A GENETIC-BASED APPROACH FOR SOLVING OPTIMAL POWER FLOW PROBLEM

طريقة جينية لحل مشكلة التوزيع الأمثل للأحمال الكهربية

M. M. El-Saadawi
Dept. of Elect. Power & Machines
Faculty of Engineering
Mansoura University

خلاصة:

إن مشكلة التوزيع الأمثل للأحمال الكهربية من مشاكل الأمثلة التي تسعى شبكات الكهرباء من خلالها لتقليل التكلفة مع استيفاء كافة القيود. وتستخدم طرق الذكاء الاصطناعي لمساعدة الشبكات الكهربية في تحديد كيفية تغذية الأحمال الكهربية بشكل اقتصادي. وتعتبر الطريقة الجينية أحد أنواع طرق الذكاء الاصطناعي المستخدمة لتحقيق هذا الغرض. ويقدم هذا البحث خوارزمية جينية تستعمل لحل مشكلة التوزيع الأمثل للأحمال الكهربية. وتعتمد الطريقة المقترحة على بناء كروموسوم وراثي جديد ينظم تمثيل الحلول. ويتم بناء الكروموسوم المقترح بحيث يخفض بشكل كبير عدد مرات حل معادلات تدفق الحمل الذي يجب على الطرق الجينية أن تقوم به. وهذا يؤدى إلى تسريع الحل بدرجة كبيرة. ويقدم البحث برنامج حاسب كتب في بيئة **Matlab** لتمثيل الطريقة المقترحة. وتم تطبيق هذا البرنامج على نظام قوى كهربي من النوع القياسي **IEEE 30-bus** ومقارنة النتائج مع تلك التي سبق الحصل عليها بالطرق التقليدية لتوضيح درجة دقة الطريقة وامكانية تطبيقها. وكذلك تم التطبيق على نظام**IEEE 118-bus** لتوضيح قدرة الطريقة المقترحة على حل مشاكل نظم القوى الكبيرة.

Abstract

The optimal power flow (OPF) is an optimization problem, in which the utility strives to minimize its costs while satisfying all of its constraints. Artificial intelligence is used to help a hypothetical electric utility meet its electric load economically. A genetic algorithm (GA)—a specific type of artificial intelligence—is employed to perform this optimization .

In this paper, a genetic algorithm is used to solve the OPF problem. A new genetic chromosome is structured to represent the solutions. The new chromosome structure is chosen in such a way that it greatly reduce the number of times the algorithm must solve the load-flow equations. Since solving the load-flow equations is time-consuming, this speeds execution of the algorithm considerably .

A computer program, written in Matlab environment, is developed to represent the proposed method. The program is applied to both the IEEE 30-bus test system, and the IEEE 118-bus test system to demonstrate its ability and its potential to be used with larger systems. Thus, the proposed algorithm is shown to be a valid tool to perform this optimization.

1. Introduction

Genetic algorithms are essentially search algorithms based on mechanics of nature and natural genetics. They combine solution evaluation with randomized, structured exchanges of information between solutions to obtain optimality. Genetic algorithms are considered to be robust methods because restrictions on solution space are not made during the process. The power of this algorithm stems from its ability to exploit historical information structures from previous solution guesses in an attempt to increase performance of future solutions [1].

Many power system problems are large enough that achieving adequate

Accepted June 30, 2004.

coverage is difficult and computationally costly. Optimal power flow (OPF) is one of the optimization problems in power system operation. The OPF algorithm is to allocate the total electric power demand (and losses) among the available generators in such a manner that minimizes the electric utility's total fuel cost [2]. Presently, application of optimal power flow is of much importance in power system operation analysis under deregulated environment of electricity industry. It is highly constrained and has large dimensions nonlinear nonconvex optimization problems, in particular when Flexible AC Transmission Systems (FACTS) devices are also present in the system. Under such situations either classical methods fail to provide any solution or provide only a very approximate solution [3]. There are many methods for solving this problem using genetic algorithm [3-7]. The existing GA-based optimal power flow can provide a reasonable solution, but they have long computation time for large scale problems.

In this paper, a genetic algorithm is used in a new way to solve the OPF problem. A new genetic chromosome is structured to represent the solutions. The new chromosome structure is chosen in such a way that it greatly reduces the number of times the algorithm must solve the load-flow equations. Since solving the load-flow equations is time-consuming, this speeds execution of the algorithm considerably. To demonstrate the effectiveness of the GA-OPF method, it is tested on test systems of varying complexity.

2. Problem Statement

2.1 Optimal Power Flow Equations

In order to compute the power flows in a power system, the system's bus admittance matrix, Y_{BUS}, must be defined. If V and I are respectively vectors of all bus voltages and net injected currents in the system, the bus admittance matrix will satisfy [2].

$$I = Y_{Bus} V \tag{1}$$

where Y_{Bus} is a square matrix which depends on the admittance of all transmission lines in the system. Let y_{Si} be the shunt admittance connected at bus i, and let y_{ij} be the series admittance connecting buses i and j. The elements of Y_{BUS} are defined as [2]

$$Y_{BUS} = \begin{cases} -y_{ij} & i \neq j \\ y_{Si} + \sum_{m \neq i} y_{im} & i = j \end{cases} \tag{2}$$

In the optimal power flow problem, it is necessary to find a relationship between the voltage magnitudes and angles and the real and reactive power at the buses. For bus l, let V_l and δ_l be the voltage magnitude and angle, respectively. Furthermore, let the P_{Gl} be the real power generated, let P_{Dl} be the real power demand (the real power load), let Q_{Gl} be the reactive power generated, and let Q_{Dl} be the reactive power demand. Then, the net real and reactive power at bus l are given by the load-flow equations [2]:

$$P_l = P_{Gl} - P_{Dl} = V_l^2 G_{ll} - V_l \sum_{m \in k(l)} V_m T_{lm} \tag{3}$$

$$Q_l = Q_{Gl} - Q_{Dl} = V_l^2 B_{ll} - V_l \sum_{m \in k(l)} V_m U_{lm} \tag{4}$$

where

$$T_{ij} = G_{ij} \cos(\delta_i - \delta_j) + B_{ij} \sin(\delta_i - \delta_j) \tag{5}$$

$$U_{ij} = G_{ij}\sin(\delta_i - \delta_j) - B_{ij}\cos(\delta_i - \delta_j) \qquad (6)$$

Note that the Jacobian is defined in terms of T_{ij} and U_{ij}, which are themselves defined in terms of the elements of Y_{bus}.

2.2 Economic Dispatch and Optimal Power Flow

Traditional economic dispatch methods are based on setting incremental costs of all units equal to each other. Losses are accounted for by incorporating penalty factors in the incremental cost. However, the equal-incremental-cost method is optimal only if the incremental cost curves are monotonically increasing [3], which are not always true. In practical applications, the incremental cost functions are often constrained to be monotonically increasing, regardless of the generator's actual behavior [8].

2.3 Implementation of a Genetic Algorithm

GAs are general purpose optimization algorithms based on the mechanics of natural selection and genetics. They operate on string structures (chromosomes), typically a concatenated list binary digits representing a coding of the control parameters phenotype of a given problem. Chromosomes themselves are composed of genes. The real value of a control parameter, encoded in a gene, is called an allele. GAs are an attractive alternative to other optimization methods because of their robustness. There are three major differences between GAs and conventional optimization algorithms. First, GAs operate on the encoded string of the problem parameters rather than the actual parameters of the problem. Each string can be thought of as a chromosome that completely describes one candidate solution to the problem. Second, GAs use a population of points rather than a single point in their search. Searching the space with GA population increases the probability of finding local optima. But the point is it does reduce the probability of ending up with a solution at local optimum but most probability at the global optimum. Third, GAs do not require any prior knowledge, space limitations, or special properties of the function to be optimized, such as smoothness, convexity, unimodality, or existence of derivatives. They only require the evaluation of the so-called fitness function (FF) to assign a quality value to every solution produced [3].

2.4 Use of Linear Algebra to Improve Convergence of the GA

Although a GA is an efficient search technique for large problems, its convergence can be improved significantly by encoding the candidate solutions in such a way that avoids generating illegal candidate solutions. For example, equality constraints are difficult to implement with a GA. One technique is to use the equality constraints to solve for some of the control variables in terms of the others [1]. This has the effect of narrowing the search space and reducing the dimensionality of the problem (since there are fewer unknowns remaining). Furthermore, this avoids wasting computation effort unnecessarily on illegal solutions.

In the OPF problem, it is not feasible to use the load-flow equality constraints to eliminate state variables. Enforcing the equality constraints requires solving the load flow equations, which is a computationally intense task. Instead, the search space is reduced via the representation of the candidate solutions. For a power system with N buses and N_g generation buses, there are $2N$ state variables (voltage magnitude and angle at each bus) but only $2N_g$ control variables (real and reactive power at each generator). If the GA produced a candidate solution by randomly choosing a list of $2N$ state variables, the solution likely would fail to meet the equality constraints. In other words, such a solution is unlikely to have the correct amount of real and reactive power at all $(N\text{–}N_g)$ load buses.

Thus, the equality constraints restrict the choice of values for the state variables. Let J be the load-flow Jacobian matrix. Here, all buses—even the slack bus—are represented in the Jacobian. Thus, J is a $2N \times 2N$ matrix. Let J_L be rectangular matrix formed by taking the rows of J corresponding to the load buses. In other words, any partial derivative involving P or Q at a load bus is kept. Thus, J_L is a $2(N\text{–}N_g) \times 2N$ rectangular submatrix of J. The matrix J_L has 2 rows for each load bus (corresponding to one P and one Q for each load bus) and 2 columns for each bus of any kind (corresponding to one voltage magnitude and one voltage angle for each bus, whether it is a load bus or not).

Let x be a state vector that satisfies the equality constraints. Any change to the state vector, Δx, will change the power vector by:

$$\Delta S = J\Delta x \tag{7}$$

where the state vector, x, and power vector, S, are defined as:

$$x = \begin{bmatrix} \delta \\ V \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} P \\ Q \end{bmatrix} \tag{8}$$

3. Solution Algorithm

The solution is composed of four parts: selecting the control variables, choosing the genetic operators and fitness function, customizing the GA for the problem at hand, and applying the load-flow equations efficiently.

3.1 Choosing the Control Variables

In the OPF problem, there are four important quantities: voltage magnitude, voltage angle, real power, and reactive power. Of these four quantities, two are independent (control, or input) variables and two are dependent (output) variables. For a traditional OPF problem, the unit incremental cost funetions are used to optimize the real and reactive power (which are the control variables in this formulation). Mathematically, the choice of independent variables is not important. For computational speed, however, choosing voltage magnitudes and angles as the independent variables will allow the algorithm to avoid solving load-flow problems for each candidate solution. Although one load-flow problem may not require a great deal of speed, evaluating many load-flows (one for each member of the population, at each generation) is quite slow.

GA convergence is much improved if redundant control variables are removed, and only an independent subset is considered. That is, it is often beneficial to use the equality constraints to eliminate unnecessary control variables. Moreover, to reduce computational effort spent on illegal solutions, the linear algebra nullspace technique is used to reduce the search space. The nullspace eliminates many (but not all) illegal solutions before they are considered. Thus, for this OPF problem, the GA control variables are chosen as:

1. Nullspace coefficients, to specify which member of the nullspace is used
2. Tap settings for the tap-changing transformers
3. Amount of VAR compensation

Each GA chromosome is a list of numbers that provides the values of these control variables. To change the transformer tap settings, the system Y-bus matrix is modified to account for the transformer's new impedance.

Once all control and output variables are known, the fitness of the candidate solution is computed.

3.2 Choosing the Genetic Operators and Fitness Function

A genetic *operator* is a set of rules for extracting new solutions from older ones. The selection of genetic operators is often a heuristic process. A *fitness* function is defined to quantify the quality of any particular candidate solution. A good choice of operators and fitness function for one type of problem can be a poor choice for another problem.

Sometimes, the choice of operators depends on the choice of fitness function. Thus, the fitness function has been included in this discussion of genetic operators.

3.2.1 Fitness Function

In this paper, the fitness function is chosen as [3]:

$$f = \frac{1}{1 + C_T + P} \qquad (9)$$

where C_T is the total generation cost and P is the penalty if any output variable violates a constraint. This penalty is the weighted sum, over all output variables, of the amount each variable exceeds its constraint. Of course, if a variable is within its allowable limits, its contribution to the penalty is zero. The weighting factors are chosen to be 10,000 for voltage magnitudes, 10,000 for line flows, and 1000 for all other variables. This choice of fitness function maps a cost in the interval $[0,\infty)$ to the interval $(0,1]$. Thus, a solution with an infinite cost (or infinite penalty) has a fitness of 0. A perfect solution (one with zero cost) has a fitness of 1.

Note that this penalty weight is not the price of power or of anything else. Instead, the weight is a coefficient set large enough to prevent the algorithm from converging to an illegal solution.

3.2.2 Genetic Operators

Crossover operators are used to generate new solutions by taking information from previous solutions. Since the GA used here works with lists of real numbers, two crossover operators used here are arithmetic crossover and two-point crossover. These operators have the advantage that they will always generate a set of control variables within their allowable ranges, provided that the original solutions were legal. However, these operators do not guarantee that a solution will satisfy

the other constraints (such as line-flow limits), even if the parents satisfied them. To illustrate arithmetic crossover, let x_1 and x_2 be vectors containing the coefficients of two "parents"—candidate solutions chosen to participate in the crossover. The two "children"—new candidate solutions resulting from the crossover—are formed by taking two weighted averages of the parents. Let a be a random number between 0 and 1.
Arithmetic crossover calculates the children according to the following equations [6]:

$$y_1 = a\,x_1 + (1-a)\,x_2 \qquad (10)$$

$$y_2 = (1-a)\,x_1 + a\,x_2 \qquad (11)$$

In contrast, two-point crossover combines information from two parents in a fundamentally different way. It literally breaks the parents apart, exchanges some of the pieces, and recombines the pieces to form two new solutions. This is illustrated in Figure 1, which shows one example of how the operator might produce children from two arbitrary parents.

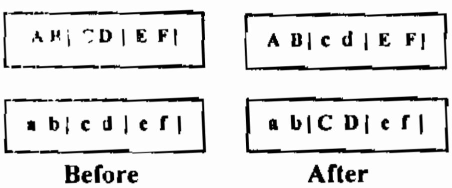

Figure 1. Illustration of Two-point Crossover

For illustrative purposes, the chromosomes (the subdivisions of the parents) are represented by the letters *A-F* and *a-f*. In the OPF problem, the chromosomes are real numbers. The crossover operator randomly selects the portion of the parents it will alter.
In this example, it is assumed that the operator will cut the parents at the positions indicated by the vertical bars—after the second and fourth positions. The two vertical bars indicate the "two points" which give this operator its name. The effect of two-point crossover is to exchange all genes appearing between the two points. Mutation operators are used both to avoid premature convergence of the population (which may cause convergence to a local, rather than global, optimum) and to fine-tune the solutions.

Two forms of mutation are used here: uniform and non-uniform mutation. In both kinds of mutation, a randomly chosen gene of a randomly chosen candidate chromosome (solution) is replaced with a new, randomly generated value. In uniform mutation, the new value is allowed to be any legal value. This provides coarse adjustment of the solutions. In non-uniform mutation, the new value is taken from a smaller and smaller neighborhood of the original value. This provides fine tuning of the solutions. Let v_k be the k^{th} chromosome of the gene v. That is, v is one complete set of parameters, and k is the randomly chosen piece of the solution to be modified. Let l_k and u_k be lower and upper limits on v_k. For the t^{th} GA generation, non-uniform mutation will replace v_k with a new chromosome v_k', which is formed according to [6]:

$$v_k' = \begin{cases} v_k + \Delta(t, u_k - v_k) & d = 0 \\ v_k - \Delta(t, u_k - l_k) & d = 1 \end{cases} \qquad (12)$$

where d is a random digit that specifies whether to increase or decrease the chromosome.

The function $\Delta(t,y)$ returns a value in the interval $[0, y]$ and is defined as:

$$\Delta(t,y) = y(1 - r^{(1-t/T)^b}) \quad (13)$$

where T is the total number of GA generations to be run, b is a parameter that specifies how fast the function $\Delta(t,y)$ should converge to 0, and r is a random number between 0 and 1. The probability that $\Delta(t,y)$ is close to 0 increases as t increases. If t equals T (that is, if the GA is performing its last generation), the function $\Delta(t,y)$ equals 0. In other words, the function converges to 0 as the GA generations progress. The non-uniform mutation operator is useful because it allows a coarse search at first (when $t \ll T$), but gradually narrows the search as the algorithm runs. This allows fine local tuning of the solutions.

3.3 Customizing the Genetic Algorithm for OPF

In order to improve its convergence, the GA is customized for the OPF problem as follows:

3.3.1 General GA parameters

The GA was run with a population size of 20 candidate solutions. The population was allowed to evolve for 10 generations. Elitism is used to guarantee that the best 5% of the population survives into the next generation. Some researchers evolve the population until the population becomes homogeneous (or nearly so). However, in this paper, evolution progresses for a fixed number of generations. Crossover probabilities are taken as 0.02, whereas, probability of both uniform and non-uniform parameter mutation are taken as 0.01 [7].

3.3.2 Accounting for Static-VAR compensation

If the static-VAR compensation has changed, a load-flow solution is required to get an exact answer. However, performing load-flow solutions is time-consuming and therefore undesirable. Thus, to save time, the effects of the static-VAR compensation are approximated through Equation (7), which uses the Jacobian to approximate the effects of a change in reactive power on the states. This approximation is not accurate enough to distinguish between two solutions of similar quality. Thus, the approximation is sufficient to determine which solutions are of poor quality and which are promising, but a fast-decoupled load flow must be used to determine the exact effect of the VAR compensation on the good-quality solutions.

3.3.3 Re-calibrating the linearization of the load-flow equations

Since the load-flow Jacobian is a linearized matrix, it is necessary to update the Jacobian if the GA's best solution has changed significantly. Recall that all members of the GA population (that is, all candidate solutions) are defined in terms of their difference with the best solution. Thus, whenever a new solution is found that improves the fitness by at least 1%, the load-flow Jacobian, rectangular submatrix, and nullspace are recalculated.

The candidate solutions are then projected onto the new nullspace. This projection is accomplished in several steps. First, the best solution in the population is chosen as the reference solution. Its state vector is used to compute the Jacobian, and all other solutions are defined with respect to this reference. For every candidate solution, the old nullspace is used to convert the nullspace eoefficients into a corresponding state vector. This state vector is substituted into the load flow equations to get the resulting real and reactive power at each generator. Because modeling errors resulting from the linearization inherent in computing the Jacobian, the real and reactive power at the load buses may not be exactly at their required values—particularly if the state vector varies greatly from the reference state vector used in computing the Jacobian. Even small changes in the states can lead to significant changes in power. To counteract this error, the load bus real and reactive powers are re-set to their required values. The new real and reactive powers are then input to a standard load-flow program to find the resulting, new state vector. The difference between the new state vector and the reference state vector is then projected onto the nullspace, which gives the updated list of nullspace coefficients for the GA population.

3.3.4 Seeding the initial GA population

In theory, the GA should be able to converge from a completely random set of initial guesses (random initial population)—if the GA is allowed to evolve for enough generations. However, convergence is hastened if any prior knowledge of the problem is incorporated into the algorithm. One of the contributions of this work is to speed convergence by not wasting time solving load-flow equations. Because of the nullspace method employed in this work, the power at load buses is never altered (to the extent that the linearization is accurate). Therefore, the initial reference guess is required to have the correct power at the load buses. This can be accomplished either by solving for the reference state via a load-flow solution or by using a state vector that is known to satisfy the load bus power requirements.

In this work, the population is seeded with initial solutions given in the literature. The 30-bus system is seeded with the initial solution used by Alsac and Stott [9]. The 118-bus system is seeded with the state vector similar to the one given by Reid and Hasdorff [10].

3.4 Applying the Load-flow Equations

In order to apply the load-flow equations efficiently, a relationship is derived to account for changes in transformer tap settings without recomputing the relevant quantities from scratch. Moreover, some convergence issues are addressed.

In order to account for changes in transformer taps, the first step is to update the system Y_{bus} matrix.

Next, it is necessary to update the load-flow Jacobian. As with the Y_{bus} matrix, it is possible—but not desirable—to recompute the Jacobian from scratch each time a tap setting is changed. Instead, a contribution of this work is the derivation of the tap settings' effect on the Jacobian. Changing one transformer's tap setting alters a 4×4 submatrix of the Jacobian. Let this submatrix be partitioned into four 2×2 submatrices:

$$\Delta J_{4\times4} = \begin{bmatrix} \Delta J_{11} & \Delta J_{12} \\ \Delta J_{21} & \Delta J_{22} \end{bmatrix} \quad (14)$$

To calculate ΔJ_{4x4}, we change one transformer tap setting at a time and subtract the old Jacobian from the new. The matrix ΔJ_{4x4} will be 0 at the positions of J not affected by the transformer. The only elements of J affected by a transformer are those elements that depend on the Y_{bus} elements connected to the transformer. Thus, the change in J will depend on the changes in Y_{bus}, so that:

$$\Delta Y_{12} = (t_0 - t)Y_L \quad (15)$$

Let ΔG and ΔB be defined respectively as the real and imaginary parts of ΔY_{12}. Similarly, let ΔG_{PP} and ΔB_{PP} be defined respectively as the real and imaginary parts of ΔY_{PP}. Define V_P and V_S respectively as the voltage magnitude at the primary and secondary of the transformer. Similarly, define δ_P and δ_S as the corresponding voltage angles. For convenience, define:

$$G_S = \Delta G \sin(\delta_P - \delta_S) \quad (16)$$

$$G_C = \Delta G \cos(\delta_P - \delta_S) \quad (17)$$

$$B_S = \Delta B \sin(\delta_P - \delta_S) \quad (18)$$

$$B_C = \Delta B \cos(\delta_P - \delta_S) \quad (19)$$

where the subscripts attached to G and B (that is, S or C) refer to whether the variables are defined in terms of the *sine* or *cosine* of the difference in angle at the primary and secondary.

The submatrices in Equation (14) are found to be:

$$\Delta J_{11} = V_P V_S \begin{bmatrix} -G_S + B_C & G_S - B_C \\ -G_S - B_C & G_S + B_C \end{bmatrix} \quad (20)$$

$$\Delta J_{12} = \begin{bmatrix} V_S(G_C + B_S) + 2V_P G_{PP} & V_P(G_C + B_S) \\ V_S(G_C - B_S) & V_P(G_C - B_S) \end{bmatrix} \quad (21)$$

$$\Delta J_{21} = V_P V_S \begin{bmatrix} G_C + B_S & -G_C - B_S \\ -G_C + B_S & G_C - B_S \end{bmatrix} \quad (22)$$

$$\Delta J_{22} = \begin{bmatrix} V_S(G_S - B_C) - 2V_P B_{PP} & V_P(G_S - B_C) \\ V_S(-G_S - B_C) & V_P(-G_S - B_C) \end{bmatrix} \quad (23)$$

4. *Numerical Applications*

A computer program has been developed to perform the proposed methodology. The program was written in MATLAB environment and run on 1.7 GHz Pentium IV computer. Two example cases are studied to illustrate the applicability of the approach to practical application.

4.1 IEEE 30-Bus Test System

The first test system is the IEEE 30-bus, 41 branch system [8-9]. It has a total of 24 control variables as follows: five unit active power outputs, six generator-bus voltage magnitudes, four transformer-tap settings, and nine bus shunt admittances. The generator data of this test system is given in Appendix (A). A summary of the output results are shown in Table 1. The generated power and total cost per hour are compared with that obtained by Alsac and Stott's paper [8] to demonstrate the accuracy of the algorithm, the comparison is shown in Table 2

Table 1, GA Optimal Power Flow Results for 30 Bus System

Bus No.	Voltage Pu	Angle deg.	Load		Generated Power		Injected Power, MVAR
			MW	MVAR	MW	MVAR	
1	1.032	0	0	0	170.32	-34.179	0
2	1.027	-3.3335	21.7	12.7	53.452	66.389	0
3	1.0123	-5.0768	2.4	1.2	0	0	0
4	1.0076	-6.0959	7.6	1.6	0	0	0
5	0.99437	-8.8272	94.2	19	20.814	25.634	0
6	1.0023	-7.1570	0	0	0	0	0
7	0.99058	-8.3792	22.8	10.9	0	0	0
8	1.0078	-7.7099	30	30	19.605	28.05	0
9	1.0077	-8.3895	0	0	0	0	0
10	1.0009	-10.6086	5.8	2	0	0	19
11	1.0189	-5.4921	0	0	13.454	0.75218	0
12	0.98883	-9.5223	11.2	7.5	0	0	0
13	1.0095	-8.0096	0	0	14.504	36.663	0
14	0.97663	-10.5433	6.2	1.6	0	0	0
15	0.97507	-10.7467	8.2	2.5	0	0	0
16	0.98623	-10.3095	3.5	1.8	0	0	0
17	0.99086	-10.7547	9	5.8	0	0	0
18	0.97131	-11.4755	3.2	0.9	0	0	0
19	0.97241	-11.6927	9.5	3.4	0	0	0
20	0.97869	-11.4870	2.2	0.7	0	0	0
21	0.98796	-11.1260	17.5	11.2	0	0	0
22	0.98853	-11.1214	0	0	0	0	0
23	0.97174	-11.3122	3.2	1.6	0	0	0
24	0.97624	-11.6726	8.7	6.7	0	0	4.3
25	0.98639	-11.7890	0	0	0	0	0
26	0.96815	-12.2353	3.5	2.3	0	0	0
27	1.0017	-11.5666	0	0	0	0	0
28	0.99727	-7.6859	0	0	0	0	0
29	0.98138	-12.8512	2.4	0.9	0	0	0
30	0.96964	-13.7743	10.6	1.9	0	0	0
Total			**283.4**	**126.2**	**292.149**	**123.30**	**23.3**

Table 2, Comparison between Traditional and GA Approaches

Quantity	**Alsac and Stott [8]**	**Proposed Approach**
Cost per hour	$802	$801.44
P(1), MW	176	170.32
P(2)	49	53.45
P(5)	22	20.81
P(8)	22	19.61
P(11)	12	13.45
P(13)	12	14.51

Thus, the proposed approach was able to find a cost within 0.04 % of that by Alsac and Stott. This demonstrates the algorithm's accuracy in finding an answer.

4.2 IEEE 118-Bus Test System

In order to demonstrate the algorithm on a more complicated system, the algorithm was run on the IEEE 118-bus system. The 118-bus data was gathered from a variety of sources [8-11] and is given in Appendix (B). For the test system, the Newton-Raphson method failed to converge. Instead, it gave unrealistic voltage values such as 10^5 p.u. However, the fast decoupled load flow did converge for this system. A summary of the output of the computer program are shown in Table 3. The resultant total cost per hour is \$18726.8. Convergence requires approximately 15 minutes.

Table 3, GA Optimal Power Flow Results for 118 bus System

Bus No.	Voltage p.u.	Angle Deg.	Load		Generation		Injected MVAR
			MW	MVAR	MW	MVAR	
1	1.0289	0	51	27	110.96	54.024	0
2	1.0073	-1.1682009	20	9	0	0	0
3	1.0164	-0.8332315	39	10	0	0	0
4	1.0083	0.06436593	30	12	0	0	0
5	1.0121	0.36175898	0	0	0	0	-40
6	1.0014	-1.4157181	52	22	0	0	0
7	1.0009	-1.5027502	19	2	0	0	0
8	1.0136	3.29982811	0	0	0	0	0
9	1.0491	9.69155844	0	0	0	0	0
10	1.06	16.3200917	0	0	402.74	-41.409	0
11	0.99736	-1.5924179	70	23	0	0	0
12	1.002	-1.2816463	47	10	184.76	-4.0382	0
13	0.98453	-4.0012414	34	16	0	0	0
14	0.9999	-3.4289152	14	1	0	0	0
15	1.0023	-7.7114209	90	30	0	0	0
16	1.004	-3.4436402	25	10	0	0	0
17	1.0338	-5.644767	11	3	0	0	0
18	1.0089	-7.842628	60	34	0	0	0
19	0.99504	-8.4258976	45	25	0	0	0
20	0.98968	-8.2167685	18	3	0	0	0
21	0.98969	-7.1757066	14	8	0	0	0
22	0.99903	-5.2797364	10	5	0	0	0
23	1.0237	-1.4154316	7	3	0	0	0
24	1.0028	-3.0635409	0	0	0	0	0
25	1.0524	6.89152024	0	0	261.87	-12.039	0
26	0.99293	8.50897632	0	0	279.94	-49.392	0
27	1.054	-5.9530176	62	13	0	0	0
28	1.071	-7.6472498	17	7	0	0	0
29	1.0973	-8.7473262	24	4	0	0	0
30	0.99504	-1.3408327	0	0	0	0	0
31	1.1093	-8.7977464	43	27	0	0	0
32	1.0461	-8.5087953	59	23	0	0	0
33	0.99912	-9.9660046	33	9	0	0	0
34	1.0075	-11.186975	59	26	0	0	14
35	1.0034	-11.622995	33	9	0	0	0
36	1.0027	-11.615546	31	17	0	0	0
37	1.0145	-10.765332	0	0	0	0	-25
38	0.97362	-5.9650497	0	0	0	0	0

39	0.99776	-14.81207	27	11	0	0	0
40	1.0003	-16.297746	20	23	0	0	0
41	0.99568	-17.017953	37	10	0	0	0
42	1.0087	-16.32754	37	23	0	0	0
43	0.9929	-12.671505	18	7	0	0	0
44	0.98785	-12.368984	16	8	0	0	10
45	0.9845	-11.330787	53	22	0	0	10
46	0.9941	-9.6543163	28	10	0	0	10
47	1.0025	-7.6873568	34	0	0	0	0
48	1.0153	-7.3006112	20	11	0	0	15
49	1.0216	-6.0601604	87	30	301.61	90.052	0
50	1.0015	-8.1898396	17	4	0	0	0
51	0.9719	-10.926853	17	8	0	0	0
52	0.96293	-11.922078	18	5	0	0	0
53	0.95515	-13.009549	23	11	0	0	0
54	0.96624	-12.206837	113	32	0	0	0
55	0.96578	-12.29851	63	22	0	0	0
56	0.96652	-12.156417	64	18	0	0	0
57	0.97801	-10.86039	12	3	0	0	0
58	0.96729	-11.742743	12	3	0	0	0
59	1.0225	-6.9448052	277	113	246.51	160.02	0
60	1.0107	-3.7738923	78	3	0	0	0
61	1.0118	-2.9344538	0	0	176.11	-61.257	0
62	1.0108	-3.5785141	77	14	0	0	0
63	0.98837	-4.3306341	0	0	0	0	0
64	0.99762	-2.9068373	0	0	0	0	0
65	1.0106	-0.7625477	0	0	351.26	-160.96	0
66	1.0425	0.18494461	39	18	389.28	101.36	0
67	1.0217	-2.339152	28	7	0	0	0
68	1.0211	-2.0690031	0	0	0	0	0
69	0.95323	-4.9893048	0	0	0	0	0
70	0.95459	-9.4761268	66	20	0	0	0
71	0.96045	-8.9369748	0	0	0	0	0
72	0.97012	-5.9266616	0	0	0	0	0
73	0.96447	-8.9822383	0	0	0	0	0
74	0.92124	-11.335371	68	27	0	0	12
75	0.92921	-10.237586	47	11	0	0	0
76	0.91488	-11.043736	68	36	0	0	0
77	0.99713	-5.3500382	61	28	0	0	0
78	0.99662	-5.592055	71	26	0	0	0
79	1.0063	-5.146123	39	32	0	0	20
80	1.0488	-2.5181436	130	26	360.32	211.11	0
81	1.0103	-2.2053094	0	0	0	0	0
82	0.98111	-3.1686784	54	27	0	0	20
83	0.97665	-1.787395	20	10	0	0	10
84	0.97227	1.05911001	11	7	0	0	0
85	0.97796	2.76812452	24	15	0	0	0
86	0.98508	0.97247899	21	10	0	0	0
87	1.0242	0.66199389	0	0	0	0	0
88	0.97926	6.32830405	48	10	0	0	0
89	0.99681	10.7114209	0	0	490.47	-9.3697	0
90	0.99223	7.18716578	78	42	0	0	0
91	0.98304	6.77578304	0	0	0	0	0
92	0.98135	5.92666157	65	10	0	0	0
93	0.97654	2.92608862	12	7	0	0	0
94	0.98075	0.79297173	30	16	0	0	0
95	0.97236	-0.9720206	42	31	0	0	0
96	0.9864	-2.1161574	38	15	0	0	0
97	1.0125	-2.6648778	15	9	0	0	0

98	1.0238	-1.9210084	34	8	0	0	0
99	1.0094	1.0723453	0	0	0	0	0
100	1.0056	2.41730328	37	18	201.69	45.243	0
101	0.9818	3.01478228	22	15	0	0	0
102	0.98046	4.8500191	5	3	0	0	0
103	1.0123	2.35708556	23	18	177.72	34.287	0
104	0.97638	-1.1494652	38	25	0	0	0
105	0.97373	-1.937796	31	26	0	0	20
106	0.96899	-2.4741406	43	16	0	0	0
107	0.96709	-3.8332506	28	12	0	0	6
108	0.98208	-2.9021963	2	1	0	0	0
109	0.98574	-3.2547364	8	3	0	0	0
110	0.99907	-3.8438503	39	30	0	0	6
111	1.0058	-3.9568946	0	0	0	0	0
112	1.0161	-5.2656417	25	13	0	0	0
113	1.0382	-5.8051948	0	0	0	0	0
114	1.0449	-6.7517189	8	3	0	0	0
115	1.0451	-6.7517189	22	7	0	0	0
116	1.0219	-2.0724408	0	0	0	0	0
117	0.98613	-2.7859435	20	8	0	0	0
118	0.91577	-11.111345	33	15	0	0	0
Total			**3678**	**1438**	**3935.24**	**387.63**	**88**

5. Conclusion

A GA solution to the OPF problem has been presented and applied to different size power systems. The advantage of the GA lies in its ability to handle any type of unit characteristic data whether smooth or not. Avoiding the repeated solution of the load-flow equations is the main advantage of the proposed solution algorithm. The unique chromosome encoding presented in this paper improves execution time substantially. The mathematical derivation of the effects of the transformer taps on the Jacobian saves execution time by avoiding the recomputation of the entire matrix. By using linear algebra's nullspace theory to reduce the search space that must be examined, the algorithm spends less time evaluating illegal solutions.

A computer program, written in Matlab environment, is developed to represent the proposed method. The program is applied to the IEEE 30-bus test system, and the results are compared with traditional OPF methods to demonstrate its applicability and accuracy. The program is then applied to the IEEE 118-bus test system to show its potential to be used with larger systems. The proposed algorithm is shown to be a valid tool to perform the optimal power flow optimization problem.

6. References

[1] D. Walters, G. Sheble, *"Genetic Algorithm Solution for Economic Dispatch with Valve Point Loading"*, IEEE Trans. on Power Systems, Vol.8, No.3, Aug. 1993.

[2] C. A. Gross, *"Power System Analysis"*, Second Edition, New York: John Wiley & Sons, 1986.

[3] A. Bakirtzis, et al., *"Optimal Power Flow by Enhanced Genetic Algorithm"*, IEEE Trans. on Power Systems, Vol. 17, No. 2, May 2002.

[4] G. B. Sheble, K. Brittig, *"Refined Genetic Algorithm-Economic Dispatch Example"*, IEEE Trans. on Power Systems, Vol. 10, No. 1, February 1995.

[5] P. H. Chen, H. C. Chang, *"Large-Scale Economic Dispatch by Genetic Algorithm"*, IEEE Transactions on Power Systems, Vol. 10, No. 4, November 1995.

[6] J. Tippayachai, W. Ongsakul, and I. Ngamroo, *"Parallel Micro Genetic Algorithm for Constrained Economic Dispatch"*, IEEE Trans. on Power Systems, Vol. 17, No. 3, August 2002.

[7] I. Damousis, et al. *"Network Constrained Economic Dispatch Using Real-Coded Genetic Algorithm"*, IEEE Trans. on Power Systems, Vol. 18, No. 1, February 2003.

[8] O. Alsac, B. Stott, *"Optimal Load Flow with Steady-State Security"*, IEEE Trans. on Power Apparatus and Systems, Vol. PAS-93, No. 3, May-June 1974.

[9] Y. Wallach, *"Calculations and Programs for Power System Networks"*, Prentice-Hall, Inc., Englewood Cliff New Jersey 1986.

[10] Washington University archive, web address: www.ee.washington.edu

[11] P. Venkatesh, et al. *"Comparison and Application of Evolutionary Programming Techniques to Combined Economic Emission Dispatch With Line Flow Constraints"*, IEEE Trans. on Power Systems, Vol. 18, No. 2, May 2003.

. *Appendices*

.1 Appendix (A), IEEE 30-bus System Data

Table A-1 Generator Data, 30-bus System*

Bus	P_{min}	P_{max}	Q_{min}	Q_{max}	a	b	c
1	0.50	2.00	-0.2	2.5	0	2.00	0.00375
2	0.20	0.80	-0.20	1.00	0	1.75	0.0175
5	0.15	0.50	-0.15	0.80	0	1.00	0.0625
8	0.10	0.35	-0.15	0.60	0	3.25	0.0834
11	0.10	0.30	-0.10	0.50	0	3.00	0.0250
13	0.12	0.40	-0.15	0.60	0	3.00	0.0250

All power data are in per-unit, with a base of 100 MVA.

7.2 Appendix (B), IEEE 118-bus System Data

Table B-1 Generator Data, 118-bus System

Bus	P_{min}	P_{max}	Q_{min}	Q_{max}	A	b	c
1	1.0	7.0	-3.0	3.0	150	1.89	0.0050
10	1.0	5.5	-1.47	2.0	115	2.00	0.0055
12	0.1	3.5	-0.35	1.2	40	3.50	0.0060
25	0.5	3.5	-0.47	1.4	122	3.15	0.0055
26	1.0	4.5	-10.0	10.0	125	3.05	0.0050
49	0.5	3.5	-0.85	2.1	120	2.75	0.0070
59	0.5	3.0	-0.6	1.8	70	3.45	0.0070
61	0.5	3.0	-1.0	3.0	70	3.45	0.0070
65	0.5	5.0	-0.67	2.0	130	2.45	0.0050
66	0.5	5.0	-0.67	2.0	130	2.45	0.0050
80	0.5	5.5	-1.65	2.8	135	2.35	0.0055
89	1.0	8.0	-2.1	3.0	200	1.60	0.0045
100	0.5	3.5	-5.0	1.55	70	3.45	0.0070
103	0	2.0	-0.6	0.6	45	3.28	0.0060

Table B-2 Limits on Static VAR Compensation, 118-bus System

Bus	Q_{cMin}	Q_{cMax}
4	-3.0	3.0
6	-0.6	0.6
15	-0.1	0.3
18	-0.6	0.6
19	-0.6	0.6
24	-3.0	3.0
27	-3.0	3.0
31	-3.0	3.0
32	-0.6	0.6
34	-0.6	0.6
36	-0.6	0.6
40	-3.0	3.0
42	-3.0	3.0
46	-1.0	1.0
54	-3.0	3.0
55	-0.6	0.6
56	-0.6	0.6
62	-0.2	0.2
69	-0.6	0.6
70	-0.6	0.6

Bus	Q_{cMin}	Q_{cMax}
72	-1.0	1.0
73	-1.0	1.0
74	-0.6	0.6
76	-0.6	0.6
77	-0.2	0.7
85	-0.6	0.6
87	-1.0	10.0
90	-3.0	3.0
91	-1.0	1.0
92	-0.6	0.6
99	-1.0	1.0
104	-0.6	0.6
105	-0.6	0.6
107	-2.0	2.0
110	-0.6	0.6
111	-1.0	10.0
112	-1.0	10.0
113	-1.0	2.0
116	-10.0	10.0