Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 376291, 19 pages
doi:10.1155/2012/376291

Research Article

A Parallel Implementation of Unscheduled Flow Control in Interconnected Power Systems

G. Ozdemir Dag[1] and Mustafa Bagriyanik[2]

[1] *Department of Computational Science and Engineering, Informatics Institute, Istanbul Technical University, Maslak, 34469 Istanbul, Turkey*
[2] *Department of Electrical Engineering, Electrical & Electronics Faculty, Istanbul Technical University, Maslak, 34469 Istanbul, Turkey*

Correspondence should be addressed to Mustafa Bagriyanik, bagriy@itu.edu.tr

Received 27 November 2011; Revised 31 January 2012; Accepted 31 January 2012

Academic Editor: Jyh Horng Chou

The unscheduled power flow problem needs to be minimized or controlled as soon as possible in a deregulated power system since the transmission systems are mostly operated at their power-carrying limits or very close to it. The time spent for simulations to determine the current states of all the system and control variables of the interconnected power system is important. Taking necessary action in case of any failure of equipment or any other occurrence of an undesired situation could be critical. Using supercomputing facilities and parallel computing techniques together decreases the computation time greatly. In this study, a parallel implementation of a multiobjective optimization approach based on both genetic algorithms and fuzzy decision making to manage unscheduled flows is presented. Parallel computation techniques are applied using supercomputers (high-performance computers). The proposed method is applied to the IEEE 300 bus test system. Two different cases for some parameters of GA are considered to see the power of parallel computation technique. Then the simulation results are presented.

1. Introduction

After the liberalization of the electricity markets, the operation of the power system has fundamentally changed. How much power flows on which transmission lines, which company uses the other's transmission lines, and/or the amount and duration of the transmission line usage have all become very important issues.

The path electrical energy takes depends on physical laws. That is, Kirchhoff's current and resistance laws determine the path for the energy to flow. Energy takes the shortest path (in terms of resistivity) instead of a contracted path. Several power flow cases could exist in a transmission network depending on the system topology and operating conditions. When

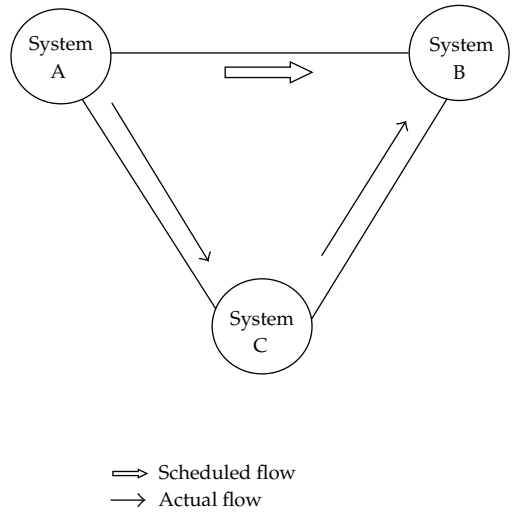

Figure 1: The loop flow.

the actual power flow exceeds system-operating limit for a transmission path, the transmission system operators must immediately mitigate the transmission overloads to reduce the actual power flow across the path. The deviation of actual electric power flows in transmission circuits from the scheduled (expected) power flows is called unscheduled flows, and these flows may result in the congestion problems and increase the active and reactive power transmission losses in the lines depending on the power to be transmitted. This can be best explained using Figure 1. System B buys power from system A; however, power travels through system C occupying different transmission lines. One needs to control (i.e., redirect the power to the transmission line directly connecting A and B) or prevent power flows between A and C. The high power losses, low efficiencies, or the path power travels through (occupying transmission lines) before arriving to the loads has not seen as an important problem in a government-controlled power system [1–3].

Under the competitive market conditions, the power flow control in an interconnected power system is now seen as a serious problem, since these flows affect the operation of the electric power system by introducing transmission bottlenecks, congesting flow paths, increasing the transmission losses, decreasing reliability, causing blackouts, and so forth. When an interconnected power transmission system is operated at or close to its power-carrying limits, the transmission congestion restricts additional power to be transferred on that line. In the deregulated power systems, Transmission System Operators (TSOs) have to provide an adequate set of paths from each buyer (seller) to multiple sellers (buyers). A transmission-pricing scheme can be used in order to achieve an efficiently operating market. Pricing unscheduled flows of electricity may also contribute to the reliability of the network. This action may also increase both competition and transmission capacity [4]. The prevention and/or control of unscheduled flows become important due to the need of prevention or reduction of fines associated with the unscheduled flows. The management of these flows requires remedial actions such as switching the lines for new topology and setting transformers and generators for new operating conditions.

There have been a number of studies in the literature that attempt to solve the problem. The operating problems caused by unscheduled flows and the complexities of these problems

were highlighted in [2]. The paper showed that a stable power flow might circulate in a ring AC power system. Suryanarayanan et al. [3] proposed an approach based on Lp-norms to estimate the unscheduled flows occurring in a wide area-interconnected system. They used contribution factors to assess the participation of generating utilities among causes of unscheduled flow. In [5], a modification to the formula for determining the contribution of generation companies to unscheduled flows in electric power networks was presented. Applications of contribution factor and game theoretic approach for managing unscheduled flows are described in [6]. The effects of parallel paths in system network topology and a survey to explore the present state of practices used to determine the transfer capability issues are well studied in [7]. The authors in [8] proposed an algorithm based on the security constrained optimal power flow for the coordinated control of several phase shifters by one operator with the objective of reducing the unscheduled flow through its system. A method based on power system state estimation was proposed in [9]. The authors presented a monitoring tool to track unscheduled flows due to power transactions not reported to the control area operator. In [10], a genetic algorithm-based procedure was designed for the topological optimization of the network against unscheduled flows. The method considered the status of substation breakers and the location (and angle) of phase-shifting transformers as the control variables.

There have been quite a few applications of fuzzy set theory to various power system problems [10, 11]. With the advent of computational power and tools, researchers tried new methods such as genetic algorithms and evolutionary algorithms on power system problems as well [12]. We have proposed a fuzzy-set-theory-based genetic algorithm for the solution of the unscheduled power flow [13]. Since the power systems include thousands of buses resulting in large dimensional matrix operations, the use of parallel environments and techniques will shorten the solutions time. This has enabled us to solve the problem in a multicore environment. Parallel and distributed computing has been applied to speed up certain calculations in power system operation and control. The state of the art on parallel processing applications to power systems was presented in [14]. The authors in [15] presented an efficient parallel GA (EPGA) for the solution of large-scale OPF with practical generators constraints and with consideration shunt FACTS devices.

We proposed a fuzzy-set-theory-based parallel genetic algorithm procedure to control and/or prevent unscheduled flows. The problem is formulated as a multiobjective problem subject to operational and electrical constraints and it is handled in fuzzy environment since in practice small variations of power systems variables (bus voltages, line currents, etc.) from their limit values can be tolerated. When the classical methods are used to control loop flow in a rigid manner, a feasible solution may not be found. However, using the proposed method, one can find a feasible solution that maximizes the satisfaction degree. Once the problem is formulated, we solve it using parallel genetic algorithms. The methods using serial GA have restricted applicability in a practical environment because they have a number of problems such as high time complexity. Since the power systems include thousands of buses resulting in large dimensional matrix operations needing the use of iterative solutions methods, the proposed method generates high-quality solutions in a short time. The solution of the problem in a multicore environment also enables us to consider more paths and loops. In the next section, the proposed approach is described. In Sections 3–5, the basic principles of genetic algorithms, parallel computation, and parallel genetic algorithms are introduced. In Section 6, the simulation results are given and discussed. Finally conclusions are provided.

2. Proposed Approach

In general, a classical multiobjective optimization problem can be formularized combining multiple objectives into a single-objective function by using some user-defined parameters as follow:

Minimize

$$F = w_1 \cdot f_1(x,u) + w_2 \cdot f_2(x,u) + \cdots + w_n \cdot f_n(x,u) \tag{2.1}$$

Such that

$$\begin{aligned} g(x,u) &= 0, \\ h(x,u) &\le 0, \end{aligned} \tag{2.2}$$

where x represents system variables, and u represents control variables. The equality constraints are $g(x,u) = 0$, and the inequality constraints are $h(x,u) \le 0$.

In the proposed multi-objective optimization model, the minimization of unscheduled line flows is the main objective. The method also tries to keep both active power losses and reactive power losses as small as possible. The constraints on transmission line flows (MVA limits), voltage magnitudes of load buses, and the settings of control variables such as transformer taps, VAR compensators, and generator voltages are inequality constraints. The equality constraints are the real and reactive power balances of the power system. The problem is formulated as follows.

2.1. Objectives

2.1.1. The Minimization of Unscheduled Line Flows

The power flow from bus i to bus j is

$$P_{ij} + jQ_{ij} = \dot{V}_i \dot{I}_{ij}^* = V_i \left(V_i^* - V_j^* \right) y_{ij}^* + V_i^2 y_{i0}^*, \tag{2.3}$$

where P_{ij} is the active power fed into the line from bus i and Q_{ij} is the reactive power fed into the line from bus j; $\dot{V}_i$ is the phasor voltage of ith bus; y_{ij} is the admittance of the series elements; y_{i0} is the half of the line charging admittance.

The proposed method minimizes unscheduled real power flows through the contracted paths. This objective is handled in fuzzy environment by using designated fuzzy membership functions.

2.1.2. The Minimization of Power Losses

The active power loss in transmission lines can be calculated as (Figure 2)

$$P_L = \sum_{k=1}^{nl} G_k \left(V_i^2 + V_j^2 - 2V_i V_j \cos\left(\delta_i - \delta_j\right) \right). \tag{2.4}$$

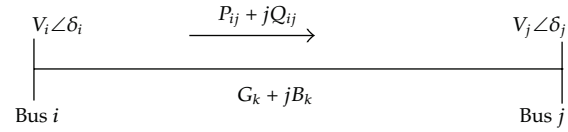

Figure 2: Transmission line.

The reactive power loss in transmission lines can be calculated as

$$Q_L = \sum_{k=1}^{nl} B_k \left(V_i^2 + V_j^2 - 2V_i V_j \cos(\delta_i - \delta_j) \right), \tag{2.5}$$

where G_k is the conductance of the kth line that connects bus i to bus j; B_k is the susceptance of the kth line that connects bus i to bus j; V_i is the voltage magnitude of ith bus; δ_i is the phase angle of ith bus; nl is the total number of lines.

Both objectives, the minimization of real power losses and the minimization of reactive power losses, are handled in fuzzy environment by using fuzzy membership functions.

2.2. Constraints

2.2.1. Equality Constraints

The total generated power should be equal to total load demand plus the total losses in the system. The equality constraints are active and reactive power-balanced equations for each generator and load bus. These equations are also known as power flow equations. The equality constraints are nonlinear equations that can be solved by the Newton-Raphson method called the power flow program.

The active power injected to ith bus in a power system is

$$P_i = \sum_{j=1}^{n} V_i V_j Y_{ij} \cos(-\theta_{ij} - \delta_j + \delta_i), \quad i = 1, 2, \ldots, n, \tag{2.6}$$

and the reactive power injected to ith bus in a power system is

$$Q_i = \sum_{j=1}^{n} V_i V_j Y_{ij} \sin(-\theta_{ij} - \delta_j + \delta_i), \quad i = 1, 2, \ldots, n, \tag{2.7}$$

where n is the total number of buses, and $\dot{Y}_{ij} = Y_{ij}\angle\theta_{ij}$ is the transfer admittance between bus i and bus j.

2.2.2. Inequality Constraints

The inequality constraints are limits on physical devices in the power system as well as the limits created to ensure system security. These constraints except for control variables limits

are fuzzified by using membership functions. The inequality constraints on control variables were considered as crisp constraint during the solution by GA. These crisp constraints are

$$\begin{aligned} V_{g,i}^{\min} &\le V_{g,i} \le V_{g,i}^{\max}, \\ Q_{g,i}^{\min} &\le Q_{g,i} \le Q_{g,i}^{\max}, \\ Q_{C,i}^{\min} &\le Q_{C,i} \le Q_{C,i}^{\max}, \\ P_{g,i}^{\min} &\le P_{g,i} \le P_{g,i}^{\max}, \\ t_i^{\min} &\le t_i \le t_i^{\max}, \end{aligned} \tag{2.8}$$

where $V_{g,i}$ is the generator bus voltage of ith bus, $Q_{g,i}$ is the generator reactive power generation of ith bus, $Q_{C,i}$ is the shunt capacitor reactive power generation of ith bus, $P_{g,i}$ is the generator active power generation of ith bus, and t_i is the transformer tap setting of ith bus.

The power transfer capacity of a transmission line is limited by thermal rating of the line (MVA rating). To maintain system security, bus voltages have maximum and minimum magnitudes. These limits are handled in fuzzy environment by using fuzzy membership functions.

A fuzzy-set-theory-based genetic algorithm procedure was proposed to control and/or prevent unscheduled flows [13]. In fuzzification step, the objectives and constraints were fuzzified by using designated membership functions, where these functions should be properly constructed to guarantee the overall performance. The fuzzy membership functions define the degrees of membership that a crisp value has in a fuzzy set. The membership functions scale both the objectives and the constraints to the unit interval $[0, 1]$, so that the satisfactory of the objectives and the constraints can be compared without being influenced by their original values. The closer the membership is to one, the better the solution is for that objective or constraint. In fuzzy decision making, the optimal solution is defined to be the one with the highest degree of membership, and thus the optimization problem becomes that of maximizing the satisfaction with the solution, subject to the crisp and fuzzy constraints [16, 17]. In our approach, the objective functions were minimization of the unscheduled line flows on the contracted paths and minimization of both total active losses and total reactive losses. The constraints (voltages remaining within the limits, line flows remaining within the limits, etc.) were also handled in fuzzy environment by using designated fuzzy membership functions. Since piecewise linear membership functions such as trapezoidal or triangular functions were used in many literatures [11, 18] because of their simplicity and efficiency with respect to computability, they were considered for the objectives and constraints except for the case of the unscheduled flows.

The exponential membership function described by four parameters (a, b, c, d) with four breakpoints of the shape, given in Figure 3, has been selected for the fuzzification of the unscheduled flows since an earlier study showed that more satisfying results can be obtained using this kind of membership function [19]. The membership function $\mu_{L,ij}(P_{ij})$

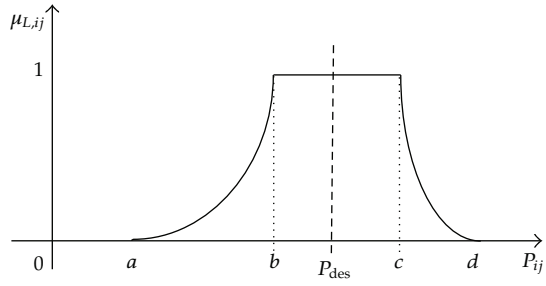

Figure 3: The fuzzy memberships function for the loop MW flows (the exponential form).

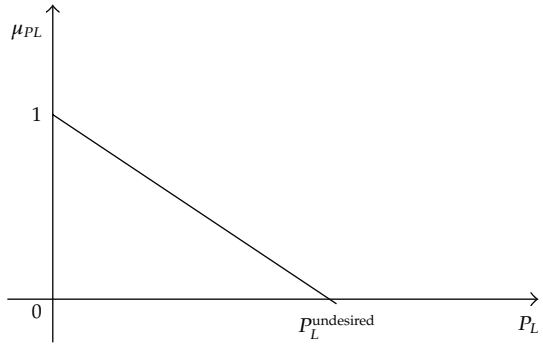

Figure 4: The fuzzy memberships function for the active power losses.

which belongs to the MW flow (line flow) through the line between buses i and bus j is defined as

$$\mu_{L,ij}(P_{ij}) = \begin{cases} \dfrac{1}{e^{(b-a)}}\left(e^{P_{ij-a}} - 1\right) & \text{if } a_{ij} < P_{ij} < b_{ij}, \\ 1 & \text{if } b_{ij} < P_{ij} < c_{ij}, \\ e^{(c-P_{ij})(d-P_{ij})} & \text{if } c_{ij} < P_{ij} < d_{ij}, \\ 0 & \text{if otherwise,} \end{cases} \tag{2.9}$$

where $a_{ij} < b_{ij} < c_{ij} < d_{ij}$ must hold.

A membership function $\mu_{PL}(P_L)$ which is considered for the active power transmission losses is shown in Figure 4, where $P_L^{\text{undesired}}$ represents the unsatisfaction limit for the total active power transmission losses. The degree of satisfaction for the objective is zero for any value of P_L greater than $P_L^{\text{undesired}}$. We want to keep the objective below unsatisfaction limit. The system operators taking into account both their experiences and system operation costs can determine these satisfaction limits on the membership function. The membership functions considered for reactive power transmission losses $\mu_{QL}(Q_L)$ are similar to $\mu_{PL}(P_L)$.

The constraints on transmission line flows (MVA limits) are fuzzified using memberships function, which is shown in Figure 5. Here, the degree of satisfaction for the MVA flow through the line between bus i and bus j is zero for any value of the MVA flow greater than the line's current carrying capacity limit (thermal limit) ($\text{MVA}_{ij}^{\text{unsatisfied}}$). The system operators,

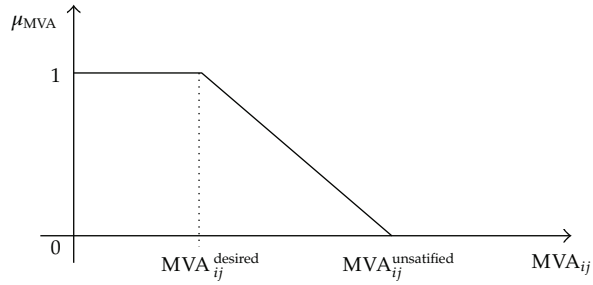

Figure 5: The fuzzy memberships function for MVA flows.

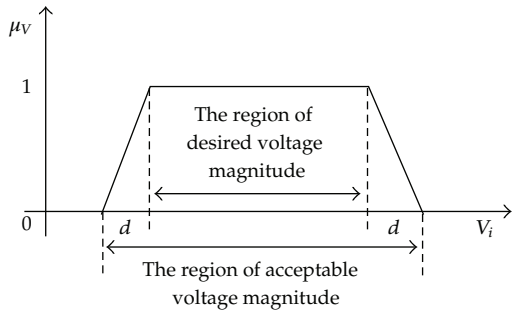

Figure 6: Fuzzy memberships function for bus voltage magnitudes.

who can take into account the feasible operating conditions, can determine the desired MVA flow limit of a specific line $\text{MVA}_{ij}^{\text{desired}}$.

The trapezoidal membership functions shown in Figure 6 were used to fuzzify the constraints on bus voltage magnitudes. These membership functions are defined according to the voltage violation of buses to respond to the practical situations as much as possible. In power systems operation, the unsatisfaction tolerance, d, is set to be the allowable maximum bus voltage violation [18].

3. Genetic Algorithms

Genetic algorithms (GAs) are nature inspired stochastic and population-based search and optimization methods. Their mechanism is based on natural selection and natural genetics. The evaluation function plays the role of the environment that rates solutions according to their fitness, that is, how good they are in solving the search problem for a population [20].

GAs work very well on both continuous and discrete combinatorial problems. There are many ways to modify the basic algorithm and many parameters according to a specific problem [21]. They are less likely to get stuck at local optima than gradient search methods.

There are three important stages to apply genetic algorithms. They are the following:

(1) definition of the objective function,

(2) definition and implementation of the genetic representation,

(3) implementation of the genetic operators (selection, crossover, mutation).

Once these three have been defined, the basic genetic algorithm should work well. To improve performance, many different variations of the operators can be tested [22].

Outline and the main operators of the basic GA [23] are as follows.

(1) [Start] Generate random population P of n chromosomes (population, suitable solutions for the problem).

(2) [Fitness] Evaluate the fitness $f(x)$ of each chromosome x in the population.

(3) [New population] Create a new population by repeating following steps until the new population is complete.

  (i) [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).

  (ii) [Crossover] With a crossover probability, cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

  (iii) [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome).

  (iv) [Accepting] Place new offspring in a new population.

(4) [Replace] Use new generated population for a further run of algorithm.

(5) [Test] If the end condition is satisfied, stop, and return the best solution in current population.

(6) [Loop] Go to step 2.

In fuzzy decision making, the optimal solution, which is the fuzzy decision μ_D, is given as an intersection of the fuzzy sets describing the constraints $(\mu_c(y))$ and the objectives $(\mu_g(y))$:

$$\mu_D = \min \left\lfloor \mu_g(y),\ \mu_c(y) \right\rfloor. \tag{3.1}$$

The fuzzy decision μ_D represents the worst satisfaction value or worst degree of fuzzy membership belongs to either system state variable, constraints, or objectives of the problem. Overall purpose is to maximize the worst degree of membership:

$$\max \mu_D. \tag{3.2}$$

Evaluation function which is called fitness needs to be defined. The fitness function for our problem is defined as follows:

$$\text{fitness} = \frac{1}{1 + \mu_D}. \tag{3.3}$$

In GA formulation, the objective function is the minimization of fitness function:

$$\min \text{fitness}. \tag{3.4}$$

The control variables such as transformer tap settings, generation bus voltage magnitudes, and generator reactive power generations with their upper and lower limits are used to create chromosomes in GA. An example is given as follows:

$$x = \left[t_i, V_j, P_k\right] \quad \{i = 1,2,3,\ldots,l,\ j = 1,2,3,\ldots,ng,\ k = 1,2,3,\ldots,m\}, \tag{3.5}$$

where l is the number of tap changing transformers, ng is the number of generator buses, m is the number of the generators in the system, and x is a possible solution set to the problem.

Population consists of many sets of solution. The number of solution sets is defined as population size. Within GA, a load flow analysis is performed for each population. The results of load flow analysis (bus voltages, line flows, losses, etc.) are passed to fuzzy decision making process, where a membership value is assigned for all constraints and objectives. The minimum of those membership values is then tried to be maximized by GA. The process continues up to the end condition defined. One way of defining an end condition is setting a maximum iteration number.

4. Parallel Computation

The main disadvantage of GAs is the high CPU time execution, and this can cause heavy computation demands in a large-scale power system case, resulting in large dimensional matrix operations. One way to overcome these computation demands is to use parallel algorithms in multiprocessor computers or in distributed environments.

Parallel computation is evolution of serial computation [24]. In general sense problems are partitioned and solved concurrently. This can be in many ways, some of which are instruction level partitioning, data partitioning, task partitioning, and so forth. To solve a computational problem, parallel computing simply involves the simultaneous use of many computing resources. Computing resources come in the form of either a single computer with multiple processors or multiple computers (computing nodes) connected by a fast network.

One of the main purposes of using parallel computing is to solve a problem faster or solve a big problem that is otherwise unsolvable in an acceptable time. Parallel computing historically intended to model difficult scientific and engineering problems that exist in the real world [25].

In most cases, there are both parallelizable and unparallelizable parts in the problem formulations. The speedup of a program using multiple processors in parallel computing is limited by the time needed for the serial part of the program. There are also several factors that limit the speedup such as communication time for sending messages. The scheduling of tasks on processors and communication between them consume extra CPU resources. Amdahl's Law says that speedup of a program is limited by the serial part of the computation time of the program [26]:

$$\text{speed} = \frac{1}{(P/N) + S}, \tag{4.1}$$

where N represents the number of processor, P represents the parallelization rate of the program, and S represents the serial computation rate of the program. Theoretically, if N goes to

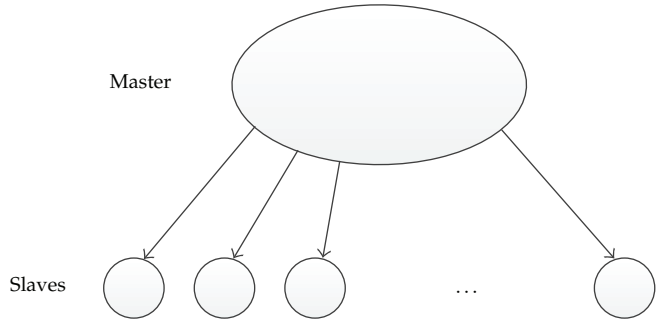

Figure 7: Master-slave parallel GA.

an infinitely large number of processor, computational time spent for the parallel part of the program goes to zero. Then the maximum speed will be

$$\text{speed}_{N\to\infty} = \frac{1}{S}. \tag{4.2}$$

5. Parallel Genetic Algorithms

GAs are very powerful search and optimization techniques that can be applied to problems in many different areas such as sciences, business, and engineering [27]. GAs simulate the evolution of a population of individuals, which are possible solutions of a specific problem, through the evolution principle. They can produce an acceptable solution to a problem in a reasonable time. However when they are applied to very hard and large problems, it takes quit a long time. In that case, there might be some effort to reduce the time into a reasonable range. One way of reducing execution time is to implement GAs in parallel environment. These topics are well discussed in literature [25, 27–31].

There are some important questions to be answered when GAs are parallelized. These are the following.

(i) How fitness computation and mutation would be done?

(ii) How would selection operator be applied, local or global?

(iii) Would there be a single or multiple populations?

(iv) If multiple populations were used, how would individuals be exchanged?

There are four major types of parallel GAs, namely, master-slave parallelization, coarse-grained parallel GA, fine-grained parallel GA, and hybrid parallel GA [30, 31]. The first type, which is one of the more popular types in practice, was used in this study. If function evaluation is expensive, master-slave implementation (global parallel GA) becomes more efficient. This type uses a single population. This is usually implemented using master-slave programs, where the master stores the population and applies the GA operators which are selection, crossover, and mutation. The slaves evaluate the fitness for the assigned individuals to them. Exploration of the search space is in the same way of serial GA. That is why it is called as global parallel GA. It is also known as master-slave model or distributed fitness evaluation. Master and slave processes are presented in Figure 7.

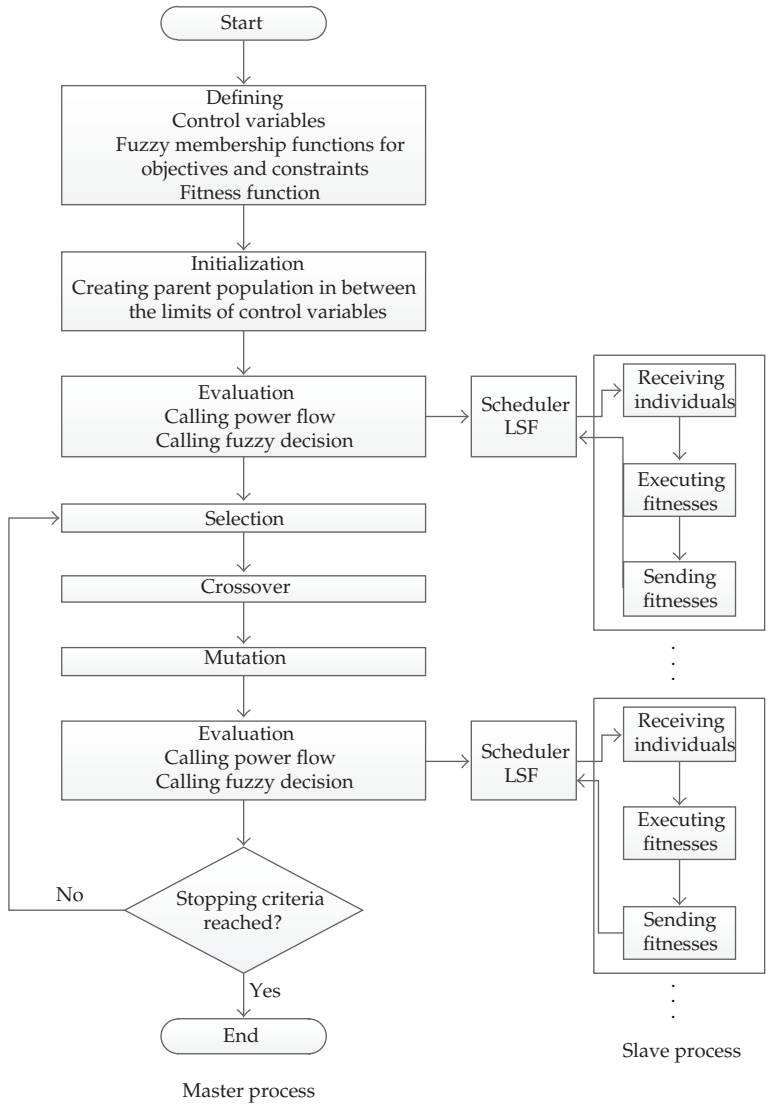

Figure 8: Flow chart of the proposed procedure.

The execution time consists of two parts. One is computation time for evaluation of the fitness of a part of the population assigned to each individual slave processor. The other is the communication time in between master and slaves. The master-slave parallel GA is particularly easy to implement. The flow chart of the proposed procedure is shown in Figure 8. Master and slaves are also presented in this figure.

6. Simulation Results

The simulations are performed using the utilities provided by the high-performance computing lab in Computational Science and Engineering Department in Istanbul Technical University. The computing system specifications are given in Table 1.

Table 1: The computing system specifications.

Computing system	Distributed cluster, with 37 nodes, each consisting of 2 CPU's having 3.40 GHz CPU, and 2 GB memory
Network type	Myrinet-2000
Operating system	RHEL 4.4
Resource management and scheduling software	LSF (load sharing facility)
Library	MPICH-GM Myrinet 1.2.7

Table 2: The contracted paths and their scheduled flows.

	Path 1	Path 2	Path 3
Line number	44	48	117
From bus	3	7	62
To bus	150	131	144
Scheduled flow [MW]	100	60	100
(a, b, c, d)	$(85, 95, 105, 115)$	$(40, 55, 65, 80)$	$(90, 95, 105, 110)$

The algorithm was implemented in MATLAB environment using toolboxes such as Genetic Algorithm Toolbox, Parallel Computing Toolbox, and Distributed Computing Server, and MATPOWER which is an open source power flow program was also used [32, 33]. Master-slave parallelization method was chosen among the types of parallel GAs. When a problem demands higher computation time, it fits better [34].

The method is applied to IEEE 300-bus test system having 69 generators and 411 lines. The detailed data of the system can be found in literature. Three lines given in Table 2 are considered as the contracted paths, and the scheduled flows for the lines are also given in the table. In the simulations, $d = 0.05$ (see Figure 6) is taken in fuzzification of the voltage magnitudes. $P_L^{\text{unsatified}}$ and $Q_L^{\text{unsatified}}$ are taken as 1.8 times of active and reactive power losses of base case, respectively. The ranges are taken large so that they would not be the dominative ones among the other objectives and constraints. Generator reactive limits are also taken into account. The parameter values used in GA solution are given in Table 3. The initial population is taken from the base case power flow solution of the system. Simulations are performed for two different cases. In the first case, population size and iteration number are taken as 100. In the second case, both population size and iteration number are taken as 150.

When evaluating a parallel system, it is important to know how much performance gain is achieved by parallelizing a given application over a serial implementation [34]. In the first case, the rate of parallelization of our method was figured out as 78% after a series of simulations. We applied Amdahl's law to obtain the theoretical maximum speedup. In this case, P is 0.78, S is 0.22, and from (2.7), the theoretical maximum speedup can be calculated as 4.5455. In other words, the maximum acceleration from the distributed computing for our program will be 4.5455 times that of 100% serial simulation.

When a distributed computing system is used, the speed of program can be calculated by using the following equation:

$$\text{speed} = \frac{T_1}{T_N}, \tag{6.1}$$

Table 3: The parameter values used in GA.

	Case 1	Case 2
The population size	100	150
Iteration number	100	150
The population type	Real valued (double vector),	
The selection type	tournament (size is 4),	
chromosome length	231	
The elite number	2	
The crossover type	Two point crossovers	
The crossover rate	0.8	
The mutation function	Adapt-feasible	

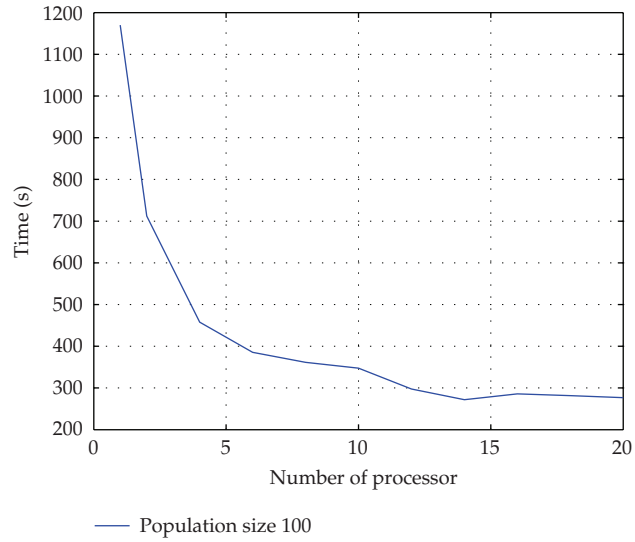

Figure 9: Execution time with different numbers of processors for Case 1.

where N is the number of processors, T_1 is the execution time for running the program completely in serial manner, and T_N is the execution time for running the program by using the N processors.

Efficiency is a measure of the fraction of time for which a processing element is usefully employed; it is defined as the ratio of speedup to the number of processing elements. In an ideal parallel system, speedup is equal to N and efficiency is equal to one. In practice, speedup is less than N and efficiency is between zero and one, depending on the effectiveness with which the processing elements are utilized [34]. The efficiency of the system used for our program can be calculated from (6.2):

$$\text{efficiency} = \left(\frac{\text{speed}}{N}\right)100\%. \tag{6.2}$$

As seen in Figure 9, there is no improvement in terms of the execution time after the usage of fourteen processors even though we continue increasing the number of processor.

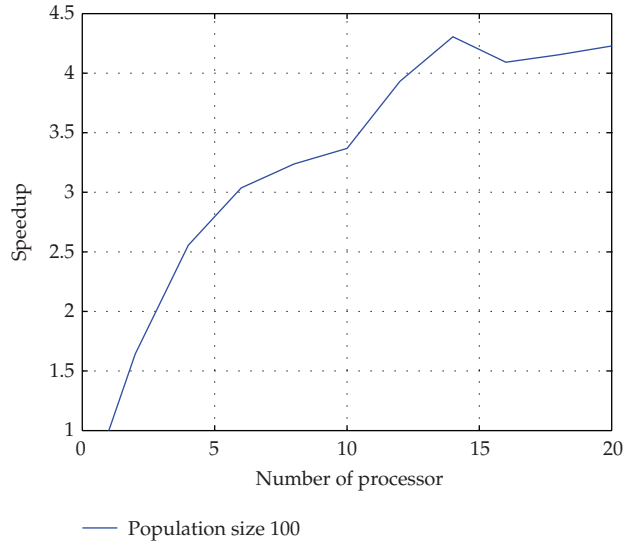

Figure 10: Speedup with different numbers of processors for Case 1.

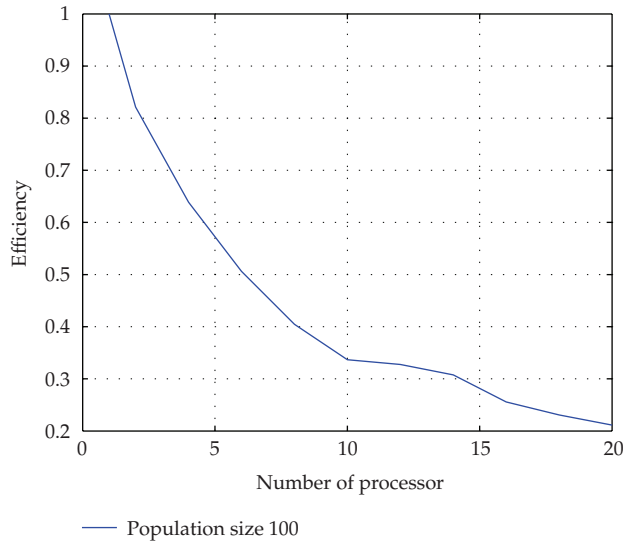

Figure 11: Efficiency with different numbers of processors for Case 1.

In Figure 10, we see the change of speed versus processor number. From this figure, it is easy to find that the speed can be increased greatly by using parallel implementation. But the implementation achieved low efficiency when the numbers of processor were increased.

Figure 12 shows the execution time for Case 2. The measured times were wall-clock time (elapsed time). As seen from Figures 9 and 12, computation times fluctuate. This was due to changes in the usage of the network at the moment of the program execution. As shown in Figure 13, reasonable speedup was obtained when the number of processors was increased. In Figure 14, the change of efficiency versus processor number can be seen. Even though we continue to use more processors, the efficiency of the program is not increasing

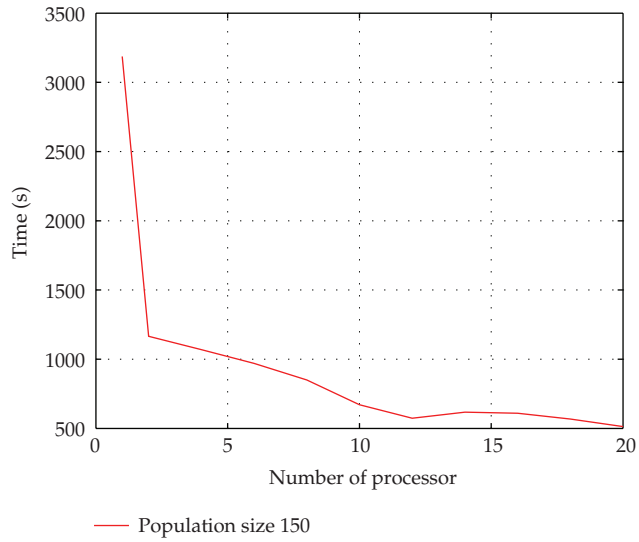

Figure 12: Execution time with different numbers of processors for Case 2.

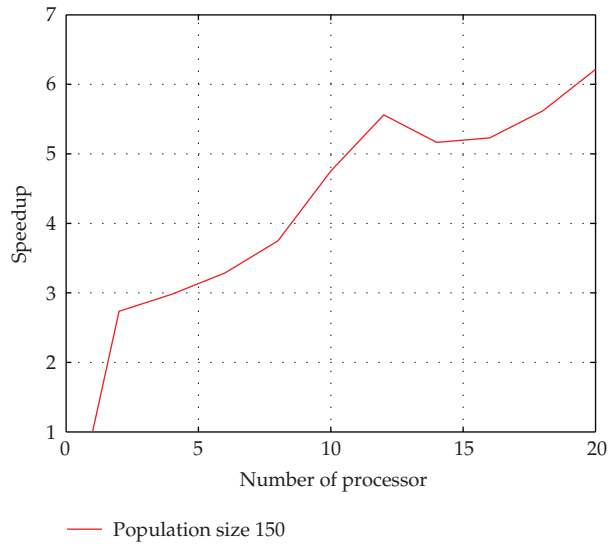

Figure 13: Speedup with different numbers of processors for Case 2.

due to not utilization of the full capacity of each processor. In the second case, the rate of parallelization of our program went up to 90%. In this case, the theoretical maximum speed obtained using Amdahl's law will be 10 times that of 100% of serial simulation. For this case the figures for time, speed, and efficiency are given in Figures 12, 13, and 14, respectively.

The system solutions for the objectives and the constraints are given in Tables 4 and 5 for the Case 1 and Tables 6 and 7 for the Case 2. When we compare the base case results to the simulation results, we see that our method is successful. As seen from Tables 4 and 6, we reached the targeted values in the given ranges. This is better understood from the averaged fuzzy membership values given in Tables 5 and 7.

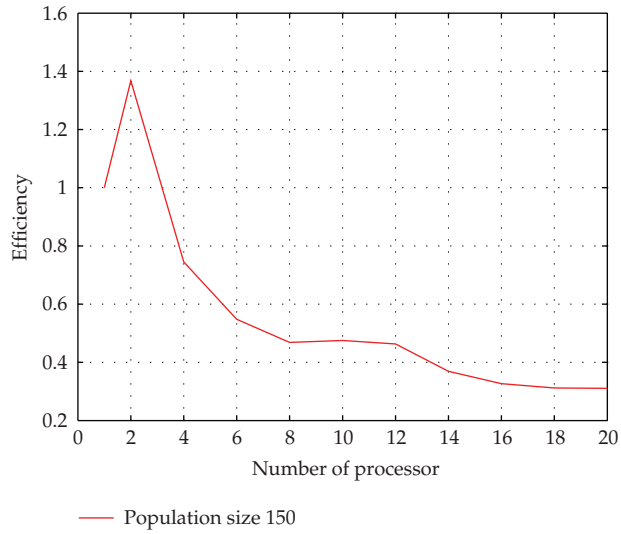

Figure 14: Efficiency with different numbers of processors for Case 2.

Table 4: Averaged values for objectives and constraints (population size is 100, iteration number is 100).

	Path 1 [MW]	Path 2 [MW]	Path 3 [MW]	P_{loss} [MW]	Q_{loss} [MVAR]
Base case	92.90	46.04	−101.16	408.316	5504.18
Target	100	60	−100		
Test results	101.1533	54.0599	−97.4213	412.2546	5526.4
Best fitness			0.7741		

Table 5: Averaged fuzzy membership values for objectives and constraints (population size is 100, iteration number is 100).

	min (μ)	$\mu_{\text{Path 1}}$	$\mu_{\text{Path 2}}$	$\mu_{\text{Path 3}}$	$\mu_{P\text{loss}}$	$\mu_{Q\text{loss}}$
Base case	0.0001	0.1163	0.0001	1.000	0.4451	0.4451
Test results	0.3750	0.8255	0.4382	0.8350	0.4391	0.4422

Table 6: Averaged values for objectives and constraints (population size is 150, iteration number is 150).

	Path 1 [MW]	Path 2 [MW]	Path 3 [MW]	P_{loss} [MW]	Q_{loss} [MVAR]
Base case	92.90	46.04	−101.16	408.316	5504.18
Target	100	60	−100		
Test results	103.3790	54.8188	−95.7833	405.3045	5492.9
Best fitness			0.7741		

As seen from Table 5, the smallest membership value for the base case is improved from the level of 0.0001 to 0.3750. The membership values for the lines where we try to control power flow are improved from 0.1163 to 0.8255 for the first path and from 0.0001 to 0.4382 for the second path. The membership value for the third path is decreased from 1.000 to 0.8350. Only decrease is occurred for the third path but this decrease is not significant when we consider other improvements.

Table 7: Averaged fuzzy membership values for objectives and constraints (population size is 150, iteration number is 150).

	$\min(\mu)$	$\mu_{\text{Path 1}}$	$\mu_{\text{Path 2}}$	$\mu_{\text{Path 3}}$	μ_{Path}	μ_{Qloss}
Base case	0.0001	0.1163	0.0001	1.000	0.4451	0.4451
Test results	0.3202	0.9792	0.6544	0.5632	0.4485	0.4456

There is no bad effect on the membership values of both active and reactive power losses. This situation was among the objectives of our problem. During the control of power flows, active and reactive power losses should not increase significantly.

Similar improvements are seen for the second case in which population size and iteration number are increased. These are seen from Tables 6 and 7.

The increment on iteration number and population size has an effect only on the total computation time. This is seen on Figures 9 and 12. The efficiency for the population size 100 is getting below 40% after utilizing eight processors as seen from Figure 11. The efficiency for the population size 150 is getting below 40% after utilizing fourteen processors as seen from Figure 14. The efficiency value below 40% is not recommended.

7. Conclusion

This paper presents a multiobjective optimization approach based on both genetic algorithms and fuzzy decision making to manage unscheduled flows. The remedial actions for prevention and/or control of unscheduled flows can be determined by the proposed method. Solving the optimization problem with serial genetic algorithm may be time-consuming when the number of system states is large, since power flow analysis is performed for each system state (each population). Using supercomputing facilities and parallel computing techniques together decreases the computation time greatly.

To speed up the computing process, parallel implementation of genetic algorithm is proposed in this study. As seen from the test results, when the population size is increased, the amount of computation job per processor is increased as well. In that case, each processor is utilized better. Some improvement on efficiency is also seen. The large portion of the computation time is spent for the function evolution in the problem. This is due to the nature of our problem. Master-slave PGA is perfectly matched with our needs.

We conclude that by using a larger system, we would gain more benefit from using a high-performance computing facility that allows parallel implementation of our program to get the system solution faster than the serial corresponding.

References

[1] IEEE Committee Report, "Operating problems with parallel flows," *IEEE Transactions on Power Systems*, vol. 6, no. 3, pp. 1024–1034, 1991.

[2] J. N. Janssens and A. Kamagate, "Loop flows in a ring AC power system," *International Journal of Electrical Power and Energy System*, vol. 25, no. 8, pp. 591–597, 2003.

[3] S. Suryanarayanan, R. G. Farmer, G. T. Heydt, and S. Chakka, "Estimation of unscheduled flows and contribution factors based on Lp norms," *IEEE Transactions on Power Systems*, vol. 19, no. 2, pp. 1245–1246, 2004.

[4] M. Lively, "Creating an automatic market for unscheduled electricity flows," *The National Regulatory Research Institute*, vol. 3, 2005.

[5] S. Suryanarayanan and G. T. Heydt, "Modification to contribution factor formula for unscheduled flows," *IEEE Transactions on Power Systems*, vol. 23, no. 2, pp. 809–810, 2008.

[6] S. Suryanarayanan, "Techniques for accommodating unscheduled flows in electricity networks and markets," in *Proceedings of the IEEE Power and Energy Society General Meeting*, pp. 1–6, July 2008.
[7] J. A. Kavicky and S. M. Shahidehpour, "Parallel path aspects of transmission modeling," *IEEE Transactions on Power Systems*, vol. 11, no. 3, pp. 1180–1190, 1996.
[8] A. Marinakis, M. Glavic, and T. Van Cutsem, "Minimal reduction of unscheduled flows for security restoration: application to phase shifter control," *IEEE Transactions on Power Systems*, vol. 25, no. 1, pp. 506–515, 2010.
[9] K. A. Clements, A. S. Costa, and A. Agudelo, "Identification of parallel flows in power networks through state estimation and hypothesis testing," *International Journal of Electrical Power and Energy Systems*, vol. 28, no. 2, pp. 93–101, 2006.
[10] G. Granelli, M. Montagna, F. Zanellini, P. Bresesti, and R. Vailati, "A genetic algorithm-based procedure to optimize system topology against parallel flows," *IEEE Transactions on Power Systems*, vol. 21, no. 1, pp. 333–340, 2006.
[11] J. A. Momoh and K. Tomsovic, "Overview and literature survey of fuzzy set theory in power systems," *IEEE Transactions on Power Systems*, vol. 10, no. 3, pp. 1676–1690, 1995.
[12] V. Miranda, D. Srinivasan, and L. M. Proença, "Evolutionary computation in power systems," in *Proceedings of the Power Systems Computation Conference (PSCC '96)*, pp. 25–40, Dresden, Germany, 1996.
[13] G. O. Dag and M. Bagriyanik, "Controlling unscheduled flows using fuzzy set theory and genetic algorithms," *International Review of Electrical Engineering*, vol. 5, no. 1, pp. 185–193, 2010.
[14] D. J. Tylavsky, "Parallel processing in power systems computation," *IEEE Transactions on Power Systems*, vol. 7, no. 2, pp. 629–638, 1992.
[15] B. Mahdad, K. Srairi, and T. Bouktir, "Optimal power flow for large-scale power system with shunt FACTS using efficient parallel GA," *International Journal of Electrical Power and Energy Systems*, vol. 32, no. 5, pp. 507–517, 2010.
[16] G. J. Klir and T. A. Folger, *Fuzzy Sets Uncertainty and Information*, Prentice-Hall International, 1988.
[17] H. J. Zimmermann, *Fuzzy Set Theory and its Applications*, Kluwer Academic, 2nd edition, 1993.
[18] K. Tomsovic, "A fuzzy linear programming approach to the reactive power/voltage control problem," *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 287–293, 1992.
[19] G. O. Dag and M. Bagriyanik, "The effect of different fuzzy membership function forms on controlling loop flows," in *Proceedings of the 44th International Universities Power Engineering Conference (UPEC '09)*, University of Strathclyde, 2009.
[20] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer-Natural Computing Series, 2003.
[21] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 1996.
[22] G. O. Dag and M. Bagriyanik, "The effects of selection strategies on fuzzy-genetic algorithm based power flow control," in *Proceedings of the International Conference on Genetic and Evolutionary Methods (GEM '09)*, Las Vegas, Nev, USA, 2009.
[23] January 2012, http://www.obitko.com/tutorials/genetic-algorithms/ga-basic-description.php.
[24] B. Barney, "Introduction to Parallel Computing," Lawrence Livermore National Laboratory, 2010, https://computing.llnl.gov/tutorials/parallel_comp/.
[25] R. Buyya, *High Performance Cluster Computing: Programming and Applications*, vol. 2, chapter 1, Prentice Hall, 1999.
[26] G. M. Amdahl, "Validity of the single-processor approach to achieving large-scale computing capabilities," in *Proceedings of the American Federation of Information Processing Societies (AFIPS '67)*, pp. 483–485, 1967.
[27] E. Cantu-Paz, *Designing Efficient and Accurate Parallel Genetic Algorithms*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1999.
[28] E. Cantu-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles*, vol. 10, no. 2, 1998.
[29] S, N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer, Berlin, Germany, 2008.
[30] J. Stender, *Parallel Genetic Algorithms: Theory and Applications*, IOS Press, 1993.
[31] E. Cantu-Paz and D. E. Goldberg, "Efficient parallel genetic algorithms: theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 221–238, 2000.
[32] MATLAB, *Relase 2009a*, vol. 14, The MathWorks Inc., Natick, Mass, USA, 2009.
[33] MATPOWER 3.2 A MATLAB Power System Simulation Package, 2007, http://www.pserc.cornell.edu/matpower/.
[34] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing*, Addison-Wesley, 2nd edition, 2003.