Improved Genetic Algorithm for Economic Load Dispatch with Valve-Point Loadings

S.H. Ling, H.K. Lam, F.H.F. Leung, and Y.S. Lee

Centre for Multimedia Signal Processing,
Department of Electronic and Information Engineering, the Hong Kong Polytechnic University

***Abstract*— Economic load dispatch is one of the optimization problems in power systems. This paper presents an improved genetic algorithm for economic load dispatch with valve-point loadings. New crossover and mutation operations are introduced. The solutions of the economic load dispatch with valve-point loadings under three cases are solved by the improved genetic algorithm. Test results are given and compared with those from different published genetic algorithms. It will be shown that the proposed improved genetic algorithm performs better than the published genetic algorithms.**

***Index Terms*—Economic load dispatch (ELD), genetic algorithm (GA).**

I. INTRODUCTION

Engineers always concern the cost of products or services. In a power system, minimizing the operation cost is very important. Economic load dispatch (ELD) is a method to schedule the power generator outputs with respect to the load demands, and to operate a power system most economically. The input-output characteristics of modern generators are nonlinear by nature because of valve-point loadings and rate limits. The problem of ELD is multimodal, discontinuous and highly nonlinear. Global search techniques had been employed to solve the ELD problems [1-5, 9-11]. These techniques include evolutionary programming [1-3], Tabu search [4] and genetic algorithm [5, 9-11].

Genetic algorithm (GA) is one of the global search techniques especially useful for the complex optimization problems with a large number of parameters of which an analytical solution is difficult to obtain. Traditional GA [6] has some drawbacks when applying to multidimensional, high-precision numerical problems. Different genetic operations have been proposed to improve the efficiency of GA. Genetic operations usually refer to the operation of crossover and mutation. While random mutation and crossover are widely used, different modifications to the crossover and mutation operations have been reported. For the crossover operation, arithmetic crossover, heuristic crossover and simple crossover were proposed [6-8]. For the mutation operation, uniform mutation and non-uniform mutation can be found [6-8].

In this paper, an improved genetic algorithm for economic load dispatch is proposed. New genetic operations of crossover and mutation are introduced. On realizing the crossover operation, the offspring spreads over the domain so that a higher chance of reaching the global optimum can be obtained. On realizing the mutation, each gene will have a chance to change its value. The search domain of the selected gene will be contracted at a rate governed by a monotonic decreasing function. Consequently, the search domain will become smaller when the training iteration number increases in order to realize a fine-tuning process. By introducing these genetic operations, the proposed GA performs more efficiently and provides a faster convergence than the published GAs.

This paper is organized as follows. Section II presents the economic load dispatch with valve-point loadings problem. Section III presents the improved genetic algorithm. The modified genetic operations will be introduced. Economic load dispatch using the proposed GA will be discussed in section IV. An application example on economic load dispatch is given in session V. A conclusion will be drawn in section VI.

II. ECONOMIC LOAD DISPATCH WITH VALVE-POINT LOADINGS PROBLEM

The economic load dispatch with valve-point loading problem can be formulated into the following objective function:

$$Min \sum_{i=1}^{n} C_i(P_{L_i}), \tag{1}$$

where $C_i(P_{L_i})$ is the operation fuel cost of generator i, and n denotes the number of generator. The problem is subject to balance constraint and generating capacity constraints as follows:

$$D = \sum_{i=1}^{n} P_{L_i} - P_{Loss}, \tag{2}$$

$$P_{L_{i,\min}} \le P_{L_i} \le P_{L_{i,\max}}, \; i = 1, ..., n, \tag{3}$$

where D is the load demand, P_{L_i} is the output power of the i-th generator, P_{Loss} is the transmission loss, $P_{L_{i,\max}}$ and $P_{L_{i,\min}}$ are the maximum and minimum output powers of the i-th generator respectively.

The operation fuel cost function with valve-point loadings of the generators is given by,

$$C_i(P_{L_i}) = a_i P_{L_i}^2 + b_i P_{L_i} + c_i + \left| e_i \times \sin\left(f_i \times \left(P_{L_{i,\min}} - P_{L_i}\right)\right)\right|, \tag{4}$$

0-7803-7906-3/03/$17.00 ©2003 IEEE.

where a_i, b_i, and c_i, are coefficients of the cost curve of the i-th generator, e_i and f_i are coefficients of the valve-point loadings. (The generating units with multivalve steam turbines exhibit a greater variation in the fuel-cost functions. The valve-point effects introduce ripples in the heat-rate curves.)

III. Improved Genetic Algorithm

The published GA process [6-8] is shown in Fig. 1. First, a population of chromosomes is created. Second, the chromosomes are evaluated by a defined fitness function. Third, some of the chromosomes are selected to undergo genetic operations. Forth, genetic operations of crossover (arithmetic, heuristic or one-point crossover, depending on which one gives the best result in each iteration) and non-uniform mutation are performed. The produced offspring replaces the chromosome with the smallest fitness value in the initial population, and forms a new population. This GA process repeats until a user-defined criterion is reached. In this paper, new genetic operations are introduced to improve the performance. The modified GA process is shown in Fig. 2. Its details are given as follows.

A. Initial Population

The initial population is a potential solution set P. The first set of population is usually generated randomly.

$$P=\left\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_{pop\_size}\right\}, \tag{5}$$

$$\mathbf{p}_i=\left[p_{i_1} \quad p_{i_2} \quad \cdots \quad p_{i_j} \quad \cdots \quad p_{i_{no\_vars}}\right], i = 1, 2, \ldots, pop\_size;$$
$$j = 1, 2, \ldots, no\_vars; \tag{6}$$

$$para^j_{\min} \le p_{i_j} \le para^j_{\max}, \tag{7}$$

where pop_size denotes the population size; no_vars denotes the number of variables to be tuned; p_{i_j}, i = 1, 2, ..., pop_size; j = 1, 2, ..., no_vars, are the parameters to be tuned; $para^j_{\min}$ and $para^j_{\max}$ are the minimum and maximum values of the parameter p_{i_j} respectively for all i. It can be seen from (5) to (7) that the potential solution set P contains some candidate solutions $\mathbf{p}_i$ (chromosomes). The chromosome $\mathbf{p}_i$ contains some variables p_{i_j} (genes).

B. Evaluation

Each chromosome in the population will be evaluated by a defined fitness function. The better chromosomes will return higher values in this process. The fitness function can be written as,

$$fitness = f(\mathbf{p}_i). \tag{8}$$

The form of the fitness function depends on the application.

```
Procedure of the published GA
begin
    τ → 0.                    // τ: iteration number
    Initialize P(τ).          //P(τ): population for iteration τ
    Evaluate f(P(τ)).         // f(P(τ)):fitness function
while (not termination condition) do
    begin
        τ → τ + 1.
        Select 2 parents p1 and p2 from P(τ − 1).
        Perform genetic operations (crossover and mutation).
        Reproduce a new P(τ).
        Evaluate f(P(τ)).
    end
end
```

Fig. 1. Published GA process.

```
Procedure of the improved GA
begin
    τ → 0.                    // τ: iteration number
    Initialize P(τ).          //P(τ): population for iteration τ
    Evaluate f(P(τ)).         // f(P(τ)):fitness function
while (not termination condition) do
    begin
    τ → τ + 1.
    Select 2 parents p1 and p2 from P(τ − 1).
    Perform crossover operation according to equations (11)-(14).
    Perform mutation operation according to equation (18).
    Select the largest offspring between that after crossover and
    mutation processes as the actual offsproing os.
    // reproduce a new P(τ)
        if random number < pa   // pa: probability of
        acceptance
            os replaces the chromosome with the smallest
            fitness value in the population.
        else if f(os) > smallest fitness value in the P(τ − 1)
            os replaces the chromosome with the smallest
            fitness value.
        end
    Evaluate f(P(τ)).
    end
end
```

Fig. 2. Improved GA process.

C. Selection

Two chromosomes in the population will be selected to undergo genetic operations for reproduction by the method of spinning the roulette wheel [6]. It is believed that high potential parents will produce better offspring (survival of the best ones). The chromosome having a higher fitness value should therefore have a higher chance to be selected. The selection can be done by assigning a probability q_i to the chromosome $\mathbf{p}_i$:

$$q_i = \frac{f(\mathbf{p}_i)}{\sum_{k=1}^{pop\_size} f(\mathbf{p}_k)}, i = 1, 2, \ldots, pop\_size. \tag{9}$$

The cumulative probability $\hat{q}_i$ for the chromosome $\mathbf{p}_i$ is defined as,

$$\hat{q}_i = \sum_{k=1}^{i} q_k, i = 1, 2, \ldots, pop\_size. \tag{10}$$

The selection process starts by randomly generating a nonzero floating-point number $d \in [0 \quad 1]$. Then, the chromosome $\mathbf{p}_i$ is chosen if $\hat{q}_{i-1} < d < \hat{q}_i$, i = 1, 2, ..., *pop_size*, and $\hat{q}_0 = 0$. In this way, a chromosome having a larger $f(\mathbf{p}_i)$ will have a higher chance to be selected. Consequently, the best chromosomes will get more offspring, the average will stay and the worst will die off. The process is repeated so that two chromosomes are selected to undergo the genetic operations.

D. Genetic Operations

The genetic operations generate some new chromosomes (offspring) from their parents after the selection process. They include the crossover and the mutation operations.

1) Crossover: The crossover operation is mainly for exchanging information from the two parents, chromosomes $\mathbf{p}_1$ and $\mathbf{p}_2$, obtained in the selection process. The two parents will produce one offspring. The details of the crossover operation are as follows. First, four chromosomes will be generated according to the following formulae,

$$\mathbf{o}_{s_c^1} = \left[o_{s_1^1} \quad o_{s_2^1} \quad \cdots \quad o_{s_{no\_vars}^1} \right] = \frac{\mathbf{p}_1 + \mathbf{p}_2}{2}, \tag{11}$$

$$\mathbf{o}_{s_c^2} = \left[o_{s_1^2} \quad o_{s_2^2} \quad \cdots \quad o_{s_{no\_vars}^2} \right] = \mathbf{p}_{\max}(1-w) + \max(\mathbf{p}_1, \mathbf{p}_2) w \quad , \tag{12}$$

$$\mathbf{o}_{s_c^3} = \left[o_{s_1^3} \quad o_{s_2^3} \quad \cdots \quad o_{s_{no\_vars}^3} \right] = \mathbf{p}_{\min}(1-w) + \min(\mathbf{p}_1, \mathbf{p}_2) w \quad , \tag{13}$$

$$\mathbf{o}_{s_c^4} = \left[o_{s_1^4} \quad o_{s_2^4} \quad \cdots \quad o_{s_{no\_vars}^4} \right] = \frac{(\mathbf{p}_{\max} + \mathbf{p}_{\min})(1-w) + (\mathbf{p}_1 + \mathbf{p}_2) w}{2} \quad , \tag{14}$$

$$\mathbf{p}_{\max} = \left[para_{\max}^1 \quad para_{\max}^2 \quad \cdots \quad para_{\max}^{no\_vars} \right], \tag{15}$$

$$\mathbf{p}_{\min} = \left[para_{\min}^1 \quad para_{\min}^2 \quad \cdots \quad para_{\min}^{no\_vars} \right], \tag{16}$$

where $w \in [0 \quad 1]$ denotes the weight to be determined by users, $\max(\mathbf{p}_1, \mathbf{p}_2)$ denotes the vector with each element obtained by taking the maximum between the corresponding elements of $\mathbf{p}_1$ and $\mathbf{p}_2$. For instance, $\max([1 \quad -2 \quad 3], [2 \quad 3 \quad 1]) = [2 \quad 3 \quad 3]$. Similarly, $\min(\mathbf{p}_1, \mathbf{p}_2)$ gives a vector by taking the minimum value. For instance, $\min([1 \quad -2 \quad 3], [2 \quad 3 \quad 1]) = [1 \quad -2 \quad 1]$. Among $\mathbf{o}_{s_c^1}$ to $\mathbf{o}_{s_c^4}$, the one with the largest fitness value is used as the offspring of the crossover operation. The offspring after crossover operation is defined as,

$$\mathbf{o}_s = \left[o_{s_1} \quad o_{s_2} \quad \cdots \quad o_{s_{no\_vars}} \right] = \mathbf{o}_{s_c^{i_{os}}} \tag{17}$$

where i_{os} denotes the index i which gives a maximum value of $f\left(\mathbf{o}_{s_c^i}\right)$, i = 1, 2, 3 ,4.

If the crossover operation can provide a good offspring, a higher fitness value can be reached in less iteration. In general, two-point crossover, multipoint crossover, arithmetic crossover or heuristic crossover can be employed to realize the crossover operation [6-8]. However, the offspring generated by these methods may not be better than that of our approach. As seen from (11) to (14), the offspring spreads over the domain: (11) and (14) will move the offspring near the centre region of the concerned domain (as w in (14) approaches 1, $\mathbf{o}_{s_c^4}$ approaches $\frac{\mathbf{p}_1 + \mathbf{p}_2}{2}$), and (12) and (13) will move the offspring near the domain boundary (as w in (12) and (13) approaches 0, $\mathbf{o}_{s_c^2}$ and $\mathbf{o}_{s_c^3}$ approaches $\mathbf{p}_{\max}$ and $\mathbf{p}_{\min}$ respectively).

2) Mutation: The offspring (17) will then undergo the mutation operation, which changes the genes of the chromosomes. Consequently, the features of the chromosomes inherited from their parents can be changed. In general, various methods like boundary mutation, uniform mutation or non-uniform mutation [6-8] can be employed to realize the mutation operation. Boundary mutation is to change the value of a randomly selected gene to its upper- or lower-bound value. Uniform mutation is to change the value of a randomly selected gene to a value between its upper and lower bounds. Non-uniform mutation is capable of fine-tuning the parameters by increasing or decreasing the value of a randomly selected gene by a weighted random number. The weight is usually a monotonic decreasing function of the number of iteration. In this paper, a different process of mutation is proposed. The details are as follows. Every gene of the offspring $\mathbf{o}_s$ of (17) will have a chance to mutate governed by a probability of mutation, $p_m \in [0 \quad 1]$, which is defined by the user. This probability gives an expected number ($p_m \times no\_vars$) of genes that undergo the mutation. For each gene, a random number between 0 and 1 will be generated such that if it is less than or equal to p_m, the operation of mutation will take place on that gene and updated instantly. The gene of the offspring of (17) is then mutated by:

$$\hat{o}_{s_k} = \begin{cases} o_{s_k} + \Delta o_{s_k}^U \text{ if } f(\mathbf{o}_s + \Delta \mathbf{o}_{s_k}^U) \geq f(\mathbf{o}_s - \Delta \mathbf{o}_{s_k}^L) \\ o_{s_k} - \Delta o_{s_k}^L \text{ if } f(\mathbf{o}_s + \Delta \mathbf{o}_{s_k}^U) < f(\mathbf{o}_s - \Delta \mathbf{o}_{s_k}^L) \end{cases}, k = 1, 2, \ldots, no\_vars, \tag{18}$$

where

$$\Delta o_{s_k}^U = w_{m_k} r\left(para_{\max}^k - o_{s_k} \right), \tag{19}$$

$$\Delta o_{s_k}^L = w_{m_k} r\left(o_{s_k} - para_{\min}^k \right), \tag{20}$$

$$\Delta \mathbf{o}_{\mathbf{s}_k}^U = \begin{bmatrix} 0 & 0 & \cdots & \Delta o_{s_k}^U & \cdots & 0 \end{bmatrix}, \quad (21)$$

$$\Delta \mathbf{o}_{\mathbf{s}_k}^L = \begin{bmatrix} 0 & 0 & \cdots & \Delta o_{s_k}^L & \cdots & 0 \end{bmatrix}, \quad (22)$$

$r \in [0 \quad 1]$ is a randomly generated number; $w_{m_k} \in (0 \quad 1]$ is a weight governing the magnitudes of $\Delta o_{s_k}^U$ and $\Delta o_{s_k}^L$. The value of weight w_{m_k} is varied by the value of $\frac{\tau}{T}$, which serves a fine-tuning purpose; τ is the current iteration number, T is the total number of iteration. In order to perform a local search, the value of w_{m_k} becomes small as $\frac{\tau}{T}$ increases in order to reduce the significance of the mutation. Under this assumption, a monotonic decreasing function governing w_{m_k} is proposed to be,

$$w_{m_k} = w_f \left(1 - \frac{\tau}{T}\right)^{\frac{1}{w_\tau}} \geq 0, \quad (23)$$

where $w_f \in [0 \quad 1]$ and $w_\tau > 0$ are both variables to be chosen to determine the initial value and the decay rate respectively. For a large value of w_f, it can be seen from (19) and (20) that $\Delta o_{s_k}^U \approx r(para_{\max}^k - o_{s_k})$ and $\Delta o_{s_k}^L \approx r(o_{s_k} - para_{\min}^k)$ initially as $\left(1 - \frac{\tau}{T}\right)^{\frac{1}{w_\tau}} \approx 1$, which ensure a large search space. When the value of $\left(1 - \frac{\tau}{T}\right)^{\frac{1}{w_\tau}} \approx 0$, it can be seen that the values of $\Delta o_{s_k}^U$ and $\Delta o_{s_k}^L$ are small to ensure a small search space for fine-tuning.

E. Reproduction

After going through the mutation process, the offspring with a larger fitness value between that of (17) and that after mutation will be taken as the actual offspring in the reproduction process. This new offspring will replace the chromosome with the smallest fitness value among the population if a randomly generated number within 0 to 1 is smaller than $p_a \in [0 \quad 1]$, which is the probability of acceptance defined by users. Otherwise, the new offspring will replace the chromosome with the smallest fitness value only if the fitness value of the offspring is greater than the fitness value of that chromosome in the population.

After the above operations, a new population is generated. This new population will repeat the same process to produce another offspring. Such an iterative process can be terminated when a defined condition is satisfied, e.g. a sufficiently large number of iteration has been reached.

F. Choosing the parameters

The GA is effectively seeking a balance between the exploration of new regions and the exploitation of already sampled regions in the search space. This balance, which critically controls the performance of the GA, is governed by the right choices of control parameters: the probability of mutation (p_m), the probability of acceptance (p_a), the population size (*pop_size*), the weight in the crossover operation (w), and the parameters in the mutation operation (w_τ and w_f). Some views about these parameters are included as follows:

- Increasing p_m tends to transform the genetic search into a random search. This probability gives us an expected number ($p_m \times no\_vars$) of genes that undergo the mutation. When p_m = 1, all genes will mutate. The value of p_m depends on the desirable number of genes that undergo the mutation operation.
- Increasing p_a will increase the chance that a poor offspring joins the population. This reduces the probability that the GA prematurely converges to a local optimum. From a p_a of 0.1 is a good enough choice for many optimization problems.
- Increasing the population size will increase the diversity of the search space, and reduce the probability that the GA prematurely converges to a local optimum. However, it also increases the time required for the population to converge to the optimal region in the search space. From experience, a population size of 10 is an acceptable choice.
- Changing the value of w will change the characteristics of the crossover operations. It is chosen by trial and error, which depends on different optimization problem. For example, as w in (14) approaches 1, $\mathbf{o}_{\mathbf{s}_c^4}$ approaches $\frac{\mathbf{p}_1 + \mathbf{p}_2}{2}$.
- Changing the parameters w_τ and w_f will change the characteristics of the monotonic decreasing function of the mutation operation w_{m_k}. The w_{m_k} will perform fine-tuning faster as w_τ is decreasing, and the value w_f is the initial value of w_{m_k}.

IV. Improved GA for ELD

In this section, the improved GA is used to solve the economic load dispatch problem. The chromosome is defined as follows:

$$\mathbf{p} = \begin{bmatrix} P_{L_1} & P_{L_2} & P_{L_3} & \cdots & P_{L_{h-1}} \end{bmatrix}, \quad (24)$$

where n denotes the number of generator and $P_{L_{i,\min}} \leq P_{L_i} \leq P_{L_{i,\max}}$, $i = 1, 2, \ldots, n-1$. From (2), we have,

$$P_{L_n} = D - \sum_{i=1}^{n-1} P_{L_i} + P_{Loss}. \quad (25)$$

In this paper, the power loss is not considered. Therefore,

$$P_{L_n} = D - \sum_{i=1}^{n-1} P_{L_i}. \quad (26)$$

To ensure P_{L_n} falls within the range $\left[P_{L_{n,\min}}, P_{L_{n,\max}}\right]$, the following conditions are considered:

$$\text{if } P_{L_n} > P_{L_{n,\max}} \begin{cases} P_{L_1} = P_{L_1} + \left(P_{L_n} - P_{L_{n,\max}}\right) \\ P_{L_n} = P_{L_{n,\max}} \end{cases}, \quad (27)$$

$$\text{if } P_{L_n} < P_{L_{n,\max}} \begin{cases} P_{L_1} = P_{L_1} - \left(P_{L_{n,\min}} - P_{L_n}\right) \\ P_{L_n} = P_{L_{n,\min}} \end{cases}. \quad (28)$$

It should be noted from (27) and (28) that if the value of P_{L_1} is also outside the constraint boundary. The exceeding portion of the power will be shared by other generators in order to make sure that all generators' output power is within the safety range. Referring to (1), the fitness function for this ELD problem is defined as,

$$fitness = -\sum_{i=1}^{n} C_i(P_{L_i}), \quad (29)$$

where $C_i(P_{L_i})$ is defined in (4). The objective is to maximize the fitness function (29), which minimizes the operation cost function.

V. Simulation Result

In this section, the improved GA (IGA) goes through the ELD problem and the results will be compared with other published GAs with arithmetic crossover and non-uniform mutation (AGA), heuristic crossover and non-uniform mutation (HGA), simple crossover and non-uniform mutation (SGA) [6-8], and AHSGA which choose the best offspring among those produced by arithmetic crossover, heuristic crossover and simple crossover in each iteration. The results are the averaged ones out of 100 runs.

The improved and the published GAs are applied to a 13-generator system, which was adopted as an example in [10]. It is a large system with many local minimum points. The data of the units with valve-point loadings are tabulated in Table I. The load demand (D) is 1800MW in this example. The population size used for all GAs is 10. The probability of crossover and mutation for AGA, HGA, SGA and AHSGA are set at 0.6 and 0.4 respectively through trial and error. The shape parameter of non-uniform mutation is set at 3. For the IGA, the probability of mutation and probability of acceptance is set at 0.4 and 0.1 respectively. The parameters of w, w_r and w_f are set at 1, 0.8 and 0.1 respectively. For all approaches, the number of iteration is 1000. The simulation results over 100 runs are shown in Table II and Fig. 3. It can be seen that the proposed IGA performs better than other published GAs (AGA, HGA, SGA, and AHSGA) in terms of operation cost and standard deviation. The average cost is \$18096.40/h and the minimum cost is \$18063.58/h. The optimal dispatch solution is summarized in Table III. In Fig. 3, it can be seen that the IGA converges much faster than the published GAs. The final dispatch solution over 100 runs is shown in Fig. 4, which reveals the reliability of the GAs. From this figure it can be seen that the IGA produces 70% solutions at or very near to the global optimum. However, the published GAs produces about 5-25% solutions at or very near to the global optimum only. The IGA is therefore more reliable.

TABLE I
UNITS DATA (13 SYSTEMS WITH VALVE-POINT LOADINGS): a ($/MW²h), b ($/MWh), c ($/h), e ($/h) AND f (rad/MW) ARE COST COEFFICIENTS.

Unit	$P_{L_{i,\min}}$ (MW)	$P_{L_{i,\max}}$ (MW)	a	b	c	e	f
1	0	680	0.00028	8.1	550	300	0.035
2	0	360	0.00056	8.1	309	200	0.042
3	0	360	0.00056	8.1	307	200	0.042
4	60	180	0.00324	7.74	240	150	0.063
5	60	180	0.00324	7.74	240	150	0.063
6	60	180	0.00324	7.74	240	150	0.063
7	60	180	0.00324	7.74	240	150	0.063
8	60	180	0.00324	7.74	240	150	0.063
9	60	180	0.00324	7.74	240	150	0.063
10	40	120	0.00284	8.6	126	100	0.084
11	40	120	0.00284	8.6	126	100	0.084
12	55	120	0.00284	8.6	126	100	0.084
13	55	120	0.00284	8.6	126	100	0.084

TABLE II
RESULTS FOR LOAD DEMAND OF 1800MW.

Method	Mean Cost	Max. Cost	Min. Cost	Std. Dev.
AGA	18195.23	18452.63	18078.39	94.620
HGA	18196.95	18529.15	18077.49	90.826
SGA	18233.11	18517.66	18083.29	101.758
AHSGA	18178.92	18385.29	18078.39	77.831
IGA	18096.40	18293.47	18063.58	45.795

TABLE III
THE OPTIMAL DISPATCH SOLUTION.

P_{L_i} (MW)	AGA	HGA	SGA	AHSGA	IGA
P_{L_1}	359.04	486.90	359.04	359.04	269.29
P_{L_2}	224.40	150.10	154.18	224.40	220.27
P_{L_3}	224.40	150.00	225.18	224.40	299.20
P_{L_4}	109.87	109.91	159.74	109.87	159.73
P_{L_5}	109.87	109.95	109.87	109.87	109.84
P_{L_6}	109.87	109.87	109.91	109.87	109.87
P_{L_7}	109.87	109.88	159.74	109.87	109.87
P_{L_8}	109.86	110.10	109.87	109.86	109.87
P_{L_9}	109.87	159.74	109.87	109.87	109.86
$P_{L_{10}}$	77.40	77.88	77.45	77.40	77.40
$P_{L_{11}}$	77.40	77.52	77.40	77.40	77.40
$P_{L_{12}}$	92.40	93.16	92.40	92.40	92.40
$P_{L_{13}}$	85.76	55.00	55.01	85.76	55.00
Total Power (MW)	1800	1800	1800	1800	1800
Total Cost ($/h)	18078.39	18077.49	18083.29	18078.39	18063.58

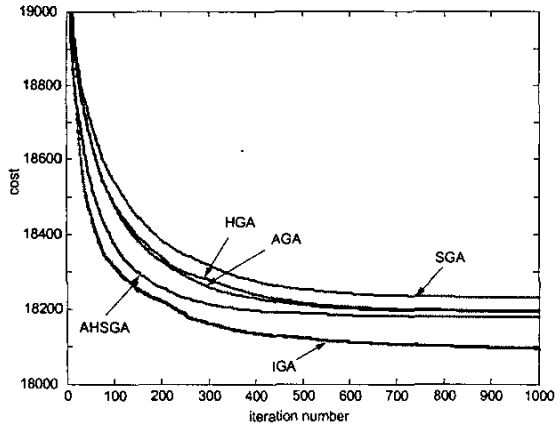

Fig. 3. Simulation results of AGA, HGA, SGA, AHSGA and IGA.

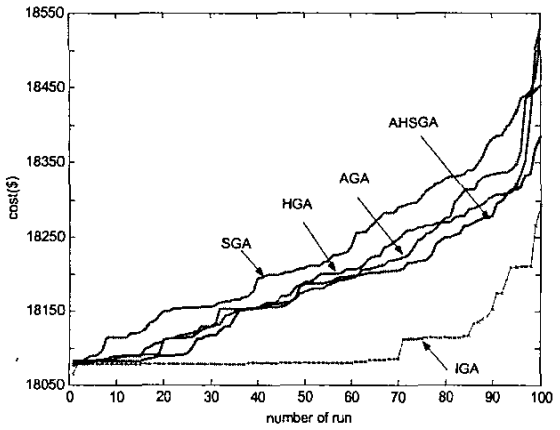

Fig. 4. Final dispatch solutions over 100 runs.

VI. Conclusion

The problem of economic load dispatch with valve-point loadings has been investigated in this paper. An Improved GA has been proposed to help solve this problem. New genetic operations (crossover and mutation) have been introduced. By using the proposed crossover operation, the offspring spreads over the domain so that the probability of reproducing good offspring is increased. In the proposed mutation operation, the search domain of the selected gene will be contracted at a rate governed by a monotonic decreasing function. An economic load dispatch problem has been presented and solved by the improved GA. The results show that the improved GA has performed better, in terms of operation cost, convergence rate and reliability, than some published GAs.

VII. Acknowledgment

The work described in this paper has been substantially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU 5127/00E).

VIII. References

[1] N. Sinha, R. Chakrabarti, and P.K. Chattopadhyay, "Evolutionary programming techniques for economic load dispatch," *IEEE Trans. Evol, Comput.*, vol. 7, no. 1, pp. 83-94, Feb. 2003.

[2] P. Attaviriyanupap, H. Kita, E. Tanaka, and J. Hasegawa, "A hybrid EP and SQP for dynamic economic dispatch with nonsmooth fuel cost function," *IEEE Trans. Power Syst*, vol. 17, no. 2, pp. 411-416, May 2002.

[3] K.P. Wong and J. Yuryevich, "Evolutionary-programming-based algorithm for environmentally-constrained economic dispatch," *IEEE Trans. Power Syst*, vol. 13, no. 2, pp. 301-306, May 1998.

[4] W.M. Lin, F.S. Cheng, and M.T. Tsay, "An improved Tabu search for economic dispatch with multiple minima," *IEEE Trans. Power Syst*, vol. 17, no. 1, pp. 108-112, Feb. 2002.

[5] M. Yoshimi, K.S. Swarup, and Y. Izui, "Optimal economic power dispatch using genetic algorithm," in *Proc. 2nd Int. Forum Applications of Neural Networks to Power Systems, 1993, ANNPS '93*, Arp. 1993, pp. 157-162.

[6] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*, 2nd extended ed. Springer-Verlag, 1994.

[7] L. Davis, *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 991.

[8] M. Srinivas, L.M. Patnaik, "Genetic algorithms: a survey," *IEEE Computer*, vol. 27, issue 6, pp. 17-26, Jun. 1994.

[9] D.C. Walter and G.B. Sheble, "Genetic algorithm solution of economic dispatch with value point loading," *IEEE Trans. Power Syst.*, vol. 8, pp.1325-1332, Aug. 1993.

[10] K.P. Wong and Y.W. Wong, "Thermal generator scheduling using hybrid genetic/simulated annealing approach," *IEE Proc. C*, vol. 142, pp. 372-380, Jul. 1995.

[11] P.H. Chen and H.C. Chang, "Large-scale economic dispatch by genetic algorithm," *IEEE Trans. Power Syst.*, vol. 10, pp. 117-124, Feb. 1995.