q_e	P_e	Q_e
6	0.75	-0.203	1.630	-0.363	0.850	-0.5865

*All values are in p.u.

TABLE IX.   Load Parameters* for the Sixth Case

Load	A		B		C	
Case	P_L	Q_L	P_L	Q_L	P_L	Q_L
6	1.250	0.500	0.900	0.300	1.000	0.350

*All values are in p.u.

TABLE X.   Minimum Damping Ratios and Corresponding Eigenvalues for Case 6

	Eigenvalue	Damping Ratio
DEPSO	-1.98±9.59i	0.2022
SPPSO	-1.71±10.13i	0.1665
MCSA	-2.89±11.90i	0.2360
PBIL	-2.64±11.04i	0.2326

From Table X, it is again observed that the MCSA is producing the best damping ratio among the four algorithms. The performance of the PBIL is very close to it. The DEPSO performance is moderate, though the minimum damping achieved by it is much less than the MCSA and PBIL. The worst performance is achieved by SPPSO having the least value of minimum damping ratio.

From all the results, it is clear that a consistently best performance is obtained from MCSA algorithm. PBIL and DEPSO are performing almost similar, though in some cases PBIL has outperformed the DEPSO algorithm. The performance of SPPSO is not up to the mark in this particular problem. Based on this, a ranking of all the four algorithms can be done as follows:

MCSA > PBIL > DEPSO > SPPSO

VI.   Conclusion

Comparison of the performances of four population based techniques, namely DEPSO, SPPSO, MCSA and PBIL for finding the optimal parameters of power system stabilizers in a multi-machine environment has been presented. The optimal tuning is achieved based on the frequency domain performances of the PSSs. Five different operating conditions are considered for the tuning process and a sixth operating condition is considered for testing the robustness of the tuned PSSs. The average performances of each algorithm are compared based on ten trial runs. It is observed that MCSA is consistently performing better than the other three algorithms. The performance of the DEPSO and PBIL are quite similar, though a slightly better performance is obtained in some cases with the PBIL algorithm. The performance of the SPPSO is not as good as the other three algorithms. Based on these observations, a ranking of these four algorithms in the PSS tuning problem is finally achieved in this paper.

Figure 2.   Average best fitness vs. iteration curve

REFERENCES

[1] P. M. Anderson and A. A. Foud, Power System Control and Stability, IEEE Press, 1994, ISBN 0-7803-1029-2.

[2] Abido, M.A, "Optimal design of power-system stabilizers using particle swarm optimization", IEEE Transaction on Energy Conversion, vol. 17, issue 3, Sept 2002, pp. 406-413.

[3] Abido, M.A, "Simultaneous Stabilization of Multimachine Power System via Genetic Algorithm", IEEE Transaction on Power Systems, vol. 14, No. 4, Nov 1999, pp. 1428-1439.

[4] T. K. Das, G. K. Venayagamoorthy, U. O. Aliyu, "Bio-Inspired Algorithms for the Design of Multiple Optimal Power System Stabilizers: SPPSO and BFA", IEEE Transactions on Industry Applications, Vol. 44, Issue 5, Sept-Oct 2008, pp. 1445-1457.

[5] Zhang, W. J., & Xie, X. F., "DEPSO: hybrid particle swarm with differential evolution operator," IEEE International Conference on Systems, Man and Cybernetics, Washington DC, USA, 2003, pp. 3816-3821.

[6] Y. Tan, Z. M. Xiao, "Clonal Particle Swarm Optimization and Its Application", 2007 IEEE Congress on Evolutionary Computation, pp. 2303-2309, 2007.

[7] Baluja S, "Population-Based Incremental Learning: A method for Integrating Genetic Search Based Function Optimization and Competitive Learning," Technical Report CMU-CS-94-163, Carnegie Mellon University, 1994.

[8] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez, R. G. Harley, "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems", IEEE Transactions on Evolutionary Computation, vol. 12, issue 2, pp. 171-195, April 2008.

[9] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", Proceedings of 1995 IEE Conference on Neural Networks, vol. 4, pp. 1942-1948.

[10] S. Mohaghegi, Y. del Velle, G. K. Venayagamoorthy and R. Harley, "A Comparison of PSO and Backpropagation for Training RBF Neural Networks for Identification of a Power System with STATCOM", Proceedings of IEEE Swarm Intelligence Symposium, 2005, pp. 381-384.

[11] Storn, R., & Price, K., "Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces," Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, March 1995.

[12] Karaboga, D., & Okdem, S., "A simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm," Turkish Journal of Electrical & Computer Engineering, Vol 12, 2004.

[13] X. Wang, "Clonal Selection Algorithm in Power Filter Optimization", Proceedings of the 2005 IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications, pp. 122-127.

[14] L. N. de Castro and J. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach, London, UK: Springer-Verlag, 2002.

[15] K. A. Folly, "Application of Evolution Algorithm in Power System Control Design", IEEE PES Power Africa Conference and Exposition, 2007.