International Journal on
"Technical and Physical Problems of Engineering"
(IJTPE)
Published by International Organization of IOTPE

ISSN 2077-3528
IJTPE Journal
www.iotpe.com
ijtpe@iotpe.com

December 2013 | Issue 17 | Volume 5 | Number 4 | Pages 88-93

GENETIC ALGORITHM APPLICATION IN ECONOMIC LOAD DISTRIBUTION

N.M. Tabatabaei [1,2] A. Jafari [1] N.S. Boushehri [1,2] K. Dursun [3]

1. Electrical Engineering Department, Seraj Higher Education Institute, Tabriz, Iran
n.m.tabatabaei@gmail.com, ali.jafari.860@gmail.com, nargesboush@yahoo.com
2. Taba Elm International Institute, Tabriz, Iran
3. Electrical Engineering Department, Ostfold University College, Fredrikstad, Norway, kamil.dursun@hiof.no

Abstract- In this paper we try to present Genetic Algorithm and how to use it for solving Economic Dispatch problem. In the first section, we present on what basis does it work, and why Genetic Algorithm (GA) is useful and what problems can be solved by it. In the second section, we present theory of GA like representation of design variables, representation of objective function and constraints, genetic operators, and in third section we review the GA steps for solving problem, and in fourth section we present classic Economic Dispatch by Genetic Algorithm and how to use it for solving ED problems, like present of constraints and variables. In fifth section we use MATLAB programing tool for write a simple program for solving a simple problem of ED that written program presented in paper attachment. In sixth section we compare results of GA and Lagrangian method for compare GA performance. To end we have conclusion of this paper.

Keywords: Genetic Algorithm (GA), Economic Dispatch, Optimization Algorithms.

I. INTRODUCTION

In some of optimization problems, number of variables in the problem is continues and number of them is discontinues. Also search space in this problems sometimes is non-convex or discontinuous. These two factors along with other factors proposed in combined optimization and continuous optimization causes use of standard methods of optimization go inefficient, and in terms of computational are very expensive. In other words, solving above problems with classical optimization methods, causes local optimal solution in the neighborhood of the starting point. A possible way to solve such complex optimization problems using a method called a Genetic Algorithm that it has the ability to find a global optimal solution with a high probability of success.

In facts the Genetic Algorithm is a search technique for finding approximate solutions to optimization problems using concepts such as biology inheritance and mutations. This algorithm that it based on Darwin's theory of evolution is built, first, the variables are coded with the appropriate binary strings then using computer simulations laws struggle to survive constantly run and more appropriate disciplines are in fact an optimal solution can be obtained. The Genetic Algorithm is a method based on probabilities, to ensure that given results are best values run the algorithm several times and compare results. However, the probability of finding the global optimum response in the event of the use of the appropriate values for the parameters of the algorithm is huge [1-4].

II. GENETIC ALGORITHM DEFINITION

A. Representation of Design Variables

In GAs, the design variables are represented as strings of binary numbers, 0 and 1. For example, if a design variable x_i is denoted by a string of length four (or a four-bit string) as (0 1 0 1), its integer (decimal equivalent) value will be 1+0+4+0=5. If each design variable x_i, i=1, 2… n is coded in a string of length q, a design vector is represented using a string of total length nq. For example, if a string of length 5 is used to represent each variable, a total string of length 20 describes a design vector with n=4. The following string of 20 binary digits denote the vector (x_1=18, x_2=3, x_3=1, x_4=4):

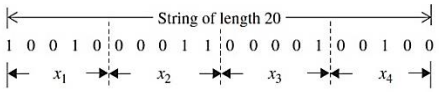

Figure 1. Example of string length

In general, if a binary number is given by $b_q b_{q-1}$, …, $b_2 b_1 b_0$, where b_k=0 or 1, k=1, 2…, q then its equivalent decimal number y (integer) is given by:

$$y = \sum_{k=0}^{q} 2^k b_k \tag{1}$$

This indicates that a continuous design variable x can only be represented by a set of discrete values if binary representation is used. If a variable x (whose bounds are given by x^l and x^u) is represented by a string of q binary numbers, as shown in Equation (1), its decimal value can be computed as:

$$x = x^l + \frac{x^u - x^l}{2^q - 1} \sum_{k=0}^{q} 2^k b_k \quad (2)$$

Thus if a continuous variable is to be represented with high accuracy, we need to use a large value of q in its binary representation. In fact, the number of binary digits needed (q) to represent a continuous variable in steps (accuracy) of Δx can be computed from the relation:

$$2^q \geq \frac{x^u - x^l}{\Delta x} + 1 \quad (3)$$

For example, if a continuous variable x with bounds 1 and 5 is to be represented with an accuracy of 0.01, we need to use a binary representation with q digits where $2^q \geq \frac{5-1}{0.01} + 1 = 401$ or $q = 9$. Equation (2) shows why GAs are naturally suited for solving discrete optimization problems [3, 5].

B. Representation of Objective Function and Constraints

Because Genetic Algorithms are based on the survival of the fittest principle of nature, they try to maximize a function called the fitness function. Thus GAs are naturally suitable for solving unconstrained maximization problems. The fitness function, $F(X)$, can be taken to be same as the objective function $f(X)$ of an unconstrained maximization problem so that $F(X) = f(X)$. A minimization problem can be transformed into a maximization problem before applying the GAs. Usually the fitness function is chosen to be nonnegative. The commonly used transformation to convert an unconstrained minimization problem to a fitness function is given by:

$$F(X) = \frac{1}{1 + f(X)} \quad (4)$$

It can be seen that Equation (4) does not alter the location of the minimum of $f(X)$ but converts the minimization problem into an equivalent maximization problem. A general constrained minimization problem can be stated as: Minimize $f(X)$ subject to $g_i(X) \leq 0$; i=1, 2, …, m and $h_j(X) \leq 0$; j=1, 2, …, p. This problem can be converted into an equivalent unconstrained minimization problem by using the concept of penalty function as:

$$\min \quad \phi(X) = f(X) + \sum_{i=1}^{m} r_i \langle g_i(X) \rangle^2 + \sum_{j=1}^{p} R_j \left(h_j(X) \right)^2 \quad (5)$$

where r_i and R_j are the penalty parameters associated with the constraints $g_i(X)$ and $h_j(X)$, whose values are usually kept constant throughout the solution process. In Equation (5), the function $\langle g_i(X) \rangle$, called the bracket function, is defined as:

$$\langle g_i(X) \rangle = \begin{cases} g_i(X) & \text{if} \quad g_i(X) > 0 \\ 0 & \text{if} \quad g_i(X) \leq 0 \end{cases} \quad (6)$$

In most cases, the penalty parameters associated with all the inequality and equality constraints are assumed to be the same constants as: r_i=r; i=1, 2, …, m and R_j=R; j=1, 2,…, p, where r and R are constants. The fitness function, $F(X)$, to be maximized in the GAs can be obtained, similar to Equation (4), as:

$$F(X) = \frac{1}{1 + \phi(X)} \quad (7)$$

The Equations (5) and (6) show that the penalty will be proportional to the square of the amount of violation of the inequality and equality constraints at the design vector X, while there will be no penalty added to $f(X)$ if all the constraints are satisfied at design vector X [3-5].

C. Genetic Operators

The solution of an optimization problem by GAs starts with a population of random strings denoting several (population of) design vectors. The population size in GAs (n) is usually fixed. Each string (or design vector) is evaluated to find its fitness value. The population (of designs) is operated by three operators' reproduction, crossover, and mutation to produce a new population of points (designs). The new population is further evaluated to find the fitness values and tested for the convergence of the process. One cycle of reproduction, crossover, and mutation and the evaluation of the fitness values is known as a generation in GAs. If the convergence criterion is not satisfied, the population is iteratively operated by the three operators and the resulting new population is evaluated for the fitness values. The procedure is continued through several generations until the convergence criterion is satisfied and the process is terminated. The details of the three operations of GAs are given below [3-5].

D. Reproduction

Reproduction is the first operation applied to the population to select good strings (designs) of the population to form a mating pool. The reproduction operator is also called the selection operator because it selects good strings of the population. The reproduction operator is used to pick above average strings from the current population and insert their multiple copies in the mating pool based on a probabilistic procedure. In a commonly used reproduction operator, a string is selected from the mating pool with a probability proportional to its fitness. Thus if F_i denotes the fitness of the string in the population of size n, the probability for selecting the ith string for the mating pool (p_i) is given by:

$$p_i = \frac{F_i}{\sum_{j=1}^{n} F_j} \quad , \quad i = 1,...,n \quad (8)$$

Note that Equation (8) implies that the sum of the probabilities of the strings of the population being selected for the mating pool is one. The implementation of the selection process given by Equation (8) can be understood by imagining a roulette wheel with its circumference divided into segments, one for each string of the population, with the segment lengths proportional to the fitness of the strings as shown in Figure 1. By spinning the roulette wheel n times (n being the population size) and selecting, each time, the string chosen by the roulette-wheel pointer, we obtain a mating

pool of size *n*. Since the segments of the circumference of the wheel are marked according to the fitness of the various strings of the original population, the roulette-wheel process is expected to select *Fi*/*F* copies of the *i*th string for the mating pool, where $\bar{F}$ denotes the average fitness of the population:

$$\bar{F} = \frac{1}{n}\sum_{j=1}^{n} F_j \tag{9}$$

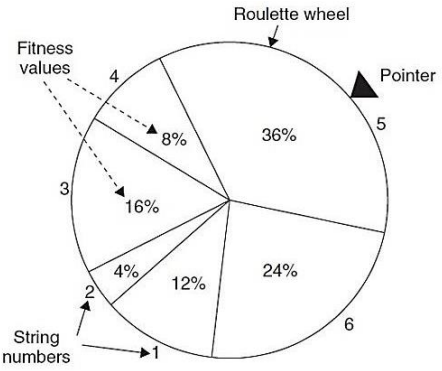

Figure 2. Roulette-Wheel selection scheme

In Figure 1, the population size is assumed to be 6 with fitness values of the strings 1, 2, 3, 4, 5, and 6 given by 12, 4, 16, 8, 36, and 24, respectively. Since the fifth string (individual) has the highest value, it is expected to be selected most of the time (36% of the time, probabilistically) when the roulette wheel is spun n times (*n*=6 in Figure 1). The selection scheme, based on the spinning of the roulette wheel, can be implemented numerically during computations as follows. The probabilities of selecting different strings based on their fitness values are calculated using Equation (8). These probabilities are used to determine the cumulative probability of string *i* being copied to the mating pool, p_i by adding the individual probabilities of strings 1 through *i* as:

$$P_i = \sum_{j=1}^{i} p_j \tag{10}$$

Thus the roulette-wheel selection process can be implemented by associating the cumulative probability range P_{i-1}-P_i to the *i*th string. To generate the mating pool of size *n* during numerical computations, *n* random numbers, each in the range of zero to one, are generated (or chosen). By treating each random number as the cumulative probability of the string to be copied to the mating pool, *n* strings corresponding to the *n* random numbers are selected as members of the mating pool. By this process, the string with a higher (lower) fitness value will be selected more (less) frequently to the mating pool because it has a larger (smaller) range of cumulative probability.

Therefore, strings with high fitness values in the population, probabilistically, get more copies in the mating pool. It is to be noted that no new strings are formed in the reproduction stage; only the existing strings in the population get copied to the mating pool. The reproduction stage ensures that highly fit individuals (strings) live and reproduce, and less fit individuals (strings) die. Thus the GAs simulate the principle of "survival-of-the-fittest" of nature [1, 3-5].

E. Crossover

After reproduction, the crossover operator is implemented. The purpose of crossover is to create new strings by exchanging information among strings of the mating pool. Many crossover operators have been used in the literature of GAs. In most crossover operators, two individual strings (designs) are picked (or selected) at random from the mating pool generated by the reproduction operator and some portions of the strings are exchanged between the strings. In the commonly used process, known as a single-point crossover operator, a crossover site is selected at random along the string length, and the binary digits (alleles) lying on the right side of the crossover site are swapped (exchanged) between the two strings. The two strings selected for participation in the crossover operators are known as parent strings and the strings generated by the crossover operator are known as child strings. For example, if two design vectors (parents), each with a string length of 10, are given by:

$$\text{(Parent 1)} \quad X_1 = \{010\,|\,1011011\}$$

$$\text{(Parent 2)} \quad X_2 = \{100\,|\,0111100\}$$

The result of crossover, when the crossover site is 3, is given by:

$$\text{(Offspring1)} \quad X_3 = \{010\,|\,0111100\}$$

$$\text{(Offspring2)} \quad X_4 = \{100\,|\,1011011\}$$

Since the crossover operator combines substrings from parent strings (which have good fitness values), the resulting child strings created are expected to have better fitness values provided an appropriate (suitable) crossover site is selected. However, the suitable or appropriate crossover site is not known beforehand. Hence the crossover site is usually chosen randomly. The child strings generated using a random crossover site may or may not be as good as or better than their parent strings in terms of their fitness values. If they are good or better than their parents, they will contribute to a faster improvement of the average fitness value of the new population. On the other hand, if the child strings created are worse than their parent strings, it should not be of much concern to the success of the GAs because the bad child strings will not survive very long as they are less likely to be selected in the next reproduction stage (because of the survival-of-the-fittest strategy used).

As indicated above, the effect of crossover may be useful or detrimental. Hence it is desirable not to use all the strings of the mating pool in crossover but to preserve some of the good strings of the mating pool as part of the population in the next generation. In practice, a crossover probability, p_c is used in selecting the parents for crossover. Thus only $100p_c$ percent of the strings in the mating pool will be used in the crossover operator while $100(1–p_c)$ percent of the strings will be retained as they are in the new generation (of population) [1, 3-5].

F. Mutation

The crossover is the main operator by which new strings with better fitness values are created for the new generations. The mutation operator is applied to the new strings with a specific small mutation probability, p_m. The mutation operator changes the binary digit (allele's value) 1 to 0 and vice versa. Several methods can be used for implementing the mutation operator. In the single-point mutation, a mutation site is selected at random along the string length and the binary digit at that site is then changed from 1 to 0 or 0 to 1 with a probability of p_m. In the bit-wise mutation, each bit (binary digit) in the string is considered one at a time in sequence, and the digit is changed from 1 to 0 or 0 to 1 with a probability p_m. numerically, the process can be implemented as follows. A random number between 0 and 1 is generated/chosen. If the random number is smaller than p_m, then the binary digit is changed. Otherwise, the binary digit is not changed. The purpose of mutation is (1)- to generate a string (design point) in the neighborhood of the current string, thereby accomplishing a local search around the current solution, (2)- to safeguard against a premature loss of important genetic material at a particular position, and (3)- to maintain diversity in the population [1, 3-5]. As an example, consider the following population of size n = 5 with a string length 10:

1 0 0 0 1 0 0 0 1 1
1 0 1 1 1 1 0 1 0 0
1 1 0 0 0 0 1 1 0 1
1 0 1 1 0 1 0 0 1 0
1 1 1 0 0 0 1 0 0 1

Here all the five strings have a 1 in the position of the first bit. The true optimum solution of the problem requires a 0 as the first bit. The required 0 cannot be created by either the reproduction or the crossover operators. However, when the mutation operator is used, the binary number will be changed from 1 to 0 in the location of the first bit with a probability of np_m.

Note that the three operator's reproduction, crossover, and mutation are simple to implement. The reproduction operator selects good strings for the mating pool, the crossover operator recombines the substrings of good strings of the mating pool to create strings (next generation of population), and the mutation operator alters the string locally. The use of these three operators successively yields new generations with improved values of average fitness of the population. Although, the improvement of the fitness of the strings in successive generations cannot be proved mathematically, the process has been found to converge to the optimum fitness value of the objective function. Note that if any bad strings are created at any stage in the process, they will be eliminated by the reproduction operator in the next generation. The GAs have been successfully used to solve a variety of optimization problems in [1, 3-5].

III. GENETIC ALGORITHM PROCEDURE

The computational procedure involved in maximizing the fitness function $F(x_1, x_2,\ldots, x_n)$ in the Genetic Algorithm can be described by the following steps:

a. Choose a suitable string length "$l=nq$" to represent the n design variables of the design vector X. Assume suitable values for the following parameters: population size m, crossover probability pc, mutation probability p_m, permissible value of standard deviation of fitness values of the population $(S_f)_{max}$ to use as a convergence criterion, and maximum number of generations (i_{max}) to be used an a second convergence criterion.

b. Generate a random population of size m, each consisting of a string of length $l=nq$. Evaluate the fitness values F_i, i=1, 2,…, m of the m strings.

c. Carry out the reproduction process.

d. Carry out the crossover operation using the crossover probability pc.

e. Carry out the mutation operation using the mutation probability p_m to find the new generation of m strings.

f. Evaluate the fitness values F_i, i=1, 2,…, m of the m strings of the new population. Find the standard deviation of the m fitness values.

g. Test for the convergence of the algorithm or process. If $S_f \leq (S_f)_{max}$, the convergence criterion is satisfied and hence process may be stopped. Otherwise, go to step h.

h. Test for the generation number. If $i \geq i_{max}$, the computations have been performed for the maximum permissible number of generations and hence the process may be stopped. Otherwise, set the generation number as $i=i+1$ and go to step (c) [1, 3-5]. A look at the steps mentioned is in Figure 3.

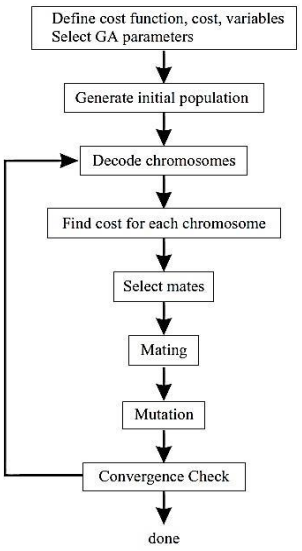

Figure 3. Flowchart of binary GA [3]

IV. CLASSIC ECONOMIC DISPATCH BY GENETIC ALGORITHM

Another type of method that is used to solve the classic Economic Dispatch problem is Genetic Algorithm. The theoretical foundation for GA was first described by Holland and was extended by Goldberg. GA provides a solution to a problem by working with a population of individuals each representing a possible solution. Each possible solution is termed a chromosome. New points of the search space are generated through GA operations, known as reproduction, crossover, and mutation. These operations consistently produce fitter offspring through successive generations, which rapidly lead the search toward global optimal [7-9, 12, 13, 14].

A. GA Based ED Solution

The classic economic dispatch problem can be stated:

$$\text{minimize } F = \sum_{i=1}^{N} F_i(P_{Gi}) \tag{11}$$

$$\sum_{i=1}^{N} P_{Gi} = P_D + P_{loss} \tag{12}$$

$$\phi = \sum_{i=1}^{N} P_{Gi} - P_D - P_{loss} = 0 \tag{13}$$

Adding penalty factor h_1 to the violation of power outputs, we can combine Equations (13) and (12):

$$F_A = \sum_{i=1}^{N} F_i(P_{Gi}) + \mathrm{h}_1 \left(\sum_{i=1}^{N} P_{Gi} - P_D - P_{loss} \right)^2 \tag{14}$$

The value of the penalty factor should be large so that there is no violation for unit output at the final solution. Since GA is designed for the solution of maximization problems, the GA fitness function is defined as the inverse of Equation (14).

$$F_{fitness} = 1/F_A \tag{15}$$

In the economic dispatch problem, the problem variables correspond to the power generations of the units. Each string represents a possible solution and is made of substrings, each corresponding to a generating unit. The length of each substring is decided based on the maximum/minimum limits on the power generation of the corresponding unit and the solution accuracy desired. The string length, which depends on the length of each substring, is chosen based on a trade-off between solution accuracy and solution time. Longer strings may provide better accuracy but result in more solution time. Thus the step size of the unit can be computed as follows: [2, 3]

$$\varepsilon_i = \frac{P_{Gi\max} - P_{Gi\min}}{2^n - 1} \tag{16}$$

V. SIMULATION

To show a simple example we assume a system that has two power generators which fuel cost functions by \$/h is as follow:

$$F_1 = 500 + 5.3P_1 + 0.004P_1^2$$
$$F_2 = 400 + 5.5P_2 + 0.006P_2^2 \tag{17}$$

The powers are determined by MW and total load is 600MW which the losses are neglected. Generators constraints (by MW) are as follow:

$$200 \le P_1 \le 450 \quad , \quad 200 \le P_2 \le 450 \tag{18}$$

In this paper we use Matlab software programing tool for write a program for solve this simple problem. The results are as follow:

$$P_1 = 369.7192 \text{ MW} \quad , \quad P_2 = 230.2808 \text{ MW} \tag{19}$$

and the optimum fuel cost is equal to 4991\$/h [2, 6-9].

VI. COMPARISON RESULTS OF GENETIC ALGORITHM AND LAGRANGIAN METHODS

For comparison we need lagrangian method results, the Lagrangian method results same as follow:

$$P_1 = 370 \text{ MW} \quad , \quad P_2 = 230 \text{ MW} \tag{20}$$

By comparison this results with results of Genetic Algorithm we understand the accuracy of GA is good if we choose parameters appropriate. Also speed of GA for searching in the big area and more number of generators is better than other search methods. Another advantage is that GA don't need to differential of cost function, and use cost function to find better result. This property is very important when my cost function curve isn't continues or cost function is piece lines.

Table 1. Results of three algorithms for solving assumed ED [14]

Parameters	ACO	GA	Lagrangian	P_{load}(MW)
$P1$ (MW)	374.6425	369.7192	370	600
$P2$ (MW)	225.3525	230.2808	230	600
Cost (\$/h)	4991.2	4991	4991	600

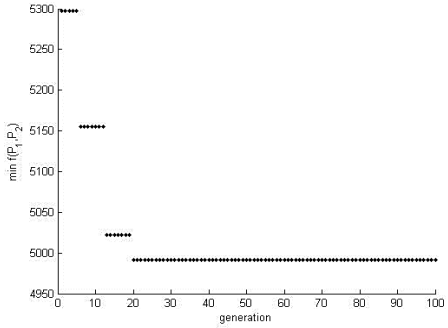

Figure 4. Minimum of cost function

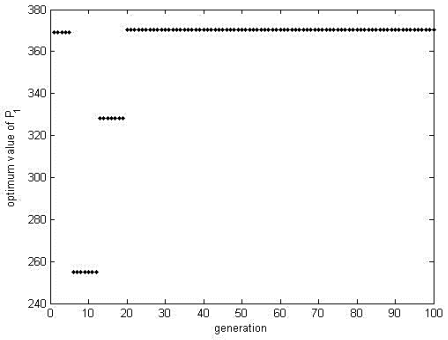

Figure 5. Value of Power output of unit 1

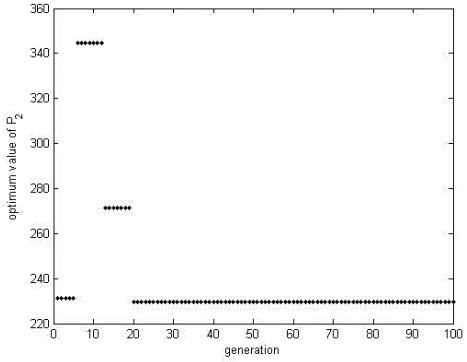

Figure 6. Value of power output of unit 2

VII. CONCLUSIONS

In this paper we have seen that genetic algorithms can be a powerful tool for solving problems and for simulating natural systems in a wide variety of scientific fields. An optimization problem because of increase security and decrease costs in power systems always was

important. So methods of optimization in power systems should have high accuracy and high speed. Some of this methods have some constraint that can affect performance of networks, and we try to troubleshooting this problems. Genetic Algorithm is method that try to troubleshooting this problem, GA has high speed, global search, high accuracy and compatible with restrictions. This advantages make GA as powerful optimization method. So we can learn GA and use it in any problem that need to optimization.

REFERENCES

[1] S.R. Singiresu, "Engineering Optimization - Theory and Practice", 4th Edition, John Wiley & Sons, 2009.
[2] J. Zhu, "Optimization of Power System Operation", IEEE Press Series on Power Engineering, John Wiley & Sons Inc., 2008.
[3] R.L. Haupt, S.E. Haupt, "Practical Genetic Algorithms", 2nd Edition, John Wiley & Sons Inc., 2004
[4] F. Merrikh Bayat, "Optimization Algorithms Inspired by Nature", Zanjan University, Zanjan, Iran, 2012.
[5] M. Melanie, "An Introduction to Genetic Algorithms", MIT Press, 1996.
[6] N.M. Tabatabaei, F. Rajabi "Optimization of Power Generation Distribution with Genetic Algorithm", 9th Baku International Congress on Energy, Ecology and Economy, pp. 70-75, International Ecoenergy Academy, Baku, Azerbaijan, June 7-9, 2007.
[7] "Economic Dispatch, Concepts, Practices and Issues FERC", Staff Palm Springs, California, Nov. 13, 2005.
[8] H. Saadat, "Power System Analysis", McGraw-Hill, 1999.
[9] H.H. Happ, "Optimal Power Dispatch", IEEE Trans. on Power Apparatus and Systems, PAS-93, pp. 820-830, 1974.
[10] A.J. Wood, B.F. Wollenberg, "Power Generation, Operation and Control", Wiley Interscience Publication, John Wiley & Sons Inc., 1996.
[11] A.G. Bakirtzis, P.N. Biskas, C.E. Zoumas, V. Petridis, "Optimal Power Flow by Enhanced Genetic Algorithm", IEEE Transactions on Power Systems, Vol. 17, No. 2, pp.229-236, May 2002,.
[12] G.B. Sheble, K. Brittig, "Refined Genetic Algorithm-Economic Dispatch Example", IEEE/PES Winter Meeting, Paper 94 WM 199-0 PWRS, 1994.
[13] K.P. Wong, Y.W. Wong, "Genetic and Genetic/Simulated-Annealing Approaches to Economic Dispatch", Inst. Elect. Eng., Gener. Transm. Distrib., Vol. 141, No. 5, pp. 507-513, Sep. 1994.
[14] N.M. Tabatabaei, A. Jafari, N.S. Boushehri, K. Dursun, "Ant Colony Algorithm Application in Economic Load Distribution", International Journal on Technical and Physical Problems of Engineering (IJTPE), Issue 16, Vol. 5, No. 6, pp. 155-160, September 2013.

BIOGRAPHIES

Naser Mahdavi Tabatabaei was born in Tehran, Iran, 1967. He received the B.Sc. and the M.Sc. degrees from University of Tabriz (Tabriz, Iran) and the Ph.D. degree from Iran University of Science and Technology (Tehran, Iran), all in Power Electrical Engineering, in 1989, 1992, and 1997, respectively. Currently, he is a Professor in International Organization of IOTPE. He is also an academic member of Power Electrical Engineering at Seraj Higher Education Institute (Tabriz, Iran) and teaches power system analysis, power system operation, and reactive power control. He is the General Secretary of International Conference of ICTPE, Editor-in-Chief of International Journal of IJTPE and Chairman of International Enterprise of IETPE all supported by IOTPE. He has authored and co-authored of six books and book chapters in Electrical Engineering area in international publishers and more than 130 papers in international journals and conference proceedings. His research interests are in the area of power quality, energy management systems, ICT in power engineering and virtual e-learning educational systems. He is a member of the Iranian Association of Electrical and Electronic Engineers (IAEEE).

Ali Jafari was born in Zanjan, Iran in 1988. He received the B.Sc. degree in Electrical Engineering from Abhar Branch, Islamic Azad University, Abhar, Iran in 2011. He is currently the M.Sc. student in Seraj Higher Education Institute, Tabriz, Iran. He is the Member of Scientific and Executive Committees of International Conference of ICTPE and also the Scientific and Executive Secretary of International Journal of IJTPE supported by International Organization of IOTPE (www.iotpe.com). His research fields are power system analysis and operation, and reactive power control.

Narges Sadat Boushehri was born in Iran. She received her B.Sc. degree in Control Engineering from Sharif University of Technology (Tehran, Iran), and Electronic Engineering from Central Tehran Branch, Islamic Azad University, (Tehran, Iran), in 1991 and 1996, respectively. She received the M.Sc. degree in Electronic Engineering from International Ecocenergy Academy (Baku, Azerbaijan), in 2009. She is the Member of Scientific and Executive Committees of International Conference of ICTPE and also the Scientific and Executive Secretary of International Journal of IJTPE supported by International Organization of IOTPE (www.iotpe.com). Her research interests are in the area of power system control and artificial intelligent algorithms.

Kamil Dursun was born in Ankara, Turkey, 1954. He received the M.Sc. and the Ph.D. degrees from The Norwegian Technical University, all in Power Electrical Engineering, in 1978 and 1984, respectively. He worked with ABB in several countries. Currently, he is an Associate Professor of Power Engineering at Ostfold University College (Fredrikstad, Norway). He is the secretary of International Conference of ICTPE. His research interests are in the area of efficient power distribution, energy management systems and virtual e-learning educational systems.