Parallel Genetic Algorithms Applied to Damping Controllers Tuning on a Linux Cluster of PCs

CARMEN L. T. BORGES ENRIQUE C. VIVEROS GLAUCO N. TARANTO
Department of Electrical Engineering
Federal University of Rio de Janeiro
PO Box 68516, 21945 – 970
BRAZIL
carmen@dee.ufrj.br

Abstract: - The coordinated tuning of power system stabilizers (PSS) consists of an optimization problem where the objective function is to maximize system damping. Since conventional optimization methods tend to obtain a local optimum instead of a global one, the application of Genetic Algorithms (GA) to this problem has been considered. However, the computational effort required by the GA approach is very high and may become prohibitive for large-scale systems. This paper presents the implementation of two categories of Parallel Genetic Algorithms (PGA) applied to the PSS tuning problem, namely: master-slave and multi-population. In the master-slave PGA, the evaluation of the many chromosomes within a sole population is performed in parallel, whereas in the multi-population PGA, many populations are evaluated concurrently on different processors. Different communication topologies, migration rates and substitution strategies have been exploited in the multi-population PGA in order to evaluate the parallel speedup. The parallel platform utilized is a Linux cluster of PCs composed of 24 microcomputers interconnected by a switched Fast-Ethernet network. The results obtained show high speedup and good parallel efficiency on actual power system models.

Key-Words: - Power System Stabilizers (PSS), Parallel Genetic Algorithms (PGA), Cluster Computing, Master-Slave PGA, Multi-Population PGA.

1 Introduction

An Electrical Power System (EPS) is frequently submitted to non-predicted disturbances that modify its operation condition. Since the EPS components do not have linear behavior, its final state is highly dependent on the initial conditions prior to the disturbance and the characteristics of the disturbance itself. In the case of small disturbances, like variation in load demand, linear models may approximately predict the EPS behavior.

Power System Stabilizers (PSS) are controllers designed to enhance the so-called small-signal stability of EPS, by damping electromechanical oscillations. There is a myriad of methods for PSS tuning, especially for the case when one PSS at a time is tuned for one operating condition. However, one hardly finds procedures for coordinated tuning of PSS when various operating conditions are to be taken into consideration during the tuning process.

The problem of coordinated tuning of PSS can be setup in the form of an optimization problem, whose objective is to maximize the minimum system damping for the various operating conditions taken into consideration. The design of multiple power system controllers requires that several specifications be accomplished in order to ensure the system operation with adequate margins for a certain number of operating conditions. The large number of controlled devices in modern power systems, associated with the increased utilization of existing equipment, calls for a more rigorous and systematic decentralized control design procedure.

The efficiency of Genetic Algorithms (GA) when applied to the coordinated PSS tuning problem is demonstrated and detailed in [1]. However, GA may require high computation effort when searching for optimal solution for large-scale EPS. One way to speedup the computation requirements is to use parallel processing. This paper presents several parallel GA implementations on a cluster of PCs. The implementations exploit the impact of communication topologies and migration strategies on the performance of the parallel GA.

2 Problem Formulation

Power system damping controllers are usually designed to operate on a decentralized way. Input signals from remote sites are considered not reliable enough and avoided. Considering the performance of the control system for several different operating conditions ensures robustness of the controllers. Tuning of power system damping controllers

typically uses a small-signal model represented by the well-known state space equations [2]:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t) \quad \textbf{(1)}$$

where x is the vector of the state variables, such as the machine speeds, machine angles and fluxes; u is the vector of the input variables, such as control signals; y is the vector of the measured variables, such as bus voltage and machine speed; A is the power system state matrix; B is the input matrix; C is the output matrix; and D is the feedforward matrix. In the frequency domain, the transfer function associated with **(1)** is given by:

$$P(s) = C(sI - A)^{-1}B + D \quad \textbf{(2)}$$

where the poles of $P(s)$ correspond to the controllable and observable eigenvalues of matrix A. Let $P_k(s)$, $k = 1, 2, \cdots, m$, in **Figure 1**, represent the set Ω of selected power system operating conditions and $PSS(s)$ be a diagonal transfer function matrix with p individual controllers. The decentralized control design requires a control law $V_{PSS} = PSS(s)\,\omega$ such that the closed-loop system is stable and, if possible, has a minimum desirable damping ratio $\zeta_{\min}$ in all m operating conditions.

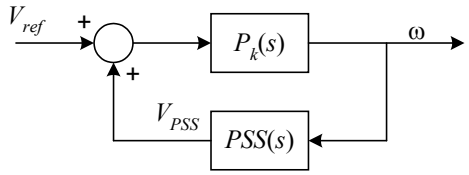

Figure 1: Closed-Loop Setup

For each one of the p controllers, it is assumed a classical control structure with the dynamic model consisting of a constant gain, a washout stage, and a l^{th} lead-lag stage as follows:

$$PSS(s) = K_i \frac{(T_w s)}{(1 + T_w s)} \times \left(\frac{1 + \frac{\sqrt{\alpha_i}}{\omega_i} s}{1 + \frac{1}{\omega_i \sqrt{\alpha_i}} s} \right)^l$$
$$i = 1, 2, \cdots, p \quad \textbf{(3)}$$

From the viewpoint of a washout function, the precise value of the associated time constant T_w is not critical. The main consideration is that it should be small enough such that stabilizing signals at the frequencies of interest will be relatively unaffected. For this reason, T_w is considered known. Therefore, K_i, α_i and ω_i, $i = 1, 2, \cdots, p$, are the parameters that should be determined by the tuning procedure. From **Figure 1** it is obtained the closed-loop state matrix (A_{cl}, where $A_{cl} \in \Re^{n+3p}$), which represents the linearized system including the tuned PSS, for each of the m system operating point. The optimization process objective is to obtain the PSS parameters that maximize the minimum damping value (ζ_j) for all system operating points, as shown in **(4)**.

$$MaxF = \sum_{i=1}^{m} \left[\sum_{j=1}^{n+3p} (\zeta_j) \right] \quad \textbf{(4)}$$

3 Genetic Algorithm Application

The search and optimization techniques based on Genetic Algorithms simulate the natural process of evolution of species, where the best fitted individuals have higher probability of propagating their genes along successive generations via cross-over between their genes [3]. The search mechanisms of GA are simple but powerful for complex function optimization. At the beginning of the search there is a highly random population with great diversity and low average fitness. Successive application of genetic operators conduce the search towards optimal solution, first exploring all search space and them concentrating on the neighborhood of good solutions.

The main genetic operators are the Selection, the Cross-over and the Mutation Operators. The Selection Operator is applied over each generation in order to choose the best fitted chromosomes. The Cross-over Operator combines two selected chromosomes to generate other two to be included in a new generation. The Mutation Operator produces random changes on the chromosomes in order to maintain the diversity of the population.

In this work it has been implemented a classical GA, where each individual is represented by a data structure composed of a chromosome and a fitness function value. The chromosome is composed by the PSS parameters to be adjusted while the fitness function is related to the system damping calculated for all considered operating conditions. It has been used the real codification for the chromosome, where the variables are represented by their numerical real values. The size of the chromosome is equal to three times the number of PSS to be tuned.

The Selection Operator is the so-called Tournament, with two chromosomes. In this type of selection, two

chromosomes are randomly chosen from the population and the one with higher fitness is selected to breed new chromosomes.

At the beginning of the optimization process, N chromosomes are randomly generated and the diagonal $PSS(s)$ transfer matrix is calculated. Given $PSS(s)$, the closed-loop system is formed as shown in **Figure 1**. The system damping coefficients are obtained from the closed-loop system eigenvalues, which are calculated by the QR method [4]. So, N QR solutions must be calculated in each generation for each operating condition m. The fitness value for each chromosome is based on the minimum damping obtained from the closed-loop system considering all operation points, as shown below:

Fitness	Damping Range
0.1	If any $(\zeta_j)_i \le \zeta_0$
β_0	If all $(\zeta_j)_i > \zeta_0$ and If any $(\zeta_j)_i \le \zeta_1$
:	:
β_k	If all $(\zeta_j)_i > \zeta_{k-1}$ and If any $(\zeta_j)_i \le \zeta_{min}$
F	If all $(\zeta_j)_i \ge \zeta_{min}$

$$0.1 < \beta_0 < \beta_1 < \beta_2 < \cdots < \beta_k < F$$

$$\zeta_0 < \zeta_1 < \zeta_2 < \cdots < \zeta_{k-1} < \zeta_{min}$$

where β_0, β_1, ..., β_k and ζ_0, ζ_1, ..., ζ_{min} are positive real pre-defined numbers.

Figure 2 shows the scheme of PSS tuning by the GA.

4 Parallel Solution

The sequential GA (SGA) described in the previous section has been applied to the coordinated tuning of PSS problem and turned out to be a promising method for the robust coordinated tuning of PSS [1]. However, the SGA requires very long computational time to obtain good solutions, what may become prohibitive for large-scale EPS. One way to reduce the solution process time requirement is to use parallel processing, in particular Parallel Genetic Algorithms (PGA). Two different strategies for parallelizing the GA have been tested: Master-Slave PGA and Multi-population PGA.

4.1 Master-Slave PGA

Most of the processing time required for the solution of the problem described is spent by the closed-loop system eigenvalues calculation by the QR method. Therefore, the most natural solution to reduce the computational time required by the GA is to parallelize the fitness function evaluation in a master-slave model [5], here called PGA_MS.

In PGA_MS, the initial population is generated with N chromosomes. Then, the population is divided into P groups of chromosomes, where P indicates the number of available processors. Each one of those groups is formed by nc chromosomes ($nc = N/P$), so it is convenient to chose N as a multiple of P. Then, P-1 groups will be transmitted to the slave processors in order to evaluate the fitness of the nc chromosomes. The master also evaluates the fitness of the chromosomes assigned to itself, in this case the first group of chromosomes.

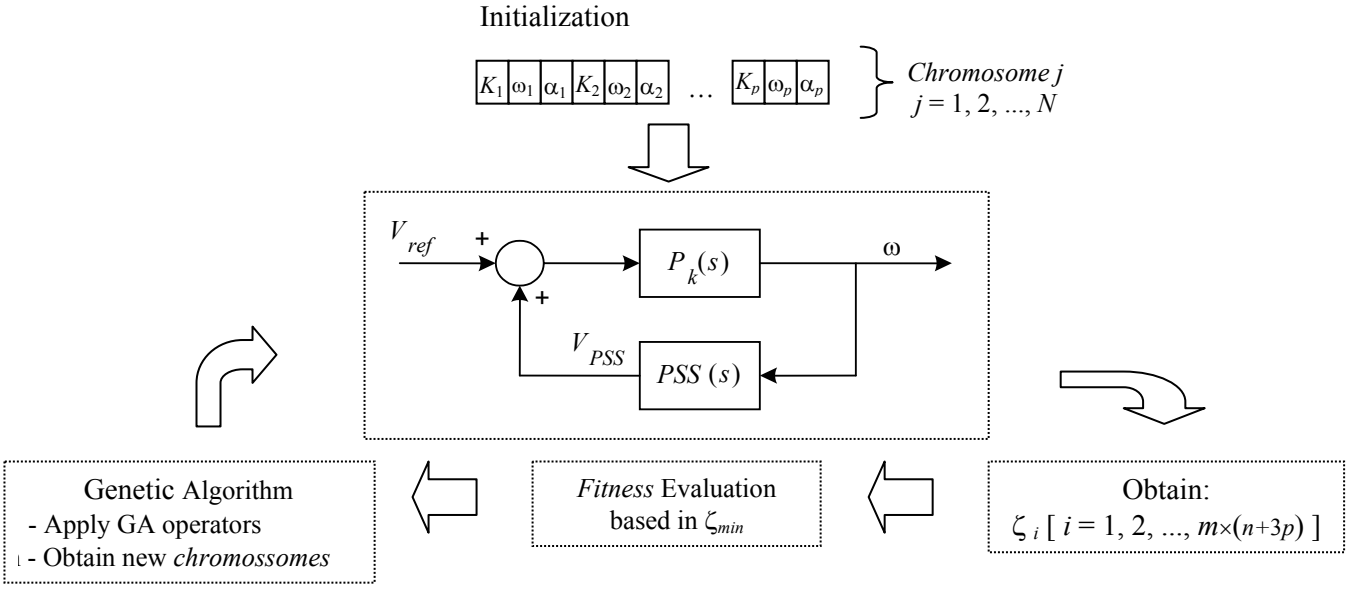

Figure 2: The GA Role

After each slave has concluded its computation, they send to the master the results of the fitness evaluation they have performed. The master then applies the genetic operators to the chromosomes of the population to obtain a new generation based on the fitness values calculated in parallel. This process continues until the maximum number of generations is reached. In PGA_MS and in SGA, the genetic operators are applied to the same population, what means that the same optimal solution is obtained sequentially and in parallel if the random number generator algorithm uses the same seed.

4.2 Multi-Population PGA

In the multi-population PGA implementation, different populations are allocated to different processors in parallel. The initial populations are created by randomly generating their chromosomes using different seeds for the random number generator algorithm in each processor. The seed calculated for each population is obtained by the following equation:

$$seed_{OP} = \frac{seed_M}{OP + 1} \quad \textbf{(5)}$$

In **(5)**, $seed_M$ means the seed supplied by the user for processor 0 and $seed_{OP}$ is the value of the seed calculated in the other processors as a function of the processor rank (OP = 1, 2, ... , P – 1). The size of the population on each processor is equal to the total number of individuals N divided by the number of processors P.

In the multi-population PGA, after a fixed number of generations, the populations interchange a certain number of individuals in a process called Migration [6]. The migration operator establishes the communication and integration strategy among the populations. The new parameters introduced by the migration operator are [7]:

- *Migration interval* – It establishes the number of generations between successive migrations.
- *Migration rate* – It indicates the number of individuals to be communicated to the other populations at each migration.
- *Individuals' Selection Strategy* – It establishes the rules for choosing the individuals that will migrate to other populations. In the present work, the individuals that have higher fitness are chosen to migrate from one population to the others.
- *Reception Strategy* – It establishes the rules for incorporating the individuals migrated from other populations into the population that receives them.

In the migration process, if the individuals are transmitted from one processor to all the others, it is said that the connectivity is dense, otherwise it is said that the connectivity is sparse.

The communication cost in Multi-Population PGA with dense connectivity may be very high since each processor communicates with all other available processors. This means that the time spent in communication tends to increase proportionally with the number of processors. For that reason a Multi-Population PGA with sparse connectivity has been implemented in this work, where each processor maintains communication with only two other processors, despite of the number of available processors. Thus, the time spent in communication remains constant even if the number of processors is increased.

The multi-population PGA algorithm has been implemented using synchronous communication, as well as the master-slave PGA. The reasons for applying synchronous communication are that each processor executes the same algorithm, the number of chromosomes assigned to each processor is the same and finally, the characteristics of the processors that compose the cluster are the same, as will be seen in the next section.

5 Experimental Results

5.1 Cluster of PCs

The high-performance cluster used as parallel platform is a dedicated PC Cluster composed of 24 Intel Pentium III 500 MHz personal computers with 512 MBytes of memory each, interconnected by a switched Fast-Ethernet network of 100 Mbits per second bandwith. The cluster is based on Linux operating system and uses MPICH implementation of MPI as message passing system [8]. All implementations were based in C programming language.

5.2 Power System Model

The power system model used to test the algorithms was the New England system described in [9], which contains 39 buses and 10 generators. In this system there are 9 PSS to be simultaneously tuned for 10 different operation conditions varying over the topology of the system, demanded load level, etc.

5.3 GA Parameters

Each chromosome is composed of 27 genes, since there are 9 PSS in the system with 3 unknown parameters each. The dimension of the open-loop state matrix A related to equation **(1)** is 48, which is equal to the number of state variables. The closed-loop state matrix related to **Figure 1** has 75 states.

For each algorithm, a global population of 120 individuals was considered (N = 120). With the purpose of increasing the convergence speed, 4 individuals that represent stable solutions for the system are included in the initial population and the rest of the population is generated randomly.

The same probabilities of applying the genetic operators (Cross-over and Mutation), as well as the same types of Selection, Cross-Over and Mutation, were used in the sequential and in the parallel implementations. The parameters used in the GA are:

- Coding Type: Real
- Selection Type: Tournament (2 Individuals)
- Cross-Over Type: One point cross-over
- Cross-Over Probability: 0.8
- Mutation Type: Constant
- Mutation Probability: 0.1

For the Multi-population PGA, it has been chosen a migration interval of 10 generations, following the suggestions made in [6,7]. The performance of this algorithm was also tested for two migration rates (1 and 3 individuals). The reception strategy used were: *Substitution Strategy*, in which the best individual received from other population substitutes the worst individual of the population than receives it, and E*xtended Population Strategy*, in which the individuals received from other populations are added to the population that receives them, forming an extended population for application of the Selection operator, but then maintaining the original population size throughout the generations.

5.4 Results Analysis

Table 1 shows the experimental results for the sequential GA (SGA) and the master-slave PGA (PGA_MS), where t_{mean} indicates the computation time calculated by the mean of 3 different runs, S_P represents the speedup calculated for p processors and E(%) represents the respective parallel efficiency. The maximum fitness value of f_{max} = 591.56 has been obtained to all these trials, meaning that a minimum of more than 15% damping is achieved in all 10 operating conditions with the PSS tuned by the algorithm.

Table 1: Results for SGA and PGA_MS

Sequential (SGA)			
P	t_{mean}		
1	18min80s		
Master-Slave PGA (PGA_MS)			
P	t_{mean}	S_P	E
8	2min25s	7.82	97.75 %
15	1min17s	14.55	97.00 %
24	49s	23.07	96.12 %

It can be observed from **Table 1** that a considerable reduction in computational time is achieved when applied the PGA_MS algorithm. It can also be observed that the PGA_MS algorithm is very efficient, with an efficiency higher than 96% on 24 processors of the cluster.

Table 2 shows the experimental results for the multi-population PGA algorithm using dense connectivity (PGA_MD) and Extended Population reception strategy. It can be observed from **Table 2** that implementations with larger migration rate are slightly less efficient. The reason is that larger migration rate increases the communication time due to the larger number of individuals transmitted between the populations, making the total computational time increase. This influence tends to be more significant when larger clusters are used.

Table 2: Results for PGA_MD

Migration Rate = 1			
P	t_{mean}	S_P	E
8	2min22s	7.98	99.75%
15	1min19s	14.30	95.33%
24	49.83s	22.69	94.57%
Migration Rate = 3			
8	2min23s	7.90	98.75%
15	1min20s	14.26	95.07%
24	50.13s	22.56	94.00%

The fitness values obtained by the multi-population PGA (f_{max} = 597.50) were higher than those obtained by SGA and PGA_MS, which represents an improvement in the optimal solution. Moreover, implementations with larger migration rate produced solutions with higher fitness values. The reason is the increase in the diversity of the search process caused by a larger number of good individuals received from other populations. However, this higher fitness value corresponds to the same minimum damping of 15 %.

In terms of efficiency, PGA_MD achieved more than 94% on 24 processors, what can also be considered a very good result. The implemented algorithms also presented good scalability and promising performance is expected in a cluster with even higher number of processors. Another test that is being done in continuation to this work is to use a larger actual power system model and verify the algorithm performance.

The speedup obtained by the parallel algorithms is almost linear, what means that the total processing time decreases in an almost linear proportion to the number of processors. **Figures 3** shows the speedup curve for parallel algorithms PGA_MS and PGA_MD on the cluster.

It was also analyzed the influence of the reception strategy and the connectivity on 8 processors of the cluster. The results are shown on **Table 3** for the multi-population PGA algorithm with connectivity 2 (PGA_M2) and migration rate = 3.

Table 3: Results for PGA_M2

Reception Strategy = Substitution			
P	t_{mean}	f_{max}	E
8	2min22s	593.10	99.20%
Reception Strategy = Extended Population			
8	2min23s	598.40	99.00%

It can be observed from **Table 3** that the fitness value is higher when multi-population PGA uses the extended population as reception strategy. The reason is that the selection operator is applied on the extended population, which contains its own good individuals and the good individuals received from the other populations. It can also be observed that a more sparse connectivity tends to improve the algorithm performance. However, this effect is not significant on 8 processors and is being confirmed on all 24 processors of the cluster. It should be mentioned that the efficiency was calculated using the processing time (sequential and parallel) in seconds and in the tables t_{mean} has been approximated to minutes.

6 Conclusion

Sequential GA has demonstrated to be an adequate method for coordinated tuning of power systems stabilizers. However, its computational time may become prohibitive for application in large power system models.

Master-slave parallel GA reduces the computational time considerably, and calculates exactly the same fitness value obtained by SGA. Multi-population parallel GA, on the other side, not only reduces the computational time but also has higher probability of finding solutions with better fitness values.

For larger and more complex power systems, the multi-population PGA algorithm that uses substitution as the reception strategy tends to be faster due to the minimum number of data interchanged and individuals involved in the cross-over process. However, multi-population PGA that used extended population as the reception strategy was the one that found highest fitness value. This is due to the increase in population diversity, what guarantees a better search and optimization performance.

Cluster computing has demonstrated to be an economical and adequate parallel environment for this kind of application. The high speedup and efficiency obtained by all implemented algorithms on 24 processors point to promising parallel performance for larger power system models on a cluster with even larger number of nodes.

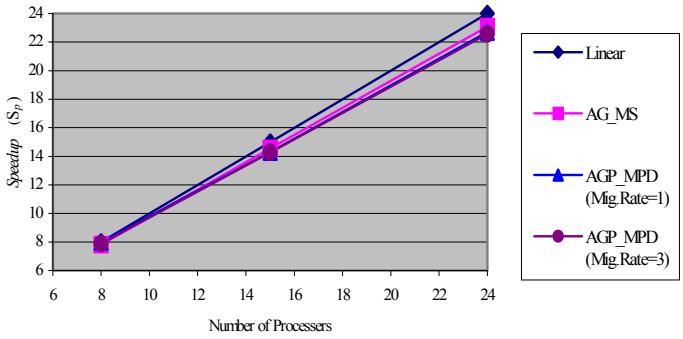

Figure 3: Speedup Curves

References:

[1] A. L. B. do Bomfim, G. N. Taranto and D. M. Falcão, "Simultaneous Tuning of Power System Damping Controllers using Genetic Algorithms". *IEEE Transactions on Power Systems*, Vol. 15, No. 1, pp. 163 – 169, February 2000.

[2] P. Kundur, *Power System Stability and Control*. Electric Power Research Institute series – EPRI series. Mc Graw Hill, Inc., 1994.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison – Wesley, MA, 1989.

[4] W. H. Press, S. A. Teukolsly, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C*: *The Art of Scientific Computing*.

[5] C.L.T. Borges, D. M. Falcão, G. N. Taranto, "Cluster Based Power System Analysis Applications". *Proceedings of IEEE International Conference on Cluster Computing - Cluster 2000*, pp.193 – 200, Chemnitz – Germany, 2000.

[6] M. O. Mejía, E. Cantú – Paz, "DGENESIS: Software para la Ejecución de Algoritmos Genéticos Distribuidos". *XX Conferência Latinoamericana de Informática (CLEI – PANEL'94)*. México, 1994.

[7] E. Cantú – Paz, "Topologies, Migration Rates, and Multipopulation Parallel Genetic Algorithms". *Illinois Genetic Algorithms Laboratory – IlliGAL Report No. 99006.* University of Illinois. Urbana – Champaign, United States. January, 1999.

[8] W. Gropp, E. Leusk and A. Skjellum, *Using MPI – Portable Parallel Programming with the Message Passing Interface.* The MIT Press, Cambridge – Massachusetts, UK, 1996.

[9] R. T. Byerly, D. E. Sherman and R. J. Bernnon, "Frequency domain Analysis of Low frequency oscillations in large electric systems". *Report EPRI EL – 726*, 1978.