A Genetic Algorithm Approach Employing Floating Point Representation for Economic Dispatch of Electric Power Systems

Ching-Tzong Su **Wen-Tsuan Tyen**
Institute of Electrical Engineering, National Chung Cheng University, Chiayi 621, TAIWAN
Fax : 886-52752013 **Email : ieects@ccunix.ccu.edu.tw**

Abstract- This paper presents a genetic algorithm (GA) approach employing floating point representation for economic dispatch of electric power systems. The objective of economic dispatch is to adjust all running generators to meet the load demand, such that the cost of generation is minimized. Floating point representation of genetic algorithm may experiment with real-coded genes together with several operators. Like that of employing binary representation, GA employing floating point representation has three primary operators : selection, crossover, and mutation. All of them move genetic algorithm closer to the optimal solution. To demonstrate the applicability an effectiveness of the proposed method, an application example comprising three generators in a simple six-bus power system is illustrated. Network power losses and some nonlinear constraints are considered in the system model. First results of the research show that the proposed method is effective in reaching an optimal solution which is superior to that of the Lambda iteration method in power loss and total generation cost.

Keywords: Power Systems, Economic Dispatch, Genetic Algorithm, Floating point

1. INTRODUCTION

The operation of power systems is characterized by having to maintain a high degree of economy and reliability. Among the options that are available to the engineers in choosing how to operate a power system, Economic dispatch (ED) is the most significant choice. Classical economic dispatch techniques included Lagrangian function method, dynamic programming method, and lambda iteration method. Among these methods, the lambda iteration method is used frequently by power utilities due to its ease of implementation (Sheble and Brittig [1995]).

This paper gives a Genetic algorithm solution to ED problem. The power of GA based on the mechanics of nature and natural genetics, stems from its ability to exploit historical information structure from previous solution guesses in an attempt to improve performance of future solutions. One of the advantages of GA is that it uses stochastic operators instead of deterministic rules to search space, another is that GA searches for many optimum points in parallel.

The binary representation traditionally used in genetic algorithm has some drawbacks when applied to multidimensional, high - precision numerical problems. For example, for 100 variables with domains in the range [-500,500] where a precision of six digits after the decimal point is required, the length of the binary solution vector is 3000. This , in turn, generates a search space of about 100^{1000}. For such problems genetic

algorithms perform poorly.

Numerical optimization has been argued by Michalewicz [1996]. The floating point representation of genetic algorithm used in this paper, which may experiment with real - coded genes together with several operators, can avoid the aforementioned problems and provide a satisfactory application to economic dispatch of electric power systems.

2. ECONOMIC DISPATCH DESCRIPTIONS

The object of economic dispatch is to supply a given demand of power at a minimum generation cost subject to various constraints. This cost is the sum of the individual plant generation costs. The economic dispatch problem is a kind of constrained optimization problem, which can be formulated as follows (Chenand et al. [1995])

$$\text{Minimize } F = \sum_{i=1}^{n} F_i(P_i) \qquad (1)$$

$$\text{Subject to } \sum_{i=1}^{n} P_i - P_{loss} = P_D \;, \quad P_{i(\min)} \le P_i \le P_{i(\max)} \qquad (2)$$

where

F : total generation cost of the system

P_i : power generation of unit i

$F_i(P_i)$: generation cost of unit i at generation P_i

n : total number of units

P_{loss} : system transmission losses

P_D : system load demand.

$P_{i(min)}$: minimum generation of unit i

$P_{i(max)}$: maximum generation of unit i

In (1), the generation cost function $F_i(P_i)$ is usually expressed as a quadratic polynomial function :

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i \qquad (3)$$

where

a_i, b_i, c_i : cost coefficients of unit i.

Since power stations are usually spread out geographically, the transmission network losses must be taken into account to achieve true economic dispatch. In the B-coefficient method, network losses are expressed as a quadratic function :

$$P_{loss} = \sum_{i=1}^{n} \sum_{j=1}^{n} P_i B_{ij} P_j \qquad (4)$$

Where

B_{ij} : loss coefficients

3. GENETIC ALGORITHM MODELING

Genetic algorithm represents a class of general-purpose stochastic search techniques which simulates natural inheritance by genetics and Darwinian ' survival of the fittest ' principle. They combine solution evaluation with randomized, structured exchanges of information between solutions to obtain optimality. Recently, genetic algorithm has become a candidate for many optimizations due to its flexibility and efficiency.

Previous efforts at GA have applied with binary representation, binary chromosome, and binary operators. But calculating the objective function for the problem uses the decoded chromosome set. According to multidimensional and high-precision problems, binary representation may increase chromosome length, which becomes heavier calculation time and memory allocation, especially, introduces various constraints. In the floating point representation GA using real-coded genes together with special genetic operators avoids chromosome encoding and decoding processes, and can easily incorporate various constraints. Overview of GA operators show that the floating point representation of GA has three basic operators : selection, crossover, and

mutation. These operators may be stated as follows (Michalewicz [1996]) :

Selection

Selection is simply an operator whereby an old chromosome is copied into a "mating pool" according to its fitness value. More highly fitted chromosomes receive a higher number of copies in the next generation. By this selection processing it eventually reaches a near-optimal solution with a high probability. This study applies a spinning roulette wheel method, which selects a single chromosome in a floating way :

(1) Calculate the fitness value $fitness(v_i)$ for each chromosome v_i (i=1,..., n_P).

(2) Find the total fitness of the population

$$S = \sum_{i=1}^{n_P} fitness(v_i)$$

(3) Calculate the probability of a selection p_i for each chromosome v_i (i=1,..., n_P).

$$p_i = fitness(v_i) / S$$

(4) Calculate a cumulative probability q_i for each chromosome v_i (i=1,..., n_P).

$$q_i = \sum_{j=1}^{i} P_j$$

(5) Generate a random number r_i from the range[0,...,1].

(6) If $r_i < q_1$ then select the first chromosome (v_1), otherwise select the i-th chromosome $v_i (2 \le i \le n_P)$ such that $q_{i-1} < r_i \le q_i$.

where

n_P is population size.

Crossover

Crossover is an extremely important operator for GA, It is a recombination operator to the individuals in the new population. In binary implementation, crossover is the process of choosing a random position in the chromosome and swapping the choice bits either left or right to generate a new offspring. In floating point implementation, many crossover operators are used to enhance the ability of searching optimal space, for example, whole arithmetical crossover, simple arithmetical crossover, and heuristic crossover are applied. We further detail these operators in the following :

(1) Whole Arithmetical Crossover :

This operator is defined as a linear combination of two genes : if genes x_1 and x_2 are to be crossed, the resulting offsprings are $x_1^o = a \cdot x_1 + (1-a) \cdot x_2$ and $x_2^o = a \cdot x_2 + (1-a) \cdot x_1$, where random value $a \in [0,...,1]$.

(2) Simple Arithmetical Crossover :

This operator is defined as follows : if $x_1 = (x_1,...,x_n)$ and $x_2 = (y_1,...,y_n)$ are crossed after the k-th position, the resulting offsprings are :

$$x_1^o = (x_1,...,x_k, y_{k+1},...,y_n)$$

and

$$x_2^o = (y_1,...,y_k, x_{k+1},...,x_n).$$

Such an operator may produce offspring outside the search domain. To avoid this, we use the property of convex spaces, which states that there exists a constant $a \in [0,...,1]$ such that

$x_1^o = < x_1,...,x_k, y_{k+1} \cdot a + x_{k+1} \cdot (1-a),..., y_n \cdot a + x_n \cdot (1-a) >$ and

$x_2^o = < y_1,...,y_k, x_{k+1} \cdot a + y_{k+1} \cdot (1-a),..., x_n \cdot a + y_n \cdot (1-a) >$ are feasible.

(3) Heuristic Crossover :

This operator generates a single offspring x_3 from two parents x_1 and x_2 according to the following rule :

$x_3 = r \cdot (x_1 - x_2) + x_2$, where r is a random number between 0 and 1, and the parent x_2 is not worse than x_1,i.e., $fitness(x_2) \geq fitness(x_1)$ for maximization problems and $fitness(x_2) \leq fitness(x_1)$ for minimization problems.

Mutation

Although selection and crossover effectively search and recombine existing chromosomes. They do not create any new genetic material in the search space. Mutation is capable of overcoming this shortcoming, which randomly changes the values at one or more chromosomes in order to search for unexplored space. In floating point implementation, there are three operators to be used, they are uniform mutation, boundary mutation, and non-uniform mutation. We further detail these operators in the following :

(1) Uniform Mutation :

This operator requires a single parent x and produces a single offspring $x^{o'}$. The operator selects a random component $k \in (1,...,n)$ of the vector $x = (x_1,...,x_k,...,x_n)$ and produces $x^{o'} = (x_1,...,x_k^o,...,x_n)$. The operator plays an important role in the early phases of the evolution process as the solutions are allowed to more freely within the search space.

(2) Boundary Mutation :

This operator requires also a single parent x and produces a single offspring x^o. The operator is a variation of uniform mutation with x^o being either maximum or minimum, each with equal probability. The operator is constructed for optimization problems where the optimal solution lies either on or near the boundary of the feasible search space.

(3) Non - uniform Mutation :

This is the operator responsible for the fine tuning capabilities of the system. It is defined as follows. For a parent $x = (x_1,...,x_k,...,x_n)$, if the element x_k was selected for this mutation, the result is $x^o = (x_1,...,x_k^o,...,x_n)$, where

$x_k^o = x_k + a(t, max - x_k)$, if a generated random binary digit is 0. Otherwise $x_k^o = x_k + a(t, x_k - min)$, if a generated random binary digit is 1.

The function $a(t, y)$ returns a value in the range $[0,...,y]$ such that the probability of $a(t, y)$ being close to 0 increases(t is the evaluation number). This property causes this operator to search the space uniformly initially (when t is small) and very locally at later stages.

4. MAIN COMPUTATIONAL PROCEDURES :

First, we input all data required for the computation, which include generation cost function, domain constraint, non-linear network losses constraint, operators probabilities, and an evaluation counter for stopping criteria. Then initialize search population and reference population sizes. In each evaluation, chromosomes with best fitness values are copied from search population into reference population. In selection process, spinning roulette wheel method is used to generate offsprings. Mutation and crossover are employed respectively to search optimal solution space. All the offsprings produced in each evaluation must satisfy the prescribed constraints. The flow chart of Figure 1 states the main computational procedures the proposed method.

5. EXAMPLE AND RESULTS

A simple system with three thermal units (Wood and Wollenberg [1996]) is used to demonstrate how the proposed approach works. The unit data are given in Table 1.

Table 1 Unit data

Units	$P_{i(min)}$ (MW)	$P_{i(max)}$ (MW)	a_i ($/ MW^2)	b_i ($/ MW)	c_i ($)
1	50	250	0.00525	8.663	328.13
2	5	150	0.00609	10.04	136.91
3	15	100	0.00592	9.76	59.16

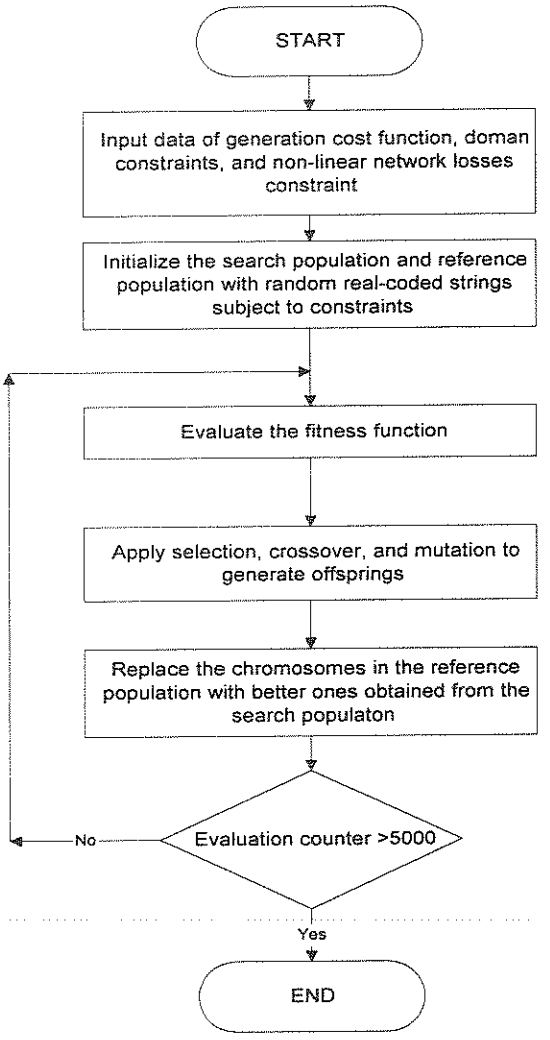

Figure 1 Main computational procedures for the proposed method

The loss coefficients are :

$$B_{ij} = \begin{bmatrix} 0.000136 & 0.0000175 & 0.000184 \\ 0.0000175 & 0.000154 & 0.000283 \\ 0.000184 & 0.000283 & 0.00161 \end{bmatrix}$$

Load demand = 300 MW.

Parameters used for GA :

search population size = 20

reference population size = 5

whole arithmetical crossover probability = 0.0667

simple arithmetical crossover probability = 0.06667

heuristic crossover probability = 0.5556

uniform mutation probability = 0.06667

boundary mutation probability = 0.06667

non-uniform mutation probability = 0.06667

maximum evaluation number = 5000

Computational results from the proposed method are shown in Table 2. For comparison, results from the typical economic dispatch method of lambda iteration are also listed in Table 2. It can be seen that the unit power generations from the two methods are very close, and the total generation cost of the proposed method is slightly smaller than that of the lambda iteration method.

Figure 2 shows the variation of total generation cost with respect to evaluation number. It is observable that the proposed method has excellent convergent characteristics.

Table 2 Computational results from the proposed and lambda iteration methods

Methods	P_1 (MW)	P_2 (MW)	P_3 (MW)	P_{loss} (MW)	Total generation cost ($)
the proposed	207.581	87.338	15.000	9.969	3619.756
Lambda iteration	207.7	87.3	15.0	10.0	3619.95

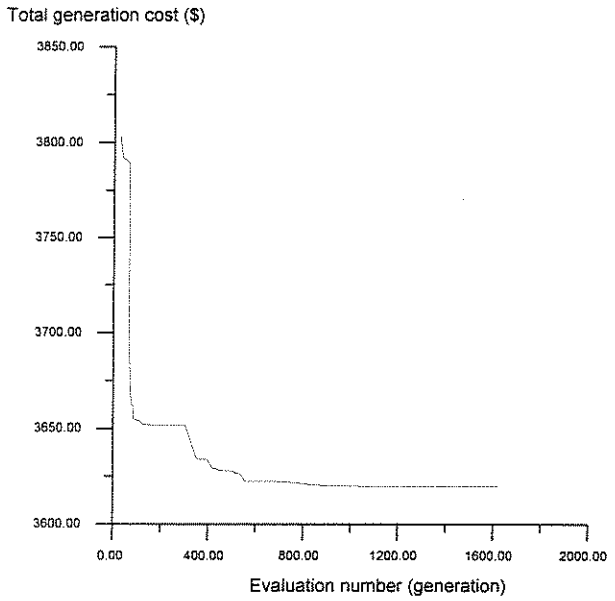

Figure 2 Variation of total generation cost with evaluation number

6. CONCLUSIONS

A new optimal economic dispatch method that uses Genetic Algorithm employing floating point representation for electric power systems is presented. In comparison with the lambda iteration method, the proposed method has better solution Floating point representation of GA uses many heuristic operators, parallel search of many optimum points , and probabilistic rules. Which enables the proposed approach to become a robust and global optimization algorithm. Moreover, this approach can also take network losses into account to make the dispatch more practical.

7. REFERENCES

Chenand, P.H., and H.C. Chang, Large- scale economic dispatch by genetic algorithm *IEEE Trans. on Power Systems*, Vol. 10, No. 4, pp.1919-1925, 1995.

Michalewicz,Z., *Genetic Algorithms + Data Structures =Evolution Programs*. Springer-Verlag, New York, 1996.

Sheble, G.B., and K. Brrittig, Refined genetic algorithm-economic dispatch example *IEEE Trans. on Power Systems*, Vol. 10, No. 1, pp.117-124, 1995.

Wood, A.J., and B.F. Wollenberg, *Power Generation, Operation and Control,* John Wiley & Sons, 569 pp., New York,1996.