Genetic Algorithm for Power Load Dispatch

Chao-Lung Chiang
Depart. of Electronic Engineering
Nan-Kai Institute of Technology
Nan-Tou, TAIWAN, R.O.C.
t129@nkc.edu.tw

Abstract—Few investigations of the genetic algorithm (GA) have been studied for the real-world power economic load dispatch (PELD) problem. This paper proposes an improved genetic algorithm with multiplier updating (IGAMU) to solve practical PELD problems of complexity having nonconvex cost curves where conventional mathematical methods are inapplicable. The proposed IGAMU integrates the improved genetic algorithm (IGA) and the multiplier updating (MU). A practical example is employed to demonstrate that the proposed algorithm has merits of straightforward concept; easy implementation; better effectiveness than previous methods; better effectiveness and efficiency than the GA with MU (GA-MU); automatic adjustment of the randomly assigned penalty to an appropriate value, and the requirement for only a small population in applications of real life PELD operations.

Keywords—economic load dispatch, nonconvex function.

I. INTRODUCTION

The GA has been applied popularly as a useful optimization tool for handling nonlinear programming problems [1]. Various modifications to the basic method have been proposed with a view to enhance speed and robustness, and these have been applied successfully on some benchmark mathematical problems [2]. But few approaches have been reported on the real-world PELD problem, which is one of the important optimizations in a power system for allocating generation among the committed units such that system constraints imposed are satisfied and energy requirements are minimized. Improvements in scheduling the unit outputs can lead to significant cost savings. For simplicity, the generator cost function was mostly approximated by a single quadratic function [3], [4]. However, the whole operating range may not be always available. Units may have prohibited operation zones (POZs) due to faults in machines or associated auxiliaries, such as boilers, feed pumps, etc., leading to instabilities in certain ranges. Take Fig. 1 for example, a power unit with three POZs has a discontinuous input-output fuel cost characteristic. The prohibited region separates the decision space into disjointed subsets, constituting a nonconvex solution space, and the PELD problem becomes a nonsmooth optimization problem having complex and nonconvex characteristics, which makes the challenge of finding the global optimum difficult.

Traditionally, the PELD problem is solved using conventional mathematical based techniques such as the lambda iteration method (λ-δ) [5] and gradient method. These techniques require increasing fuel cost curves should be monotonically increasing to find the global optimal solution. Whereas, the input-output characteristics of units are inherently highly nonlinear because of POZs, thus the traditional methods may generate either multiple local minimum points or infeasible solutions for the PELD problem. Recently, various evolutionary algorithms are proposed for the PELD problem such as simulated annealing (SA) [6], GA [1], [7] and an integrated artificial intelligence (ETQ) [8]. The SA was devoted to solve the high nonlinear PELD problem without restrictions on the shape of the fuel cost function. Nevertheless, it is difficult to tune the related control parameters of the annealing schedule and may be too slow when applied to a practical power system. The GA can find a global solution after sufficient iterations, but has a high computational burden. The ETQ integrates EP, Tabu search and quadratic programming methods to solve the PELD problem of units with POZs.

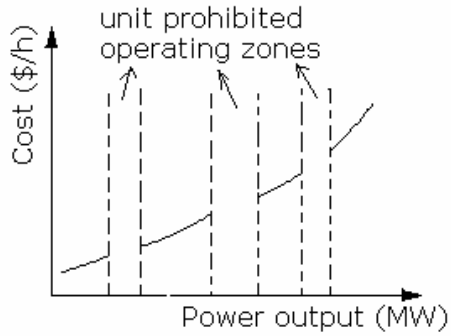

Fig. 1 Unit input/ouput characteristic

II. PROBLEM FORMULATION

The PELD problem can be described as an optimization process with objective [5]:

$$\textit{Minimize} \quad \sum_{i=1}^{n_p} F_i(P_i) \tag{1}$$

Where $F_i(P_i)$ is the fuel cost function of the i^{th} unit, P_i is the power generated by the i^{th} unit, and n_p is the number of on-line units. Subject to the equality constraint of the power balance as:

$$\sum_{i=1}^{n_p} P_i = P_d + T_L \tag{2}$$

Where P_d is the system load demand and T_L is the transmission loss, and generating capacity constraints are expressed as:

$$P_i^{\min} \le P_i \le P_i^{\max}, \quad i = 1, \cdots, n_p \tag{3}$$

Where P_i^{min} and P_i^{max} are the minimum and maximum power outputs of the i^{th} unit.

Additional constraints on the unit operating range denote the effects of a generator with POZs [7]:

$$\begin{aligned} & P_i^{\min} \le P_i \le P_{i,1}^{l} \quad or \\ & P_{i,j-1}^{u} \le P_i \le P_{i,j}^{l}, \quad j = 2, \cdots, n_i \quad or \\ & P_{i,ni}^{u} \le P_i \le P_i^{\max}, \quad \forall i \in \omega \end{aligned} \tag{4}$$

Where $P_{i,j}^{l}$ and $P_{i,j}^{u}$ respectively denote the lower and upper bounds of the j^{th} prohibited zone for the i^{th} unit, n_i is the number of prohibited zones in the i^{th} unit, and ω is the set of all on-line units with prohibited zones.

Clearly, a dispatching unit with n_i prohibited zones has its entire operating region to be divided into (n_i+1) disjoint operating sub-regions. The total number of decision sub-spaces resulting from that division may be counted as follows:

$$N = \prod_{i \in \omega} (n_i + 1) \tag{5}$$

Equation (5) indicates that the total number of decision sub-spaces increases extremely rapidly as the number of units with prohibited zones increases.

III. The Proposed Algorithm

This section describes the proposed IGAMU. First, the IGA is provided, then the MU for managing system constraints is introduced, and finally solution procedures of the proposed IGAMU are presented.

A. IGA

To enhance GA's computational efficiency, an improved evolutionary direction operator (IEDO) altered from [9] and a migration [10] are embedded in GA to form the IGA. The IEDO [11] includes choosing the three best solutions in each generation to perform the evolutionary direction operator algorithm to find a new solution superior to the original best solution.

The IGA has been applied successfully to constrained optimization problems such as: design of an induction motor controller for tracking control [11], and nonlinear mixed-integer optimization problems [12]. The operations of IGA had been gone into details in [11] and [12].

B. MU

Considering the NLP with general constraints as follows:

$$\begin{aligned} & \min_x f(x) \\ subject \quad to \quad & h_k(x) = 0, \quad k = 1, ..., m_e \\ & g_k(x) \le 0, \quad k = 1, ..., m_i \end{aligned} \tag{6}$$

where $h_k(x)$ and $g_k(x)$ stand for equality and inequality constraints, respectively.

Michalewicz et al. [13] surveyed and compared several constraint-handling techniques used in evolutionary algorithms, and showed that the penalty function method is among the most popularly methods for managing constraints. Powell [14] noted that classical optimization methods with a penalty function have certain disadvantages that become severe when using large penalty parameters, because the penalty function becomes "ill-conditioned", making good solutions difficult to obtain. However, a situation in which the penalty parameters are too small makes it impossible for the constraint violation to contribute a high cost to the penalty function. Therefore, selecting suitable penalty parameters is not trivial. Herein, the MU [12] is introduced to manage constrained optimization problems. Such a method can eradicate the ill-conditioned property of the objective function.

The augmented Lagrange function [12] for constrained optimization problems is defined as:

$$\begin{aligned} L_a(x, \nu, \upsilon) = & f(x) + \sum_{k=1}^{m_e} \alpha_k \left\{ \left[h_k(x) + \nu_k \right]^2 - \nu_k^2 \right\} \\ & + \sum_{k=1}^{m_i} \beta_k \left\{ \left\langle g_k(x) + \upsilon_k \right\rangle_+^2 - \upsilon_k^2 \right\} \end{aligned} \tag{7}$$

Where α_k and β_k are the positive penalty parameters, and the corresponding Lagrange multipliers $\nu = (\nu_1, \ldots, \nu_{m_e})$ and $\upsilon = (\upsilon_1, \ldots, \upsilon_{m_i}) \ge 0$ are associated with equality and inequality constraints, respectively.

The contour of the augmented Lagrange function does not change shape between generations while constraints are linear. Therefore, the contour of the augmented Lagrange function is simply shifted or biased in relation to the original objective function, *f(x)*. Consequently, small penalty parameters can be used in the MU. However, the shape of contour of L_a is changed by penalty parameters while the constraints are nonlinear, demonstrating that large penalty parameters still create computational difficulties. Adaptive penalty parameters of the MU are employed to alleviate the above difficulties, and Table I presents computational procedures of the MU.

C. Solution Procedures of the Proposed Algorithm

Figure 2 displays the flow chart of the proposed algorithm, which has two iterative loops. The augmented Lagrange function is used to obtain a minimum value in the inner loop with the given penalty parameters and multipliers, which are then updated in the outer loop toward producing an upper limit of L_a. When both inner and outer iterations become sufficiently large, the augmented Lagrange function converges to a saddle-point of the dual problem [12]. Advantages of the proposed IGAMU are that the IGA efficiently searches the optimal

solution in the economic dispatch process and the MU effectively tackles system constraints.

TABLE I. COMPUTATIONAL PROCEDURES OF THE MU

Step 1. Set the initial iteration $l=0$. Set initial multiplier, $\nu_k^l=\nu_k^0=0,k=1,...,m_e$, $\upsilon_k^l=\upsilon_k^0=0,k=1,...,m_i$, and the initial penalty parameters, α_k>0, k=1,…,m_e andβ_k>0, k=1,…,m_i . Set tolerance of the maximum constraint violation,ε_k (e.g. ε_k =10^{32}), and the scalar factors, ω_1 >1 andω_2 >1 .

Step 2. Use a minimization solver, e.g. IGA, to solve $L_a\left(x,\nu^l,\upsilon^l\right)$. Let x_b^l; be a minimum solution to the problem $L_a\left(x,\nu^l,\upsilon^l\right)$.

Step 3. Evaluate the maximum constraint violation as $\hat{\varepsilon}_k=\max\{\max_k|h_k|,\max_k|\max(g_k,-\upsilon_k)|\}$, and establish the following sets of equality and inequality constraints whose violations have not been improved by the factorω_1:

$$I_E=\{k:|h_k|>\frac{\varepsilon_k}{\omega_1},\quad k=1,\cdots,m_e\}$$

$$I_I=\{k:|\max(g_k,-\upsilon_k)|>\frac{\varepsilon_k}{\omega_1},\quad k=1,\cdots,m_i\}$$

Step 4. If $\hat{\varepsilon}_k\ge\varepsilon_k$, let $\alpha_k=\omega_2\alpha_k$ and $\nu_k^{l+1}=\nu_k^l/\omega_2$ for all $k\in I_E$, let $\beta_k=\omega_2\beta_k$ and $\upsilon_k^{l+1}=\upsilon_k^l/\omega_2$ for all $k\in I_I$, and go to step7. Otherwise, go to step 5.

Step 5. Update the multipliers as follows:

$$\nu_k^{l+1}=h_k\left(x_b^l\right)+\nu_k^l$$

$$\upsilon_k^{l+1}=\left\langle g_k\left(x_b^l\right)+\upsilon_k^l\right\rangle_+=\upsilon_k^l+\max\left\{g_k\left(x_b^l\right),-\upsilon_k^l\right\}$$

Step 6. If $\hat{\varepsilon}_k\le\varepsilon_k/\omega_1$, let $\varepsilon_k=\hat{\varepsilon}_k$ and go to step 7. Otherwise, let $\alpha_k=\omega_2\alpha_k$ and $\nu_k^{l+1}=\nu_k^l/\omega_2$ for all $k\in I_E$, and let $\beta_k=\omega_2\beta_k$ and $\upsilon_k^{l+1}=\upsilon_k^l/\omega_2$ for all $k\in I_I$. Let $\varepsilon_k=\hat{\varepsilon}_k$ and go to step 7.

Step 7. If the maximum iteration reaches, stop. Otherwise, repeat steps 2 to 6.

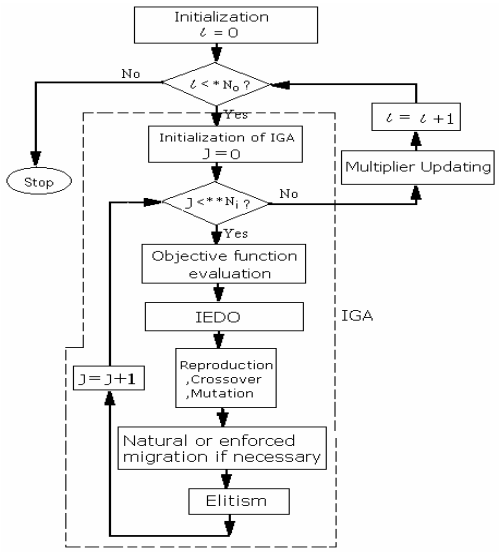

Fig. 2 The flow chart of the IGAMU

*No : maximum number of iterations of outer loop

**Ni : maximum number of iteartions of inner loop

IV. SYSTEM SIMULATIONS

This section applies a practical system to illustrate the effectiveness of the proposed IGAMU with respect to the quality of the solution obtained for realistic PELD problems with POZs. The Elitism technique [1] was used in the proposed IGAMU and the GA-MU. Both the IGAMU and the GA-MU were directly coded using real values, and were implemented on a personal computer (PIII-700) in FORTRAN-90. References of [11] and [12] suggest setting factors for the proposed algorithm. Setting factors employed in this example was as follows: the iteration number of the IEDO operation N_L was set to 4; the population size N_p was set to 5 and 20 for the IGAMU and the GA-MU, respectively, and iteration numbers of the outer loop and inner loop were set to (outer, inner) as (30, 3000).

A. Simulation Results

A practical system of units with POZs having nonconvex cost functions was considered in this example, and the system data are identical to that used in [7]. This system has 15 on-line units that supply a system demand of 2650MW. Among these dispatching generators, units 2, 5 and 6 have three POZs, and unit 12 has two POZs, that form 192 decision sub-spaces for this realistic system. The implementation of this example can be represented as follows:

$$L_a(x,\nu,\upsilon)=f(x)+\alpha_1\{[h_1(x)+\nu_1]^2-\nu_1^2\}+\sum_{k=1}^{4}\beta_k\{\langle g_k(x)+\upsilon_k\rangle_+^2-\upsilon_k^2\} \tag{8}$$

$$objective:\ \min_{x=(P_1,P_2,...P_{15})}\ f(x)=\sum_{i=1}^{15}F_i(P_i) \tag{9}$$

$$subject\ \ to\ \ h_1:\ \sum_{i=1}^{15}P_i-P_L-P_d=0 \tag{10}$$

$$\begin{aligned}
&g_1: P_2^{150}\le P_2\le P_{2,1}^{185}, or\quad P_{2,1}^{225}\le P_2\le P_{2,2}^{305},\\
&\quad or\quad P_{2,2}^{335}\le P_2\le P_{2,3}^{420},\quad or\quad P_{2,3}^{450}\le P_2\le P_2^{455}\\
&g_2: P_5^{105}\le P_5\le P_{5,1}^{180}, or\quad P_{5,1}^{200}\le P_5\le P_{5,2}^{260},\\
&\quad or\quad P_{5,2}^{335}\le P_5\le P_{5,3}^{390},\quad or\quad P_{5,3}^{420}\le P_5\le P_5^{470}\\
&g_3: P_6^{135}\le P_6\le P_{6,1}^{230}, or\quad P_{6,1}^{255}\le P_6\le P_{6,2}^{365},\\
&\quad or\quad P_{6,2}^{395}\le P_6\le P_{6,3}^{430},\quad or\quad P_{6,3}^{455}\le P_6\le P_6^{460}\\
&g_4: P_{12}^{20}\le P_{12}\le P_{12,1}^{30}, or\quad P_{12,1}^{55}\le P_{12}\le P_{12,2}^{65},\\
&\quad or\quad P_{12,2}^{75}\le P_{12}\le P_{12}^{80}
\end{aligned} \tag{11}$$

This complex optimization problem contains one objective function with fifteen variable parameters, (P_1, P_2, …, P_{15}), one equality constraint, (h_1), and four inequality constraints, (g_1 to g_4), owing to four units having the POZs. The proposed IGAMU compared with λ-δ[7], SGA [7], DCGA [7], ETQ [8] and the GA-MU with respect to the best dispatch solution obtained, are given in Table II. Unit loadings in Table II show why the λ-δ method is not capable of solving this PELD problem. For example, using λ-δ technique results in a solution that requires unit 5 to operate in one of the POZs. The proposed

algorithm on the other hand, provides final optimal loadings that do not fall in any of the 'illegal' zones. The proposed algorithm is superior to other methods in the solution quality, and it has the exact generation for the system load demand. From Table II, the proposed IGAMU also outperforms the GA-MU in terms of the solution quality and the convergence rate. Hence, for PELD problems of different size and complexity, the proposed IGAMU proves to be the best algorithm among previous methods mentioned above.

TABLE II. COMPARED RESULTS OF PREVIOUS METHODS, THE GA-MU, AND THE PROPOSED IGAMU

Methods	λ-δ	SGA	DCGA	ETQ	GA-MU	IGAMU
Unit1	455	451.4	406.1	450	454.9783	454.9794
Unit2	455	455	453.8	450	454.9881	454.9638
Unit3	130	130	130	130	129.9941	129.9848
Unit4	130	129.1	130	130	129.9946	129.9919
Unit5	295.3*	337.1	355	335	259.9989	259.9884
Unit6	460	429.5	456.8	455	459.9893	459.9892
Unit7	465	464.4	459.8	465	464.9859	464.9731
Unit8	60	60	60	60	60.0172	60.0355
Unit9	25	26.6	26.6	25	25.0110	25.0137
Unit10	20	27.1	21.6	20	20.0095	20.0383
Unit11	43.4	25.7	36.2	20	58.4360	70.0417
Unit12	56.3	59	59	55	76.5870	64.9759
Unit13	25	25	25	25	25.0040	25.0105
Unit14	15	15	15	15	15.0029	15.0077
Unit15	15	15	15	15	15.0032	15.0061
TP (MW)	2650.0	2649.9	2649.9	2650	2650.0000	2650.0000
TC ($/h)	32503#	32517	32515	32507.5	32506.3740	32506.3390

* An unit loading in a prohibited zone

An infeasible result

B. *The Performance of the Proposed Algorithm*

Table III lists the relative frequencies of convergence for the proposed algorithm and the GA-MU among 100 randomly initiated trials, and the proposed IGAMU has shown the better solution quality with a high probability to demonstrate its excellent performance. Moreover, the compared tests are also itemized in Table IV. The proposed IGAMU performs better than the GA-MU in terms of mean cost as well as mean time, even though the proposed method applies a small population. Tables III and IV clearly illustrate that the proposed IGAMU is superior to the GA-MU in both effectiveness and efficiency.

C. *Penalty Setting Discussion*

This test compared the proposed IGAMU with IGA using the fixed penalty (IGAFP) in terms of penalty settings, demonstrates that the IGAMU has the advantage of automatically adjusting the randomly given penalty to an appropriate value. Herein, the IGAMU with initial penalty parameters of 10, 10^3 and 10^6, was employed to solve this example again, and the sum of constraint violations is defined as $SCV = |h_1| + \sum_{k=1}^{4} \max\{g_k, 0.0\}$, to explain the effect of constraint feasibility on the final solution. Table V shows three cases of the best solutions obtained among 100 randomly initiated trials in each case. The adaptive penalty parameters and multipliers, (α_k, ν_k, β_k and υ_k), are automatically updated by the MU to avoid ill conditioning. Experimental results of these three cases produced the same *TP* and the almost *TC*, and all *SCVs* were less than 10^{-11}. These clearly confirm that the IGAMU is effective in managing constraints of the realistic PELD problem by automatically altering the randomly given penalty to an appropriate value and requiring only a small population. The IGAFP was also employed to solve this example. The IGAFP, like the IGAMU, applies 90,000 iterations. Table VI presents three cases of the best solutions obtained for various fixed penalty parameters among 100 randomly initiated trials in each case. For the case of fixed penalty parameter 10, the *TC* yielded by the IGAFP is below those produced by the IGAMU. Nevertheless, such comparison is meaningless, since all *SCV*s of the IGAFP are larger than those of the IGAMU. This case of IGAFP violated the system constraint (h_1) and yielded an infeasible solution. Therefore, this test clearly indicates that the IGAMU can automatically alter the randomly given penalty to an appropriate value to achieve a global (or near global) solution. However, the IGAFP can produce a satisfactory result only using a large penalty parameter. Because the IGAMU actively uses a flexible multiplier updating strategy to probe for the new solution, while the IGAFP only employs a fixed penalty function to test passively feasible and infeasible solutions.

TABLE III. RELATIVE FREQUENCIES OF CONVERGENCE USING THE GA-MU AND THE PROPOSED IGAMU

Methods	Range of cost ($/h)						
	32650 - 32675	32625 - 32650	32600 - 32625	32575 - 32600	32550 - 32575	32525 - 32550	32500 - 32525
GA-MU	3	3	4	5	5	52	28
IGAMU	0	2	0	4	4	47	43

TABLE IV. SUMMARIZED RESULTS OF THE COMPARISON BETWEEN THE GA-MU AND THE PROPOSED IGAMU

Methods	Mean time (s)	Best time (s)	Mean cost ($/h)	Max. cost ($/h)	Min. cost ($/h)
GA-MU	20.72	20.62	32543.2681	32664.9540	32506.3740
IGAMU	7.69	7.47	32530.5600	32640.5020	32506.3390

V. CONCLUSIONS

This paper presented the IGAMU to solve practical PELD problems of complexity having nonconvex cost curves where conventional mathematical methods are inapplicable. The IGA enhances the proposed method to efficiently search and actively explore the solution, and the MU supports the proposed approach to efficaciously manage system constraints. The proposed algorithm integrates the IGA and the MU such that it has the following merits: straightforward concept; easy implementation; better effectiveness than previous methods; better effectiveness and efficiency than the GA-MU; automatic adjustment of the randomly assigned penalty to an appropriate value, and the requirement for only a small population in realistic PELD problems. The comparative results showed that the proposed algorithm has the merits mentioned above for applying real-world PELD operations.

ACKNOWLEDGMENT

Financial support given to this research by the National Science Council, Taiwan, R.O.C. under Grant No. 96-2628-E-252-001 is greatly appreciated.

TABLE V. RESULTS OF THE PROPOSED IGAMU WITH VARIOUS INITIAL PENALTY PARAMETERS

(a) Initial penalty =10

Constraints		final penalties and multipliers of L_a				The proposed IGAMU Initial penalty=10
		α_k, β_k		ν_k, υ_k		
h_1	-1.8189E-12	α_1	1.0000E+22	ν_1	-2.9944E-13	*SCV*: 0.0000
g_1	0.0000E+00	β_1	1.0000E+20	υ_1	0.0000E+00	*TP*(MW): 2650.0000
g_2	0.0000E+00	β_2	1.0000E+20	υ_2	0.0000E+00	*TC*($/h): 32506.3390
g_3	0.0000E+00	β_3	1.0000E+20	υ_3	3.8452E-21	CPU_time(s): 7.49
g_4	0.0000E+00	β_4	1.0000E+20	υ_4	1.3128E-20	

(b) Initial penalty =10^3

Constraints		final penalties and multipliers of L_a				The proposed IGAMU Initial penalty=10^3
		α_k, β_k		ν_k, υ_k		
h_1	-4.5474E-13	α_1	1.0000E+21	ν_1	7.1524E-13	*SCV*: 0.0000
g_1	0.0000E+00	β_1	1.0000E+21	υ_1	0.0000E+00	*TP*(MW): 2650.0000
g_2	0.0000E+00	β_2	1.0000E+21	υ_2	0.0000E+00	*TC*($/h): 32506.7050
g_3	0.0000E+00	β_3	1.0000E+21	υ_3	0.0000E+00	CPU_time(s): 7.63
g_4	0.0000E+00	β_4	1.0000E+21	υ_4	0.0000E+00	

(c) Initial penalty =10^6

Constraints		final penalties and multipliers of L_a				The proposed IGAMU Initial penalty=10^6
		α_k, β_k		ν_k, υ_k		
h_1	-9.0950E-13	α_1	1.0000E+21	ν_1	-1.7934E-12	*SCV*: 0.0000
g_1	0.0000E+00	β_1	1.0000E+16	υ_1	0.0000E+00	*TP*(MW): 2650.0000
g_2	0.0000E+00	β_2	1.0000E+16	υ_2	0.0000E+00	*TC*($/h): 32506.7980
g_3	0.0000E+00	β_3	1.0000E+16	υ_3	0.0000E+00	CPU_time(s): 7.52
g_4	0.0000E+00	β_4	1.0000E+16	υ_4	0.0000E+00	

TABLE VI. RESULTS OF THE IGAFP WITH VARIOUS FIXED PENALTY PARAMETERS

	IGAFP		
Fixed penalty	$\alpha_k, \beta_k = 10$	$\alpha_k, \beta_k = 10^3$	$\alpha_k, \beta_k = 10^6$
h_1	5.35410E-01	5.26512E-03	5.21392E-06
g_1	0.00000E+00	0.00000E+00	0.00000E+00
g_2	0.00000E+00	2.88972E-07	0.00000E+00
g_3	0.00000E+00	0.00000E+00	0.00000E+00
g_4	0.00000E+00	0.00000E+00	0.00000E+00
SCV	0.5354	0.0053	0.0000
TP (MW)	2649.4646	2649.9947	2650.0000
TC ($/h)	32500.4487	32507.7015	32507.8321
CPU_time (s)	7.25	7.14	7.25

REFERENCES

[1] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Reading, MA: Addison-Wesley, 1989.

[2] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd. ed. Berlin, Heidelberg: Springer-Verlag, 1996.

[3] C. E. Lin and G. L. Viviani, "Hierarchical economic dispatch for piecewise quadratic cost functions," IEEE Trans. Power Apparatus and Syst., PAS-103, no. 6, 1984, pp. 1170-1175.

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

[5] A. J. Wood and B. F. Wollenberg, Power Generation, Operation, and Control, New York: Wiley& Sons, the 2nd ed., 1996.

[6] K. P. Wong and C. C. Fung, "Simulated annealing based economic dispatch algorithm," IEE Proc. C, vol.140, no. 6, 1993, pp. 509- 515.

[7] S. O. Orero and M. R. Irving, "Economic dispatch of generators with prohibited operating zones: a genetic algorithm approach," IEE Proc. – Gener. Transm. Distrib., vol. 143, no. 6, 1996, pp. 529 -534.

[8] W. M. Lin, F. S. Cheng, and M. T. Tsay, "Nonconvex economic dispatch by integrated artificial intelligence," IEEE Trans. Power Syst., vol. 16, no. 2, 2001, pp. 307- 311.

[9] K. Yamamoto and O. Inoue, "New evolutionary direction operator for genetic algorithms," AIAA Journal Technical Notes, vol. 33, no. 10, 1995, pp.1990-1993.

[10] J. P. Chiou and F. S. Wang, "A hybrid method of differential evolution with application to optimal control problems of a bioprocess system," in Proc. 1998 IEEE on Evolutionary Computation Conf., pp. 627-632.

[11] C. L. Chiang and C. T. Su, "Tracking control of induction motor using fuzzy phase plane controller with improved genetic algorithm," Electric Power Systems Research, vol. 73, 2005, pp. 239- 247.

[12] C. L. Chiang, C. T. Su, and F. S. Wang, "Augmented Lagrangian method for evolutionary optimization of mixed-integer nonlinear constrained problems," Intern. Math. J., vol.2, no. 2, 2002, pp. 119-154.

[13] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," Evolutionary Computation, vol.4, no.1, 1996, pp.1-32.

[14] M. J. D. Powell, "Algorithms for nonlinear constraints that use Lagrangian functions," Math. Programming, vol. 14, April 1955, pp. 224-248.