DSP-Based PSO Implementation for Online Optimization of Power System Stabilizers

Parviz Palangpour, Pinaki Mitra, Swakshar Ray and Ganesh K. Venayagamoorthy
Real-Time Power and Intelligent Systems Laboratory
Missouri University of Science and Technology
pmpv3b@mst.edu, pm33d@mst.edu, sr8r8@mst.edu and gkumar@ieee.org

Abstract

Real-time implementations of controllers require optimization algorithms which can be performed quickly. In this paper, a Digital Signal Processor (DSP) implementation of Particle Swarm Optimization (PSO) is presented. PSO is used to optimize the parameters of two stablizers used in a power system. The controllers and PSO are both implemented on a single DSP in a hardware-in-loop configuration. Results showing the performance and feasibility for real-time implementations of PSO are presented.

1 Introduction

System-level adaption has become a large area of interest since many systems operate in changing, unpredictable environments. There are two main components to these adaptive systems, a reconfigurable platform and an algorithm or mechanism to optimize the systems behavior. Most often, a programmable device is coupled with an Evolutionary Algorithm (EA). While the system or controller is implemented on the programmable device, an EA is used to tune the performance of the system.

Adaptive systems have been implemented on a variety of platforms, each allowing different degrees of flexibility. Common platforms for this application include Field Programmable Gate Arrays (FPGAs), Digital Signal Processors (DSPs) and Application Specific Integrated Circuits (ASICs). ASICs are simply integrated circuits that are designed specifically for one application. These devices typically consume less power and are physically smaller than comparable DSPs and FPGAs. After fabrication, the hardware architecture of ASICs are fixed, though authors have designed ASICs that have values that can be tuned post-fabrication [6]. DSPs are microprocessors, so while their hardware architecture is fixed like ASICs, they run software that can be changed, offering much more flexibility. While the hardware structure of ASICs and DSPs are not flexible, FPGAs can be reconfigured at the hardware level; as a result, FPGAs exhibit the highest level of flexibility. However, FPGAs also consume the most power, require more physical space and offer much slower operating speeds. Although ASICs can be designed to operate at high speeds and consume both minimal power and space, they are very expensive to design. DSPs offer many advantages as a reconfigurable platform for adaptive systems: flexibility, high-speed operation and moderate power consumption.

Many different variations of EAs for adapting system behavior have been extensively explored. In principle, all EAs are population-based optimization algorithms. The population consists of candidate solutions to the problem being studied and during each iteration of the algorithm a series of operators are applied to the population. After the population has been passed through the operators, the candidate solutions are evaluated and given a level of 'fitness' that represents their degree of performance for the problem being studied. Each of the operators are based on evolution and play a role in combining and randomly modifying portions of the population. As the fitness of candidate solutions play a role in what solutions are selected to combine and modify, the population as a whole improves over time. Particle Swarm Optimization (PSO) is another population-based algorithm which begins with a population of potential solutions and continually evolves the solutions until they reach a desired level of fitness [4]. While EAs and PSO are similar, PSO requires less operations. This allows PSO to be used in real-time applications where speed is critical. Recently, authors have used PSO to tune analog PID controllers, though the analog controller and PSO are implemented on separate hardware [2].

In this study, a successful implementation of PSO on a DSP is achieved in a real-time environment. Power System Stabilizers (PSSs) are lead-lag compensator controllers used in power systems. In [1], PSO was used to tune the parameters of PSSs and illustrated in a non-real-time simulation environment. Here, the PSO technique is applied to

978-0-7695-3166-3/08 $25.00 © 2008 IEEE
DOI 10.1109/AHS.2008.70

IEEE computer society

Authorized licensed use limited to: University of Missouri. Downloaded on February 2, 2009 at 11:54 from IEEE Xplore. Restrictions apply.

obtain the optimal parameters of PSSs for a power system simulated in real-time. The two PSSs controlling two generators are implemented in a DSP and the remainder of the power system is simulated in a Real Time Digital Simulator (RTDS). The entire tuning process is done with a hardware-in-loop arrangement.

The paper is organized as follows. In Section 2, the PSO algorithm is explained. In Section 3 we describe the benchmark power system used and how it was simulated in real-time. Section 4 describes the digital PSS controller implementation. Our DSP implementation of PSO and the PSSs is described in Section 5. The results from the DSP implemented PSSs can be found in Section 6. Section 7 summarizes and concludes this work.

2 Particle Swarm Optimization

The PSO algorithm was developed by Kennedy and Eberhart and is based on the social behavior of bird flocking [4]. Each particle in the population has a position vector which represents a potential solution to the problem. The particles are initialized to random positions throughout the search space and for each iteration of the algorithm a velocity vector is computed and used to update each particles position. Each particles velocity is influenced by the particles own experience as well as the experience of its neighbors.

There are two basic variants of PSO, *local* and *global*. In this study the global version of the PSO algorithm is applied. The population consists of n particles. For each iteration, a cost function f is used to measure the fitness of each particle i in the population. The position of each particle i is then updated, which is influenced by three terms, the particles velocity from the last iteration, the difference between the particles known best position and the particles current position, and the difference between the swarms best known position and the particles current position. The latter two terms are each multiplied by a random number in [0,1] to randomly vary the influence of each term, as well as an acceleration coefficient to scale and balance the influence of each term. The best position each particle attained is stored in the vector p_i, also known as p_{best}, while the best position attained by any particle in the population is stored in the vector p_g, also known as g_{best}. The velocity vector $v_i(t)$ for each particle is then updated.

$$v_i(t+1) = v_i(t) + c_1\rho_1(p_i - x_i(t)) + c_2\rho_2(p_g - x_i(t)) \quad (1)$$

where c_1 and c_2 are positive and ρ_1 and ρ_2 are uniformly distributed random numbers in [0,1]. The term c_1 is called the cognitive acceleration term and c_2 is called the social acceleration term. These two values balance the influence between the particles own best performance and that of the population. The velocity is constrained between the parameters V_{min} and V_{max} to limit the maximum change in position.

$$v_i(t+1) = \begin{cases} v_{Max} & \text{if } v_i(\text{t + 1}) > V_{max} \\ v_{Min} & \text{if } v_i(\text{t + 1}) < V_{min} \\ v_i(t+1) & \text{else} \end{cases} \quad (2)$$

The position of each particle is then updated using the new velocities.

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (3)$$

The position in each dimension is limited between the parameters X_{min} and X_{max}.

$$x_i(t+1) = \begin{cases} x_{Max} & \text{if } x_i(\text{t + 1}) > X_{max} \\ x_{Min} & \text{if } x_i(\text{t + 1}) < X_{min} \\ x_i(t+1) & \text{else} \end{cases} \quad (4)$$

The original PSO algorithm uses static parameters, though several studies have shown that using dynamic parameters for PSO can greatly improve the convergence speed and reduce the probability of converging on a local minima [3] [10] [5] [9]. Many approaches use parameters that are time-varying and typically favor global exploration at the beginning of the search and local search toward the end. Shi and Eberhart suggested using an inertia term, w, in the velocity update equation to control the amount of momentum added [8].

$$\begin{aligned} v_i(t+1) = w * v_i(t) &+ c_1\rho_1(p_i - x_i(t)) \\ &+ c_2\rho_2(p_g - x_i(t)) \end{aligned} \quad (5)$$

3 Power System and Real-Time Simulation

The test system in this paper is a standard symmetrical two-area power system as illustrated in Figure 1 [7]. The system consists of four 20 kV, 900 MVA generator's connected by two parallel 230 kV transmission lines. Four PSSs are present in the power system, PSS_1 controls generator G1, PSS_2 controls generator G2, PSS_3 controls generator G3 and PSS_4 controls generator G4. To simulate the two-area power system in real-time, a RTDS is used. The RTDS is a fully digital power system simulator capable of continuous real-time operation, displayed in Figure 2. The real-time simulation provided by the RTDS allows controllers to be tested in hardware-in-loop configurations [7].

4 Digital PSS Implementation

The optimal parameters of two PSSs (PSS_1 and PSS_3) are obtained by PSO. The control block for the PSS is shown in Figure 3. It can be noticed that the control signal is limited in [-0.1, 0.1] to avoid instability in the power

Authorized licensed use limited to: University of Missouri. Downloaded on February 2, 2009 at 11:54 from IEEE Xplore. Restrictions apply.

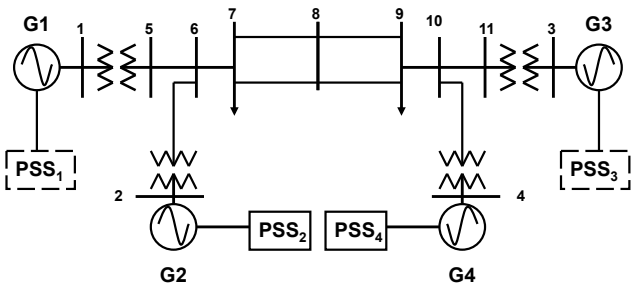

Figure 1. Two-area power system with a PSO-tuned PSSs on generators G1 and G3.

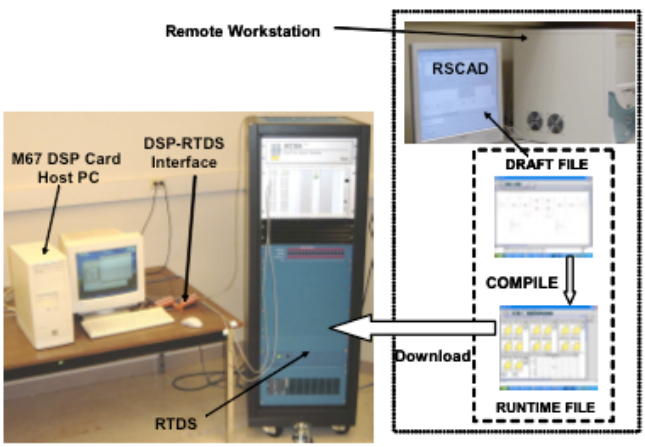

Figure 2. Laboratory hardware setup with RTDS and DSP.

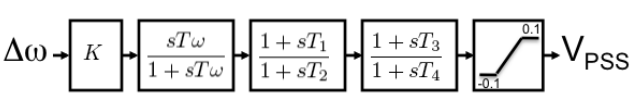

Figure 3. PSS controller block diagram.

system. The transfer function of a PSS in the continuous time domain is given by:

$$\frac{V_{PSS}(s)}{\Delta w(s)} = K\left(\frac{sT\omega}{1+sT\omega}\right)\left(\frac{1+sT_1}{1+sT_2}\right)\left(\frac{1+sT_3}{1+sT_4}\right) \tag{6}$$

For digital implementation, the transfer function in the Z-domain is as follows:

$$\frac{V_{PSS}(z)}{\Delta\omega(z)} = K'\frac{1+a_1z^{-1}+a_2z^{-2}+a_3z^{-3}}{1+b_1z^{-1}+b_2z^{-2}+b_3z^{-3}} \tag{7}$$

where K', a_1, a_2, a_3, b_1, b_2 and b_3 are all functions of the gain and time constants of (6).

When written in the difference equation form, the expression of the output signal $V_{PSS}(k)$ (at the kth sample instance) becomes:

$$\begin{aligned} V_{PSS}(k) = A_1V_{PSS}(k-1) + A_2V_{PSS}(k-2) \\ +A_3V_{PSS}(k-3) + A_4\omega(k) + A_5\omega(k-1) \\ +A_6\omega(k-2) + A_7\omega(k-3) \end{aligned} \tag{8}$$

where, A_1, A_2, A_3, A_4, A_5, A_6 and A_7 are the functions of K', a_1, a_2, a_3, b_1, b_2 and b_3. So tuning the digital PSS means tuning the coefficients A_1, A_2, A_3, A_4, A_5, A_6 and A_7.

5 DSP Implementation of PSO and PSSs

The PSO algorithm and the two PSSs are implemented on an Innovative Integration M67 which uses the Texas Instruments TMS3206701 DSP. The M67 operates at 160 MHz and is equipped with two A/D conversion and D/A conversion modules. The analog signals provided to the M67 are the speed deviations of generators G1 and G3, $\Delta\omega_1$ and $\Delta\omega_3$ respectively, which come from the RTDS. Those are converted to digital signals through the A/D block of the DSP and then passed through the two digital PSSs implemented inside the M67. The signals to the DSP are sampled at 40 Hz.

Since there are two PSSs in this study, each having 7 coefficients (A_1 to A_7 in (8)) the total number of coefficients to be tuned is 14. In the PSO algorithm, the number of dimensions of each particle is therefore 14. As the objective is to minimize the speed oscillations of generators G1 and G3, each set of PSS coefficients is used under a fault condition which disturbs the power system and causes generator speed deviations.

For each set of PSS coefficients, the transient energy of generators G1 and G3 (in the post-fault operating condition) is calculated. This transient energy is considered to be the fitness function in this case since a good PSS has to dampen the speed deviations fast. Therefore, the fitness function is

Authorized licensed use limited to: University of Missouri. Downloaded on February 2, 2009 at 11:54 from IEEE Xplore. Restrictions apply.

the summed transient energy of generators G1 and G3 over a period of time after the fault occurs:

$$fitness = \sum_{k=0}^{n} T_s[\Delta\omega_1^2(k) + \Delta\omega_3^2(k)]k \qquad (9)$$

where T_s is the sample period (25 ms), $\Delta\omega_1$ is the speed deviation of generator G1 at sample k and $\Delta\omega_3$ is the speed deviation of generator G3 at sample k. The transient energy is multiplied by k to put emphasis on also reducing the settling time. In this study, the transient energy of the generators is summed over a period of 10 seconds (n = 400 samples).

In order to synchronize the action of the DSP implementation of PSO and the simulation in RTDS, a binary signal is sent to the DSP from the RTDS to indicate that a fault has occurred. During the evaluation of each particle, the PSO implementation loads the coefficients into the two PSSs and wait for this binary signal before it begins calculating the fitness. The flowchart for the PSO implementation is given in Figure 4.

6 Results

The system is tested under two operating points with different loads in area 1 and area 2. For the first operating point, area 1 has a local load of 967 MW and Area 2 has a local load of 1767 MW. The second operating point has a local load of 1100 MW in area 1 and 1600 MW local load in area 2. The first operating point was used during the execution of the PSO algorithm while the second operating point was only used for testing the PSSs coefficients found by PSO. For both operating points, generator G2 and G4 have the conventional manually tuned PSSs which are simulated in the RTDS hardware. Whereas, generators G1 and G3 have the PSO-tuned digital PSSs which are implemented on the DSP. In order to tune the PSSs, a 167 ms three phase to ground symmetrical fault is simulated at bus 8. The parameters selected for PSO are shown in Table 1, the role of these parameters are discussed in Section 2. PSO is run for a maximum of 50 iterations. The values of the coefficients, as obtained after tuning by PSO for the two PSSs, are given in Table 2.

Table 1. PSO parameters.

n	w	c_1	c_2	x_{min}	x_{max}	v_{min}	v_{max}
20	0.8	2.0	2.0	-100	100	-10	10

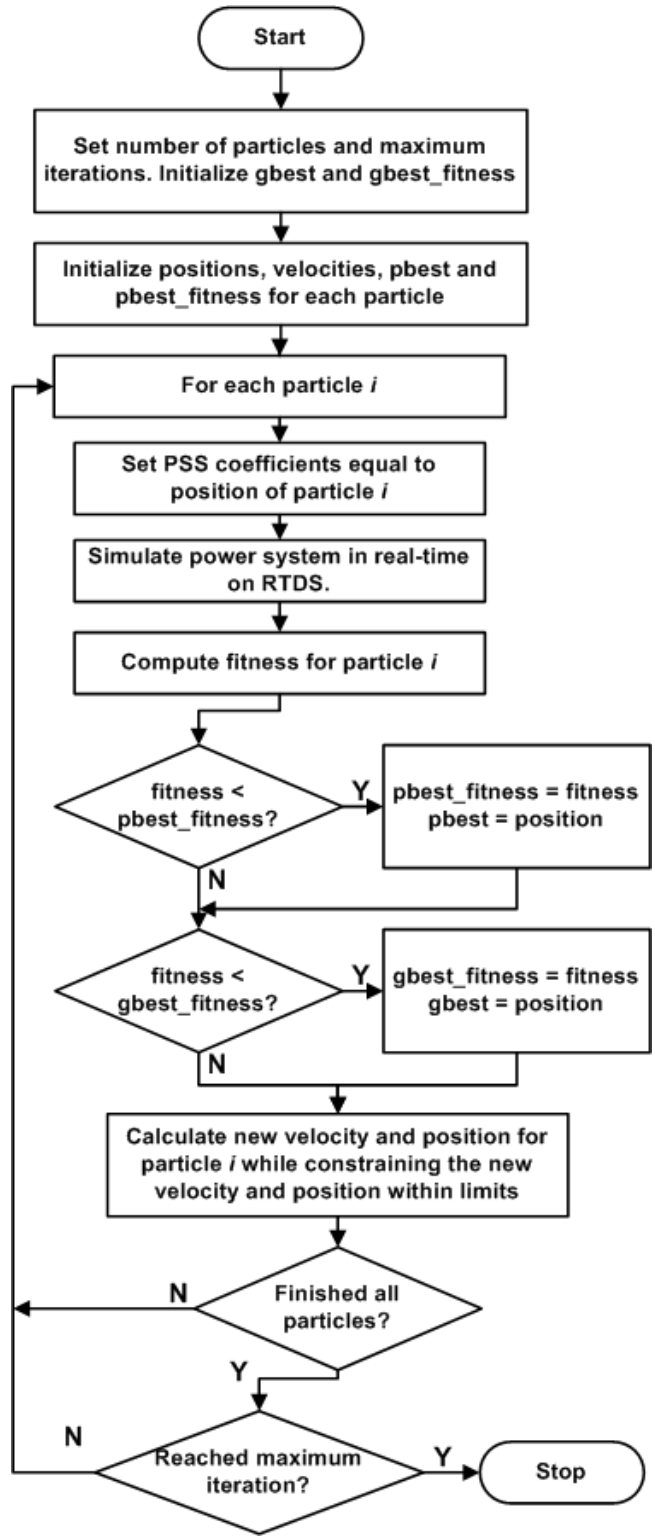

Figure 4. PSO flowchart.

Authorized licensed use limited to: University of Missouri. Downloaded on February 2, 2009 at 11:54 from IEEE Xplore. Restrictions apply.

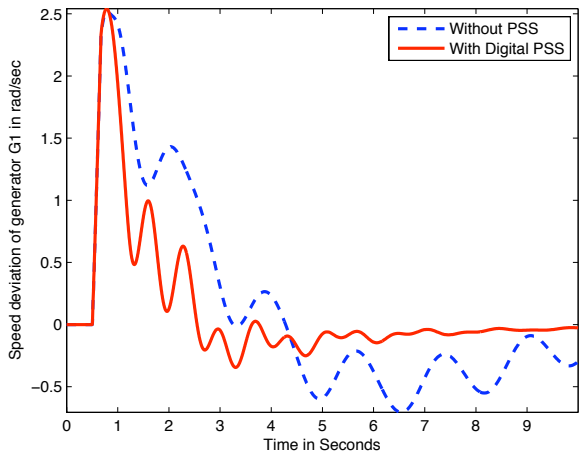

Figure 5. Speed deviation of generator G1 for a 167 ms short circuit fault at bus 8 under the first operating point.

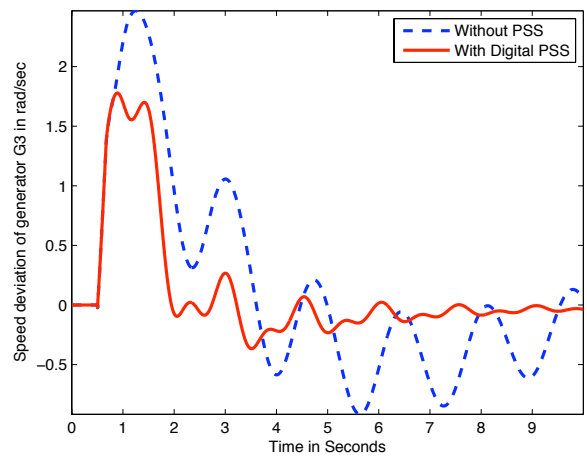

Figure 6. Speed deviation of generator G3 for a 167 ms short circuit fault at bus 8 under the first operating point.

Table 2. The PSS coefficients as tuned by PSO.

	PSS_1	PSS_3
A_1	0.0329	-0.1004
A_2	-0.1334	0.2344
A_3	0.0724	0.2945
A_4	0.4868	0.1879
A_5	0.4025	0.1135
A_6	-0.1139	-0.0143
A_7	-0.0348	0.0426

The coefficients found by PSO for PSS_1 and PSS_3 are then tested for each operating point . The speed deviations of generator G1 for the first operating point both with the PSO-tuned PSS and without, is given in Figure 5. The PSO-tuned PSS_1 is clearly able to quickly dampen the speed oscillations while the system continues oscillating without the PSS. This is the same case for generator G3, which is given in Figure 6. The PSSs are then tested under the second operating point. The speed deviations of G1 are given in Figure 7, with and without the PSSs. Under the second operating point it is also clear that the PSO-tuned PSS is able to dampen the speed oscillations, though the speed deviations of generator G3 had longer standing oscillations than generator G1, given in Figure 8.

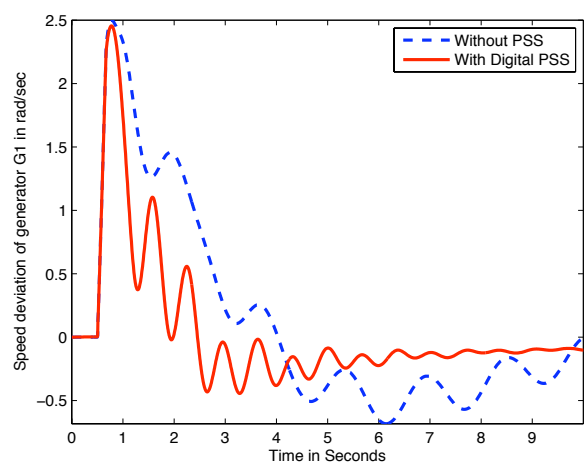

Figure 7. Speed deviation of generator G1 for a 167 ms short circuit fault at bus 8 under the second operating point.

Authorized licensed use limited to: University of Missouri. Downloaded on February 2, 2009 at 11:54 from IEEE Xplore. Restrictions apply.

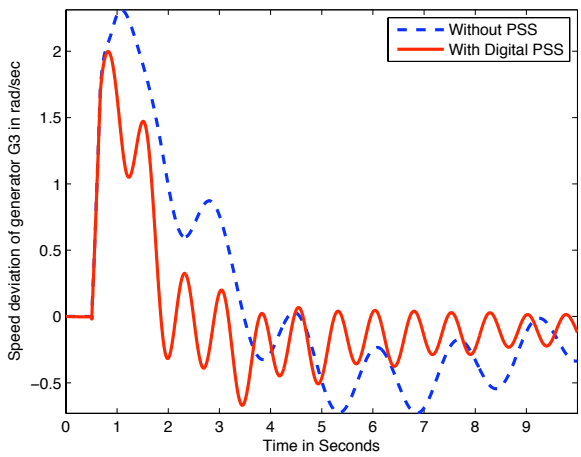

Figure 8. Speed deviation of generator G3 for a 167 ms short circuit fault at bus 8 under the second operating point.

7 Conclusion

A DSP implementation of PSO for PSS controller tuning has been presented. This results in an elegant and compact self-tuning process as the two PSSs and the PSO algorithm are implemented on a single DSP. The two PSO-tuned PSSs are shown to provide damping of power system oscillations. Utilizing the high processing speed of the DSP and the simplicity of PSO, the PSSs are successfully implemented in a real-time environment for tuning and testing. Classical optimal controllers can only be designed if the parameters of the system are known, while iterative tuning methods such as PSO can be applied to any system even if the system parameters are unknown.

While the PSSs are tuned in real-time, a simulator was used to simulate the power-system. Because of the nature of power systems, online tuning of the PSS controllers is not possible. However, a DSP implementation of PSO for control could be adapted for online tuning if the system being controlled does not need to maintain optimal performance during tuning. Otherwise extra caution taken during the tuning process can make such an approach useful.

References

[1] M. Abido. Optimal design of power-system stabilizers using particle swarm optimization. *Energy Conversion, IEEE Transactions on*, 17(3):406–413, Sept. 2002.

[2] V. Aggarwal, M. Mao, and U.-M. O'Reilly. A self-tuning analog proportional-integral-derivative (pid) controller. In *Proc. First NASA/ESA Conference on Adaptive Hardware and Systems AHS 2006*, pages 12–19, 15–18 June 2006.

[3] Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 81–86, Seoul, May 2001.

[4] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948, Perth, WA, Nov./Dec. 1995.

[5] R. A. Krohling. Gaussian swarm: a novel particle swarm optimization algorithm. In *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, volume 1, pages 372–376, Dec. 2004.

[6] M. Murakawa, E. Takahashi, T. Susa, and T. Higuchi. Post-fabrication clock timing adjustment for digital lsis with genetic algorithms ensuring timing margins. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3670–3674, 10–13 Oct. 2004.

[7] S. Ray and G. Venayagamoorthy. Real-time implementation of a measurement-based adaptive wide-area control system considering communication delays. *IET Generation, Transmission & Distribution*, 2(1):62–70, January 2008.

[8] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73, Anchorage, AK, May 1998.

[9] Y. Shi and R. C. Eberhart. Empirical study of particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, Washington, DC, July 1999.

[10] G. Ueno, K. Yasuda, and N. Iwasaki. Robust adaptive particle swarm optimization. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 4, pages 3915–3920, Oct. 2005.

Authorized licensed use limited to: University of Missouri. Downloaded on February 2, 2009 at 11:54 from IEEE Xplore. Restrictions apply.