Power System Controller Design using Multi-Population PBIL

Komla Agbenyo Folly
Department of Electrical Engineering
University of Cape Town
Cape Town, South Africa
Komla.Folly@uct.ac.za

Ganesh Kumar Venayagamoorthy
Real-Time Power and Intelligent Systems Laboratory
Holcombe Department of Electrical & Computer Eng.
Clemson University, SC 29634, USA
gkumar@ieee.org

***Abstract*—The application of a multi-population based Population-Based Incremental Learning (PBIL) to power system controller design is presented in this paper. PBIL is a combination of evolutionary optimization and competitive learning derived from artificial neural networks. Single population PBIL has recently received increasing attention in various engineering fields due to its effectiveness, easy implementation and robustness. Despite these strengths, PBIL still suffers from issues of loss of diversity in the population. The use of multi-population is seen as one way of increasing the diversity in the population. The approach is applied to power system controller design. Simulations results show that the multi-population PBIL approach performs better than the standard PBIL and is as effective as PBIL where adaptive learning is used.**

Keywords— Adaptive learning rate; low-frequency oscillations; multi-population; PBIL; power system stabilizer

I. INTRODUCTION

In the past few decades, there has been an increasing interest in biologically motivated approaches to solving optimization problems, including Artificial Neural Networks (ANNs), Genetic Algorithms (GAs), and Evolution Strategies (ESs). GAs are one of the most used Evolutionary Algorithms (EAs) for function optimization. They are robust and can be applied to a wide range of problems [1]. However, GAs still have several limitations, such as genetic drift which prevents GAs from maintaining the diversity in the population, the difficulty in selecting optimal GA operators such as selection and mutation rates, crossover probability, and population size [2]-[4].

To cope with the above limitations, several other biological motivated algorithms have been proposed in the last few years. These include Differential Evolution (DE) [5]-[6], Artificial Immune Systems (AIS) [7], Bacterial Foraging Algorithm (BFA) [8], Ant Colony (ACO) [9], and Particle Swarm Optimization (PSO) [10] which belong to the family of Swarm intelligence. Among all these algorithms, PSO has been widely used for parameter optimization in controller design [11]-[12]. PSO is simple and easy to implement; however, the algorithm is very sensitive to some of its parameters such as inertia weights, and acceleration factors. In addition, the optimal selection of these parameters may be difficult to achieve.

The financial supports of the TESP and THRIP as well as the NSF EFRI # 1238097 grants are acknowledged.

Recently, a new type of EA called Population-Based Incremental Learning (PBIL) has been proposed by Baluja [13], [14]. PBIL is simpler and more effective than GAs. In addition PBIL has less overhead than GAs [15]-[16]. There are few parameters to tune in PBIL as compared to GAs or PSO. This makes PBIL more attractive to a wide range of researchers.

In PBIL, the crossover operator is abstracted away and the role of population is redefined [13], [14]. PBIL works with a probability vector (PV) instead of the whole population. One only needs to store a single PV and the solution being evaluated. The PV is used to control the random bit strings generated by PBIL and to create other individuals through learning. Learning in PBIL consists of using the current probability vector (PV) to create N individuals. The best individual is used to update the probability vector, increasing the probability of producing solutions similar to the current best individuals [14]. It has been shown that PBIL outperforms standard GAs approaches on a variety of optimization problems including commonly used benchmark problems [14]-[18].

In [19]-[22], the standard PBIL with fixed learning rate was used to design power system controllers known as power system stabilizers (PSSs). The main purpose of a PSS is to damp low frequency oscillations arising from small disturbances, such as gradual and small changes in loads or generations. However, a fixed learning rate may not be effective in dynamic environment to maintain the required trade-off between exploration and exploitation [23]-[25]. Also, the use of one probability vector (PV) to represent the whole population may reduce the diversity in the population and thereby degrading the global search ability of the algorithm [26].

Recently some authors have reported that PBIL suffers from diversity loss making the algorithm to converge to local optima [27]-[28]. Maintaining the diversity in PBIL's population is directly linked to the learning rate. In [23], the authors investigated the effect of learning rate on the performance of PBIL. It was shown that using adaptive learning rate where the learning rate starts with a very small value and increases monotonically with the generation provides better results as compared to using a fixed learning rate. The authors in [24] used hyper-learning for PBIL where the

learning rate is temporary raised whenever the environment is changed. In [25], opposition-based computing is used to alter the probability update rule with a more advanced mutation rule. However, the authors introduce new parameters which increase the complexity of the algorithm.

In [26], adaptive updating of the learning rate was used where the learning rate was varied according to the characteristics of specific search process. Maintaining the diversity in PBIL's population was achieved through the use of multiple probability vectors where every individual uses different probability vectors to generate its own children. In [29], multi-population PBIL is used to solve dynamic optimization problems. The ideas of parallel PBIL and dual PBIL are introduced. In parallel PBIL, the population is divided into two parts. Two probability vectors (PVs) are then used. One of these PVs is initialized to 0.5 and the other PV is randomly initialized. In Dual PBIL, the idea of complementary or duality is introduced and the PVs are initialized to complement each other.

In this paper, the idea of parallel PBIL (PPBIL) using multi-population where both PVs are initialized to 0.5 is explored. The proposed approach is applied to power system controller design. The idea of parallel PBIL is not new, but to the knowledge of the authors it is the first time that it has been applied to controller design in power systems. The effectiveness of the proposed algorithm is demonstrated by comparing it to the Adaptive PBIL (APBIL) introduced in [23] and the standard PBIL with fixed learning rate. Simulation results show that the parallel PBIL based on multi-population is as effective as APBIL and better than the standard PBIL.

II. Overview of Population-Based Incremental Learning

PBIL is a technique that combines aspects of GA and simple competitive learning derived from artificial neural networks [13], [14]. PBIL belongs to the family of Estimation of Distribution Algorithms (EDAs), which use the probability (or prototype) vector (PV) to generate sample solutions. There is no crossover operator in PBIL; instead a single probability vector is updated using solution with the highest fitness values [16]. Initially, the values of the probability vector are set to 0.5 to ensure that the probability of generating 0 or 1 is equal. As the search progresses, these values are moved away from 0.5, towards either 0.0 or 1.0.

Like in GA, mutation is also used in PBIL to maintain diversity. In this paper, the mutation is performed on the probability vector rather than on the sample solutions. A forgetting factor is used to relax the probability vector toward a neutral value of 0.5 [15], [20]-[23]. A summary of the PBIL used in this paper is given below [20]-[23]:

Step 1. Initialize element of the probability vector (PV) to 0.5 to ensure uniformly-random bit strings.

Step2. Generate a population of uniformly-random bit strings and comparing it element-by-element with the PV. Wherever an element of the PV is greater than the corresponding random element, a "1' is generated, otherwise a '0' is generated.

Step 3. Interpret each bit string as a solution to the problem and evaluate its merit in order to identify the "Best".

Step 4. Adjust PV by slightly increasing PV (i) to favor the generation of bit strings which resemble "Best", if Best (i) = 1 and decrease PV(i) if Best(i) = 0.

Step 5. Apply the mutation and generate a new population reflecting the modified distribution. Stop if satisfactory solution is found. Otherwise, go to step 2.

The update rule of the probability vector is given as follows:

$$PV_i(g+1) = (1-LR) \times PV_i(g) + LR * B_i(g) \quad (1)$$

where $0 \le LR \le 1$ is the learning rate that determines the distance the probability vector is pushed for each iteration, PV is the probability vector, g is the number of iterations or generations, i denotes each locus ($i = 1, 2\ldots, l$), and l is the binary encoding length, B is the best solution.

The mutation used for the PBIL adopted in this paper is slightly different from the one initially used in [13], [14]. The main idea is to relax the PV towards the neutral 0.5. The pseudocode for PBIL is shown in Fig. 1.

The probability vector guides the search and produces the next sample point from which learning takes place. The learning rate determines the speed at which the probability vector is shifted to resemble the best (fittest) solution vector. If the learning rate is fixed during the run (as it is the case with standard PBIL), it cannot provide the flexibility needed to achieve a trade-off between exploration and exploitation. Therefore, there is a possibility of premature convergence. Diversity in the population can be lost if the degree of exploitation is higher than that of exploration. For the algorithm to perform optimally, it is crucial that diversity is not lost early in the run and that balance between exploration and exploitation is maintained. In the next section two methods to increase diversity will be discussed.

Begin
g:= 0;
//initialize probability vector
for i:=1 to l, do $PV_i^0 = 0.5$;
endfor;
while not termination condition **do**
generate sample $S(g)$ from (PV(g) , pop.)
Evaluate samples $S(g)$
Select best solution $B(g)$
// update probability vector $PV(g)$ toward best solution according to (1)
//mutate $PV(g)$
Generate a set of new samples using the new probability vector
g=g+1
end while // e.g., g>G_{max}

Fig. 1. Pseudocode for standard PBIL

III. Overview of PBIL with Adaptive Learning Rate

The learning rate in the standard PBIL is usually fixed to a specific value. This means that the user has to spend a lot of time and try several values of the learning rate before deciding on the "best" value to use. If the learning rate value is too high, the algorithm will learn towards the best too quickly. This may lead to premature convergence and the lost of the diversity earlier in the run. The algorithm could converge to local optima. If on the other hand, the learning rate is too small, the algorithm may be slow to converge and will require more time to find the optimal solution. This may be computationally costly and a waste of resources. It is therefore crucial that the learning rate provides a trade-off between exploration and exploitation.

To develop the adaptive learning algorithm, we assume that at the start of the run, diversity will be needed for the algorithm to be able to explore thoroughly the search space. Therefore, at the start, a very small value of learning rate (LR≈ 0) is selected.

This means that at the beginning of the run, the emphasis is placed on the exploration rather than exploitation. As the run progresses and good individuals start to emerge, the emphasis is shifted gradually from exploration to exploitation. The learning rate is allowed to increase slowly and linearly according to the change in generation as given in the following equation:

$$LR(i) = LR * G(i) / G_{max} \quad (2)$$

where

LR(i) is the learning rate at the *ith* generation

LR is the final learning rate

G(i) is the *ith* generation

G_{max} is the maximum generation allowed

The pseudocode for APBIL is similar to that of standard PBIL except that the learning rate is now varying according to (2).

IV. Overview of PBIL with Adaptive Learning Rate

For the Parallel PBIL (PPBIL), two populations are used with two probability vectors (PV_1 and PV_2). Each probability vector is initialized to 0.5 and sampled to generate solutions independently from each other. The PVs are updated independently according to the best solution generated by each. Initially, each probability vector has equal sample solutions. That is, the total population is divided by 2 and one half assigned to each PV. As the run progresses, the probability vector (PV) that performs better is allowed to increase its share of samples. The sample sizes of the probability vectors are slightly adapted within the range [pop_{min} pop_{max}] = [0.4*pop 0.6*pop] according to their relative performances. The probability that outperforms the other is increased by a constant value Δ = *LR**pop, where *LR* is the learning rate. For the PPBIL used in this paper, the learning rate was selected to be 0.1 for both probability vectors. Fig. 2 shows the pseudocode of PPBIL.

Begin
g:= 0;
//initialize probability vector
for *i*:=1 to *l*, do $PV_{i1}^0 = PV_{i2}^0 = 0.5$;
endfor;
// initialize the sizes of the probability vectors such that: pop1= pop 2= pop/2
while not termination condition **do**
generate sample $S_1(g)$ from ($PV_1(g)$, pop1.)
generate sample $S_2(g)$ from ($PV_2(g)$, pop2.)
Evaluate samples ($S_1(g)$, $S_2(g)$)
Select best solutions $B_1(g)$*and* $B_2(g)$
// update probability vectors $PV_1(g)$ and $PV_2(g)$
toward bests solution $B_1(g)$and $B_2(g)$ according to (1)
If $f(B_1(g)) > f(B_2(g))$
then $pop_1 = \min [(pop_1 + \Delta) \quad pop_{max}]$
If $f(B_1(g)) < f(B_2(g))$
then $pop_1 = \max [(pop_1 - \Delta) \quad pop_{min}]$
$pop_2 = pop - pop_1$
//mutate $PV_1(g)$ and $PV_2(g)$
g=g+1
end while // e.g., g>G_{max}

Fig. 2. Pseudocode of PPBIL

V. System model and Operating conditions

The power system considered in this paper is the two-area four-machine power system as shown in Fig. 3 [30]. Each machine is represented by the detailed six order differential equations. The machines are equipped with simple exciter systems. The dynamics of the system are described by a set of nonlinear differential equations. The system consists of two similar areas connected by a tie-line. Each area consists of two coupled conventional generator units, each generator is rated 900 MVA and 20 kV. The generators are modeled as round-rotor generators and represented by the detailed six order differential equations. The dynamics of the system are described by a set of nonlinear differential equations. However, for the purpose of controller design these equations are linearized around the nominal operating conditions.

The linearized equation of the system is given by [22], [23]

$$\begin{aligned} \dot{x} &= A_o x + B_o u \\ y &= C_o x + D_o u \end{aligned} \quad (3)$$

where

x are the state variables, *y* the system output and *u* the control input. A_o, B_o, C_o, D_o are constant matrices of appropriate dimensions.

Three operating conditions are considered during the design of the controller. These are listed in Table I. Case 1 is the light load condition, where about 200 MW of real power is transferred from area 1 to area 2. Case 2 is the nominal condition, under this operating condition, there is a transfer of 400 MW power from area 1 to area 2. Case 3 is the heavy load condition where about 500 MW of power is transferred from area 1 to area 2. The system exhibits two local modes one in area 1 and the other in area 2 and one inter-area mode. However, for the purpose of this study only the inter-area mode will be considered since it is the most difficult to control.

Matlab Power System Toolbox (PST) was used for all the simulations.

TABLE I. OPERATING CONDITIONS

cases	Active Power Flow (MW)	Eigenvalues
1	200	-0.35±j3.92
2	400	0.0096±j3.84
3	500	0.148±j3.09

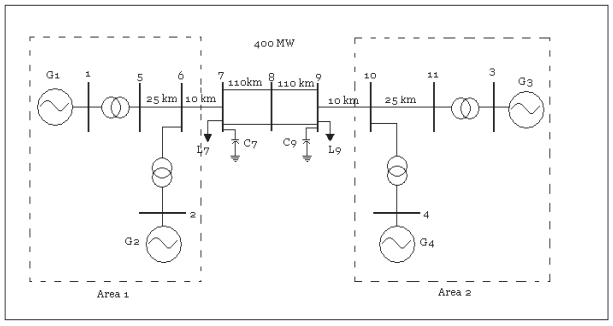

Fig. 3. System model

VI. PROBLEM STATEMENT

As can be seen in Table I, except for the light load condition, the system is unstable for the nominal and heavy operating conditions. This instability can be explained by the positive values of the real part of the eigenvalues. These oscillations should be damped for the system to perform adequately.

The purpose of the design is to optimize the parameters of the generator excitation controls (i.e., PSS) simultaneously and in a coordinated and decentralized manner such that adequate damping is provided to the system for the operating conditions considered in this paper, while keeping the structure of the PSS as simple as possible. The structure of the widely used conventional PSS was adopted here [30]. The objective function used is given in (4)

$$J = \max(\min(\zeta_{ij})) \tag{4}$$

$i = 1, 2, \ldots, n$ *eigenvalues*

$j = 1, 2, \ldots, m$ *operating conditions*

where $\zeta_{ij} = \dfrac{-\sigma_{ij}}{\sqrt{\sigma_{ij}^2 + \omega_{ij}^2}}$ is the damping ratio of the i^{th} closed – loop eigenvalue of the j^{th} operating condition. σ_{ij} is the real part of the eigenvalue and ω_{ij} is the frequency.

VII. CONTROLLER DESIGN

A. Structure of the Controller

The structure of the controller to be designed is as shown in (5). It is required to simultaneously optimize the parameters of the controller such that adequate damping is provided for a wide range of operating conditions. In total 10 parameters are to be optimize (i.e., 5 parameters for each area). The washout parameter T_w is not critical and has not been optimized but was chosen to be 10 s.

$$K(s) = K_p \left(\frac{sT_w}{1+sT_w} \right) \left(\frac{1+sT_1}{1+sT_2} \right) \left(\frac{1+sT_3}{1+sT_4} \right) \tag{5}$$

In (5), T_1 to T_4 represent the time constants that need to be optimized to obtain adequate damping.

B. Application of Standard PBIL to Controller Design

The parameter's configuration that was used in SPBIL is as follows:

Population: 10

Generation: 400

Learning rate: 0.1

Forgetting factor: 0.005

C. Applicationof Adaptive PBIL to Controller Design

The parameter's configuration that was used in APBIL is as follows:

Population: 10

Generation: 400

Initial Learning rate: 0.0005

Final Learning rate: 0.2

Forgetting factor: 0.005

D. Applicationof Parallel PBIL to Controller Design

The parameter's configuration that was used in PPBIL is as follows:

Population: 10

Initial population for PV_1: 5

Initial population for PV_2: 5

Maximum population: 6

Minimum population: 4

Generation: 400

Learning rate: 0.1

Forgetting factor: 0.005

For all the controllers, the parameter domain is as follows:

$$0 \le K_p \le 30$$

$$0 \le T_1, T_3 \le 1$$

$$0.010 \le T_2, T_4 \le 0.3$$

Table II shows the parameters of the controllers that provide the best fitness values. It can be seen that the gains of all the controllers are similar. Time constant T_1 varies widely from a maximum of 0.9929 (SPBIL) to a minimum of 0.0609 (PPBIL). T_2 varies from a maximum of 0.2997 (APBIL) to a minimum of 0.01 (SPBIL, APBIL, PPBIL). For all the approaches, T_1 and T_2 form lead networks. T_3 varies from a maximum of 0.9987 (PPBIL) to a minimum of 0.0534 (APBIL); whereas T_4 varies from a maximum of 0.30 (APBIL, PPBIL) to a minimum of 0.01 (APBIL). T_3 has the largest variations followed by T_1; whereas, T_2 and T_4 vary in the same range.

The singular value Bode plots of the three controllers are shown in Fig. 4. It can be seen that at low frequencies below 1 rad/sec all the controllers have the same shapes. At higher frequencies, APBIL gain is smaller than the gains of PPBIL and SPBIL. The shapes of PPBIL and SPBIL are almost similar at high frequencies.

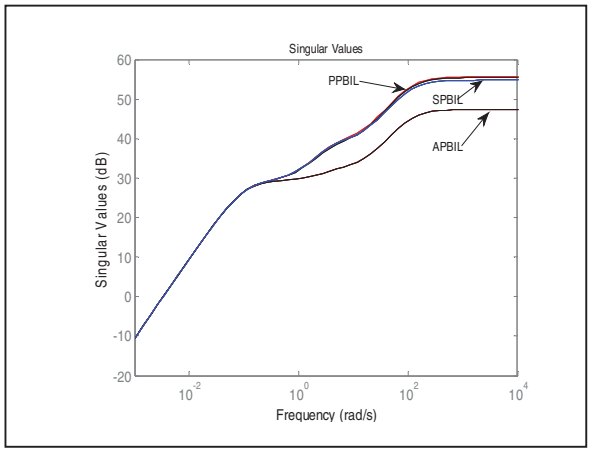

Fig. 4. Singular value Bode plots of the controllers

VIII. Comparison of SPBIL, APBIL, and PPBIL

In order to investigate the effectiveness of the PBIL algorithms, various aspects the algorithms are compared such as the convergence rate, the best fitness values and the capability of the algorithm in maintaining the diversity in the population. For each algorithm, several trials were run and the best fitness value curves are selected.

Figs. 5-7 show the convergence rate of SPBIL, APBIL and PPBIL. It can be seen that APBIL and PPBIL converge to highest fitness values compared to SPBIL. APBIL converged to a slightly higher value of 0.5140 (which occurs at generation 395) compared to 0.5018 for PPBIL. From the simulation results, it can be seen that APBIL has more diversity in the population at the middle of the run between generation 100 and 200 than PPBIL and SPBIL. This can be attributed to the small value of learning rate.

Table III shows the comparison between minimum, maximum and average fitness values. It can be seen that on average SPBIL and PPBIL have practically the same fitness. This average (approximately 0.434) is higher than the average of APBIL. The main reason for this is that APBIL has the lowest minimum fitness value at 0.09 compared to PPBIL at 0.1243 and SPBIL at 0.1663. In terms of the distance between the maximum and the minimum fitness values, APBIL has the highest, followed by PPBIL. This suggests that both APBIL and PPBIL have more diversity than SPBIL.

TABLE II. Controller Parameters

		K_p	T_1	T_2	T_3	T_4
SPBIL	Gen 1&2	29.989	0.9929	0.0100	0.0555	0.2998
	Gen 3&4	29.6265	0.1283	0.0345	0.1274	0.1620
APBIL	Gen 1&2	29.9652	0.9464	0.2997	0.0626	0.0100
	Gen 3&4	29.8773	0.4391	0.0100	0.0534	0.3000
PPBIL	Gen 1&2	29.8892	0.0609	0.0100	0.9987	0.2999
	Gen 3&4	29.5413	0.3602	0.2769	0.0710	0.0251

TABLE III. Maximum, Average and Minimun Fitness Values

Fitness	SPBIL	APBIL	PPBIL
Maximum	0.4835	0.5140	0.5018
Average	0.4343	0.3788	0.4339
Minimum	0.1663	0.0900	0.1243

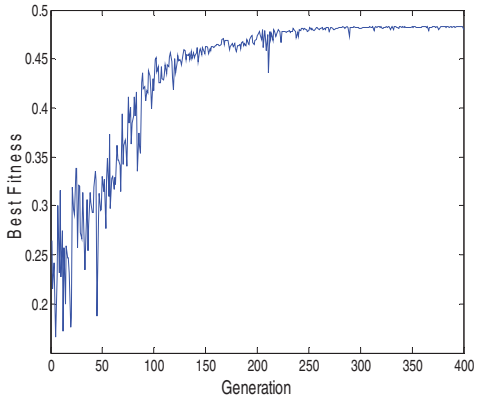

Fig. 5. SPBIL convergence rate

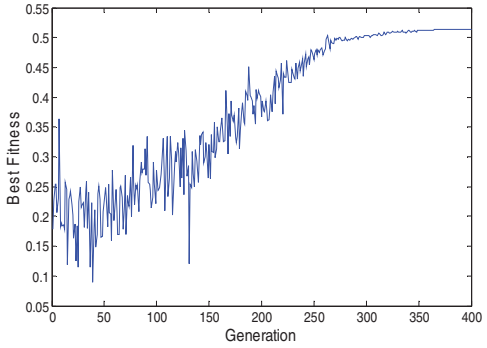

Fig. 6. APBIL convergence rate

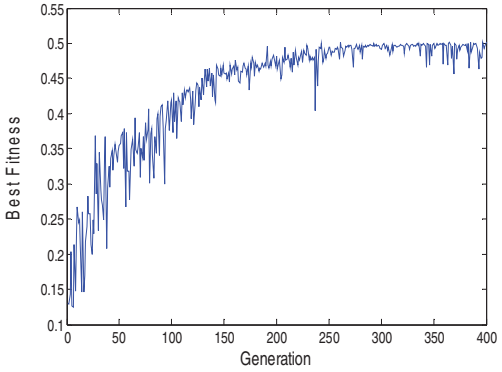

Fig. 7. PPBIL convergence rate

IX. SIMULATION RESULTS

Table IV shows the eigenvalues and damping ratios in brackets of the closed-loop systems with the three controllers. Only the inter-area mode is considered. It can be seen that PPBIL gives a better performance than SPBIL. The performance of APBIL is similar to that of PPBIL. This suggests that by maintaining the diversity in the population of PBIL longer, we are able to improve the performance of the controllers.

TABLE IV. EIGENVALUES AND DAMPING RATIOS

Case	SPBIL	APBIL	PPBIL
1	-1.13 ± j2.04 (0. 48)	-1.54 ± j2.57 (0.51)	-1.53 ± j2.53 (0.52)
2	-0.778 ± j1.78 (0.44)	-1.26 ± j2.11 (0. 51)	-1.26 ± j2.08 (0. 52)
3	-0.582 ± j1.25 (0.42)	-1.15 ± j1.52 (0.61)	-1.14 ± j1.54 (0.60)

X. CONCLUSION

The performance of a parallel PBIL based on multi-population that uses two probability vectors to increase the diversity in the population has been investigated in this paper. The effectiveness of the proposed controller is assessed by comparing it to the Adaptive PBIL (APBIL) and the standard PBIL (SPBIL). It is shown that both the PPBIL and APBIL perform better than SPBIL in maintaining the diversity in the populations and improving the damping of the system. Simulations results show that APBIL is slightly better in maintaining the diversity in the population. In the future, comparative studies with other parallel EA will be carried out and the results extended to a much larger power system.

REFERENCES

[1] J. H. Holland, Adaptation in Nature and Artificial Systems. The University of Michigan Press, 1975.

[2] L. Davis, Handbook of Genetic Algorithms,. International Thomson Computer Press, 1996.

[3] D. E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, 1989.

[4] J. Yao, N. Kharma, & P. Grogono, "Bi-objective Multipopulation Genetic Algorithm for Multimodal Function Optimization," IEEE Trans. Evol. Comput., 14 (1). 80-102, 2010.

[5] K. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: Apractical Approach to Global Optimization, Springer, ISBN 978-3-540-20950-8, 2005.

[6] T. Mulumba, K. A. Folly, "Power System Stabilizer Design: Comparison Analysis between Differential Evolution and Population Based Incremental Learning", in: 20th Southern African Universities' Power Engineering Conference (SAUPEC), 2011.

[7] de Castro, L. N. & Timmis, J. (2002), Artificial Immune Systems: A New Computational Intelligence Approach, Springer-Verlag, London., 2002.

[8] TK Das, GK Venayagamoorthy, UO 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, September/October 2008, pp. 1445 – 1457.

[9] Dorigo, M. & Di Caro, G. (1999), "The Ant Colony Optimization Meta-Heuristic", In New Ideas in Optimization, D. Corne, M. Dorigo & F. Glover (eds.), McGraw-Hill,1999, pp. 11-32.

[10] J. F. Kennedy, R. C. Eberhart R.C., & Y. Shi, Swarm Intelligence. Morgan Kaufmann, 2001.

[11] T. K. Das, and G.K. Venayagamoorthy "Design of Power System Stabilizers using Small Population Based PSO," IEEE PES General Meeting 2006.

[12] H. Shayeghi, A Safari, H. A. Shayanfar, "Multimachine Power System Stabilizers Design using PSO Algorithm", International Journal of Elec. Power and Energy Syst. Eng. 1, 2008, pp. 226-233.

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

[14] .S. Baluja, R. Caruana, R., "Removing the Genetics from the Standard Genetic Algorithm," Tech. Rep. CMU-CS-95-141), Carnegie Mellon University, 1995

[15] J. R. Greene, "Population-Based Incremental Learning as a Simple,Versatile Tool for Engineering Optimization,"In Proceedings of the First International Conf. on EC and Applications, 1996, pp.258-269.

[16] F. Southey, F. Karray, "Approching Evolutionary Robotics through Population-Based Incremental Learning," in: IEEE International Conference on Systems, Man and Cybernetics, Vol. 2, 1999, pp. 710-715.

[17] Q. Jiang, Y. Ou, D. Shi-Du, "Optimizing Curriculm Scheduling Problem using Population-Based Incremental Learning Algorithm," in: Second Workshop on Digital Media and its Application in Museum and Heritages, 2007, pp. 448-453.

[18] Gosling,T., Jin, N., Tsang, "Population-Based Incremental Learning Versus Genetic Algorithms: Iterated Prisoners Dilemma", Tech. Rep. CSM-40, University of Essex, England, 2004.

[19] K.A. Folly, "Robust Controller Design Based on a Combination of Genetic Algorithms (GAs) and Competitive Learning," In: International Joint Conference on Neural Networks, 2007, pp. 3045-3050.

[20] K.A. Folly, "Design of Power System Stabilizer: A Comparison Between Genetic Algorithms (GAs) and Population-Based Incremental Learning (PBIL),". In Proc. of the IEEE PES 2006 General Meeting, 2006.

[21] P. Mitra, C. Yan, L. Grant, G.K. Venayagamoorthy, K. Folly, " Comparative Study of Population-Based Techniques for Power System Stabilizer Design,", in Proc. of Intelligent System Applications to Power Systems, 2009.

[22] S. Sheetekela., K.A. Folly, "Power System Controller Design: A Comparison Between Breeder Genetic Algorithm (BGA) and Population-Based Incremental Learning (PBIL)," in: Proc. of the Int. Joint Conference on Neural Networks –IJCNN, 2010.

[23] KA Folly, G.K. Venayagamoorthy, "Effect of Learning Rate on the Performance of the Population-Based Incremental Learning Algorithm," in: Proc. of the International Joint Conf. on Neural Network (IJCNN), 2009.

[24] S. Yang and H. Richter, "Hyper-Learning for Population-Based Incremental Learning in Dynamic Environment," in: IEEE Congress on Evolutionary Computation, 2009.

[25] M. Ventresca, H. R. Tizhoosh, "A diversity Maintaining Population Based Incremental Learning Algorithm", Information Sciences, 178, 2008, pp. 4038-4056.

[26] S. Y. Yang, S.L. Ho, G.Z. Ni, J.M. Machado and K.F. Wong, "A new Implementation of Population-Based Incremental Learning Method for Optimizations in Electromagnetics", IEEE Trans. On Magnetics 43 (4), 2007, 1601-1604.

[27] C. Conzalez, J.A. Lozano and P. Larranaga, " The convergence behavior of the PBIL Algorithm: A preliminary approach," in: Proc. of Artificial Neural Nets and Genetic Algorithms, 2001.

[28] R. Rastegar, A. Hariri, M. Mazoochi, "The Population-Based Incremental Learning Algorithm Converges to Local Optima," Neurocomputing, 69, 2006, pp. 1772-1775.

[29] S. Yang and X. Yao, "Experimental Study on Population-Based Incremental Learning Algorithms for Dynamic Optimization Problems", Soft Computing, 9(11), 2005, pp. 815-834.

[30] P. Kundur, Power System Stability and Control. McGraw – Hill, Inc. 1994.