Journal of ELECTRICAL ENGINEERING, VOL. 57, NO. 4, 2006, 211–217

ECONOMIC POWER DISPATCH USING EVOLUTIONARY ALGORITHM

Mimoun Younes — Mostefa Rahli — Lahouari Abdelhakem Koridak

In this paper we consider the important problem of economic operation of power systems:how to operate a power system to supply all loads at minimum cost. Here we assume that we have some flexibility in adjusting the power delivered by each generator. Usually the total load is less than the available generator capacity. But, there are many possible generation assignments.

In this work, genetic algorithm (GA) is a solution to the economic dispatch problem of IEEE 9-bus, 30 bus and 57-bus model systems.

K e y w o r d s: Economic dispatch, genetic algorithms

1 INTRODUCTION

The optimal power flow (OPF) [1] is a nonlinear programming problem, and is used to determine optimal outputs of generators, bus voltage and transformer tap setting in a power system with objective to minimize the total production cost while the system is operating within its security limit.

Since OPF was introduced in 1968 [2], several methods have been employed to solve this problem, *eg* Gradient base [2], linear programming method [3] and quadratic programming [4]. However, all of these methods suffer from three main problems.

Firstly, they may not be able to provide the optimal solution and usually getting stuck at a local optimum [5]. Secondly, all these methods are based on the assumption of continuity and differentiability of the objective function, which is not true in a practical system.

Finally, all these methods cannot be applied with discrete variables which are transformer taps. It seems that GA is an appropriate method to solve this problem, which eliminates the above drawbacks.

This method was tested on the IEEE 9 bus, 30 bus and 57 bus test systems. The result of the study is compared with those obtained from matpower [6].

2 PROBLEM FORMULATION

If the input-output [7] characteristic of a generator i is represented by function f_i, then the standard OPF [8, 9] can be described mathematically as follows:
Objective function

$$\mathrm{Min}\Big\{f\big(P_G,Q_G\big)=\sum_{i=1}^{nG}\big(f_i\big(P_{Gi}\big)+\big(f_i\big(Q_{Gi}\big)\Big\} \qquad (1)$$

subject to equality constraints

$$\sum_{i=1}^{nG}P_{Gi}-\sum_{j=1}^{nc}P_{Dj}-P_L=0 \qquad (2)$$

$$\sum_{i=1}^{nG}Q_{Gi}-\sum_{j=1}^{nc}Q_{Dj}-Q_L=0 \qquad (3)$$

and inequality constraints

$$P_{Gi}^{\min}\le P_{Gi}\le P_{Gi}^{\max} \qquad i=1 \text{ to } N_G \qquad (4)$$

$$Q_{Gi}^{\min}\le Q_{Gi}\le Q_{Gi}^{\max} \qquad i=1 \text{ to } N_D \qquad (5)$$

where

$f\big(P_G,Q_G\big)$: Total production cost (\$/h).

$f_i\big(P_{Gi}\big)$: The costs of active power generation, for generator i at a given dispatch point.

$f_i\big(Q_{Gi}\big)$: The costs of reactive power generation, for generator i at a given dispatch point.

P_{Dj},Q_{Dj} : active and reactive power load at bus j

P_{Gi},Q_{Gi} : active and reactive power generation at bus i

3 GENETIC ALGORITHM

GA are stochastic search and optimization techniques shaped by the evolution theory. Goldberg [10, 11] states the difference between GA and classical optimisation and search methods in four ways:

1. GA work with a coding of the parameter set, not the parameters themselves.
2. GA search from a population of points, not a single point.

* Department of Mechanical Engineering University of Sidi Bel Abbes, BP 89 Sidi DJillali, Algéria ; ** Department of Electrical Engineering, U.S.T.O, B.P.1505, Oran El m'naouer, Algéria; E-mail: mnyounes2000@yahoo.fr, rahlim@yahoo.fr, Koridak@univ-usto.dz

ISSN 1335-3632

3. GA use payoff (objective function) information, not derivatives or other auxiliary knowledge.
4. GA use probabilistic transition rules, not deterministic rules. Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved.

The algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope that the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are elected according to their fitness — the more suitable they are the more chances they have to reproduce.

This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

3.1 Parameters of GA

Recommendations are often results of some empiric studies of GA [12], which were often performed only on binary encoding.

Crossover rate — this rate generally should be high, about 80 %–95 %. (However some results show that for some problems crossover rate about 60 % is the best.)

Mutation rate — on the other side, mutation rate should be very low. Best rates reported are about 0.5 %–1 %.

Population size — it may be surprising that a very big population size usually does not improve performance of GA (in meaning of speed of finding solution). A good population size is about 20–30, however sometimes sizes 50–100 are reported as best. Some research also shows that best population size depends on encoding, on size of encoded string. It means, if you have a chromosome with 32 bits, the population should be say 32, but surely two times more than the best population size for chromosome with 16 bits.

3.2 Encoding

Encoding depends on the problem and also on the size of instance of the problem.

The chromosome should in some way contain information about solution which it represents. The most used way of encoding is a binary string. The chromosome then could look like this:

Chromosome 1	1101100100110110
Chromosome 2	1101111000011110

Each chromosome has one binary string. Each bit in this string can represent some characteristic of the solution. Or the whole string can represent a number. Of course, there are many other ways of encoding. This depends mainly on the solved problem. For example, one can encode directly integer or real numbers; sometimes it is useful to encode some permutations and so on.

3.3 Crossover and mutation type

Operators depend on encoding and on the problem.

Genetic algorithms have been used for difficult problems. Advantage of GA is in their parallelism [11]. GA is travelling in a search space with more individuals (and with genotype rather than phenotype) so they are less likely to get stuck in a local extreme like some other methods.

They are also easy to implement. Once you have some GA, you just have to write a new chromosome (just one object) to solve another problem. With the same encoding you just change the fitness function and it is all. On the other hand, choosing encoding and fitness function can be difficult.

As you can see from the genetic algorithm outline, the crossover and mutation are the most important parts of the genetic algorithm. The performance is influenced mainly by these two operators.

After we have decided what encoding we will use, we can make a step to crossover. Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way how to do this is to choose randomly some crossover point and everything before this point copy from a first parent and then everything after a crossover point copy from the second parent. Crossover can then look like this (| is the crossover point):

Chromosome 1	**11011**|**00100110110**
Chromosome 2	11011|11000011110
Offspring 1	**11011**|11000011110
Offspring 2	11011|**00100110110**

There are other ways how to make crossover, for example we can choose more crossover points.

Crossover can be rather complicated and strongly depends on encoding of the encoding of chromosome. A specific crossover made for a specific problem can improve performance of the genetic algorithm.

After a crossover is performed, mutations take place. This is to prevent falling all solutions in population into a local optimum of the solved problem. Mutation changes randomly the new offspring. For binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Mutation can then be following:

Original offspring 1	110**1**1111000011110
Original offspring 2	110110**0**100110110
Mutated offspring 1	110**0**111000011110
Mutated offspring 2	110110**1**100110**0**0

The mutation depends on the encoding as well as the crossover. For example when we are encoding permutations, mutation could be exchanging two genes.

3.4 Crossover and Mutation Probability

There are two basic parameters of GA — crossover probability and mutation probability.

Crossover probability — says how often crossover will be performed. If there is no crossover, offspring is an exact copy of parents. If there is a crossover, offspring is made

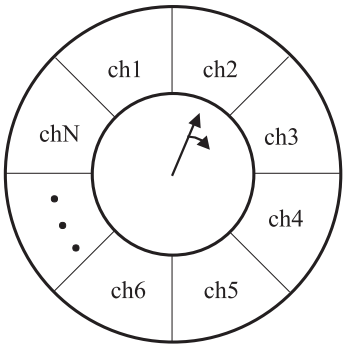

Fig. 1. Roulette Wheel.

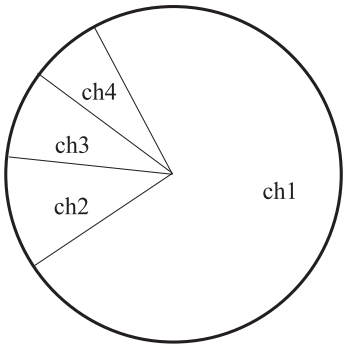

Fig. 2. Situation before ranking.

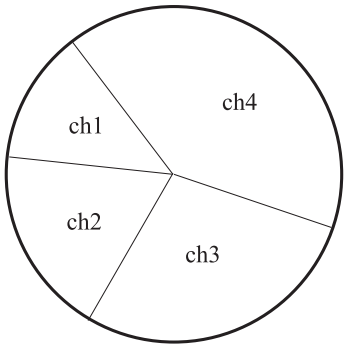

Fig. 3. Situation after ranking.

from parts of parents' chromosome. If crossover probability is 100 %, then all offspring is made by crossover. If it is 0 %, whole new generation is made from exact copies of chromosomes from old population (but this does not mean that the new generation is the same!).

Crossover is made in hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However it is good to leave some part of population survive to the next generation.

Mutation probability — says how often parts of chromosome will be mutated. If there is no mutation, offspring is taken after crossover (or copy) without any change. If mutation is performed, part of chromosome is changed. If mutation probability is 100 %, whole chromosome is changed, if it is 0 %, nothing is changed.

Mutation is made to prevent falling GA into a local extreme, but it should not occur very often because then GA will in fact change to random search.

There are also some other parameters of GA. One also important parameter is the population size. The population size says how many chromosomes are in population (in one generation). If there are too few chromosomes, GA have a few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA slows down. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to increase population size because it does not make solving the problem faster.

3.5 Selection

Chromosomes are selected from the population to be parents to crossover. The problem is how to select these chromosomes. According to Darwin's evolution theory the best ones should survive and create new offspring. There are many methods how to select the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others.

Roulette Wheel Selection

Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a roulette wheel where are placed all chromosomes in the population, every has its place big accordingly to its fitness function, like in Fig. 1. Then a marble is thrown there and selects the chromosome. Chromosome with bigger fitness will be selected more times.

Rank Selection

The previous selection will have problems when the fitnesses differs very much. For example, if the best chromosome fitness is 90 % of all the roulette wheel then the other chromosomes will have very few chances to be selected. Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 *etc.* and the best will have fitness N (number of chromosomes in population).

You can see in the following picture, how the situation changes after changing fitness to order number.

After this all the chromosomes have a chance to be selected. But this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones.

3.6 Steady-State Selection

This is not a particular method of selecting parents. The main idea of this selection is that big part of chromosomes should survive to the next generation.

GA then works in a following way. In every generations a few (good — with high fitness) chromosomes are selected for creating a new offspring. Then some (bad — with low fitness) chromosomes are removed and the new offspring is placed in their place. The rest of population survives to the new generation.

3.7 Elitism

The idea of elitism has already been introduced. When creating a new population by crossover and mutation, we have a big chance that we will loose the best chromosome.

Elitism is the name of a method which first copies the best chromosome (or a few best chromosomes) to the new population. The rest is done in the classical way. Elitism can very rapidly increase performance of GA because it prevents losing the best found solution.

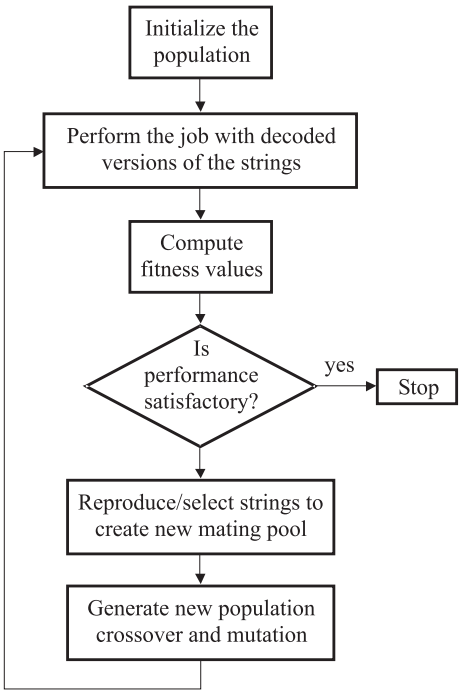

Fig. 4. Basic steps of a genetic algorithm [15].

3.8 Exploitation

When traversing a search space, Exploitation is the process of using information gathered from previously visited points in the search space to determine which places might be profitable to visit next.

An example is hill-climbing, which investigates adjacent points in the search space, and moves in the direction giving the greatest increase in Fitness. Exploitation techniques are good at finding local maxima.

3.9 Exploration

The process of visiting entirely new regions of the search space to see if anything promising may be found there. Unlike Exploitation, Exploration involves leaps into the unknown. Problems which have many local maxima can sometimes only be solved by this sort of random search.

4 MATPOWER

MATPOWER is a package of Matlab m-files for solving the power flow and optimal power flow problems.

It is intended as a simulation tool for researchers and educators which will be easy to use and modify. MATPOWER is designed to give the best performance possible while keeping the code simple to understand and modify.

The data files used by MATPOWER are simply Matlab m-files which define and return the variables baseMVA, bus, branch, gen, area, and gencost. The bus, branch, and gen variables are matrices.

Each row in the matrix corresponds to a single bus, branch, or generator, respectively.

The columns are similar to the columns in the standard IEEE and PTI formats. The details of the specification of the MATPOWER.

4.1 Power Flow

MATPOWER has three power flow solvers. The default power flow solver is based on a standard Newton's method [16] using a full Jacobian, updated at each iteration. This method is described in detail in many textbooks. The other two power flow solvers are variations of the fast-decoupled method [17].

MATPOWER implements the XB and BX variations as described in [18]. Currently, MATPOWER's power flow solvers do not include any transformer tap changing or feasibility checking capabilities.

Performance of the power flow solvers should be excellent even on very large-scale power systems, since the algorithms and implementation take advantage of Matlab's built-in sparse matrix handling. On a Sun Ultra 2200, MATPOWER solves a 9600-bus test case in about 10 seconds, and a 38400 bus case in about 50 seconds.

4.2 Optimal Power Flow

MATPOWER includes two solvers for the optimal power flow (OPF) problem. The first is based on the constr function included in Matlab's Optimization Toolbox, which uses a successive quadratic programming technique with a quasi-Newton approximation for the Hessian matrix.

The second approach is based on linear programming. It can use the LP solver in the Optimization Toolbox or other Matlab LP solvers available from third parties.

The performance of MATPOWER's OPF solvers depends on several factors. First, the constr function uses an algorithm which does not exploit or preserve sparsity, so it is inherently limited to small power systems. The LP-based algorithm, on the other hand, does preserve sparsity. However, the LP-solver included in the Optimization Toolbox does not exploit this sparsity. In fact, the LP-based method with the default LP solver performs worse than the constr-based method, even on small systems.

Fortunately, there are LP-solvers available from third parties which do exploit sparsity.

In general, these yield much higher performance. One in particular, called bpmpd [19] (actually a QP-solver), has proven to be robust and efficient.

It should be noted, however that even with a good LP-solver, MATPOWER's LP-based OPF solver, unlike its power flow solver, is not suitable for very-large scale problems. Substantial improvements in performance may still be possible, though they may require significantly more complicated coding and possibly a custom LP-solver. On a Sun Ultra 2200, the LP-based OPF solver using bpmpd solves a 30-bus system in under 4 seconds and a 118-bus case in under 25 seconds.

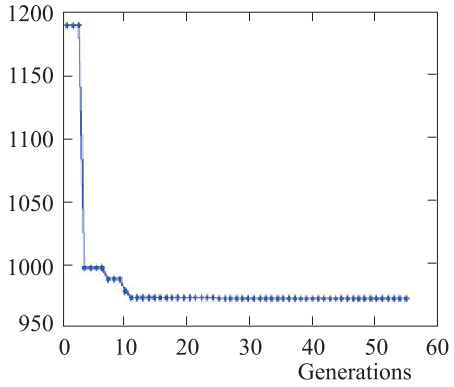

Fig. 5. Evolution of the GA (9 bus).

5 SIMULATION RESULTS

Characteristic values of the IEEE 9 bus test system and losses respectively, are as follows:

Table 1. Generator operating limits and quadratic cost function coefficients.

Bus	Pmin	Pmax	Qmin	Qmax	a	b	c
1	10	250	–300	300	0.11	5	150
2	10	300	–300	300	0.085	1.2	600
3	10	270	–300	300	0.1225	1	335

$$\sum P_D = 315.00 \tag{6}$$

$$\sum Q_D = 115.00 \tag{7}$$

$$P_L = 307.3\,\text{MW}, \quad Q_L = 46.36\,\text{MVAR} \tag{8}$$

The problem is minimize the objective function

$$\text{Min}\Big\{f(P_G) = \sum_{i=1}^{nG} f_i(P_G)\Big\} \tag{9}$$

Subject to constraints

$$P_{G1} + P_{G2} + P_{G3} = 31.318\,\text{MW} \tag{10}$$

$$Q_{G1} + Q_{G2} + Q_{G3} = 31.318\,\text{MW} \tag{11}$$

$$\sum P_D = 00.315 \tag{12}$$

$$\sum Q_D = 00.115 \tag{13}$$

Table 2. Parameter values for GA.

Population size	30
Number of generations	60
Crossover probability	0.8
Mutation probability	0.05
Generation-elitism	5
Number of bit for encode active power generation	16

Table 3. Result of the study.

methods	GA	Matpower
P_{G1}^{opt} [MW]	87.70	89.80
P_{G2}^{opt} [MW]	135.68	134.32
P_{G3}^{opt} [MW]	94.910	94.19
Q_{G1}^{opt} [MWAR]	–29.61	12.97
Q_{G2}^{opt} [MWAR]	–7.93	0.03
Q_{G3}^{opt} [MWAR]	27.91	–22.63
Coût [$/h]	4545.84	5296.69

Characteristic values of the IEEE 9 bus test system and losses respectively, are as follows:

Table 4. Generator operating limits and quadratic cost function coefficients.

Bus	Pmin	Pmax	Qmin	Qmax	a	b	c
1	0.00	575.88	–200	300	0.01	0.30	0.20
2	0.00	100.00	–17	50	0.01	0.30	0.20
3	0.00	140.00	–10	60	0.01	0.30	0.20
6	0.00	100.00	–8	25	0.01	0.30	0.20
8	0.00	550.00	–140	200	0.01	0.30	0.20
12	0.00	410.00	–150	155	0.01	0.30	0.20

$$P_L = 307.3\,\text{MW}, \quad Q_L = 46.36\,\text{MVAR} \tag{14}$$

The problem is minimize the objective function

$$\text{Min}\Big\{f(P_G)\sum_{i=1}^{nG} f_i(P_G)\Big\} \tag{15}$$

Subject to constraints

$$P_{G1} + P_{G2} + P_{G3} + P_{G4} + P_{G5} + P_{G6} = 189.2\,\text{MW} \tag{16}$$

$$Q_{G1} + Q_{G2} + Q_{G3} + Q_{G4} + Q_{G5} + Q_{G6} = 189.2\,\text{MW} \tag{17}$$

$$\sum P_D = 189.2 \tag{18}$$

$$\sum Q_D = 107.2 \tag{19}$$

Table 5. Parameter values for GA

Population size	30
Number of generations	2500
Crossover probability	0.8
Mutation probability	0.05
Generation-elitism	5
Number of bit for encode active power generation	16

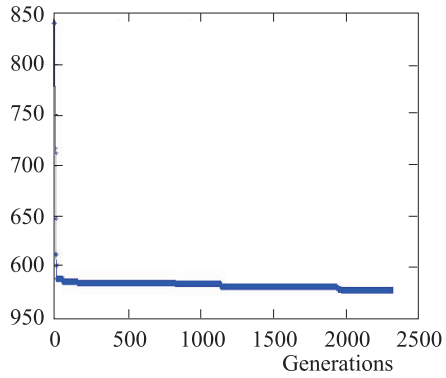

Fig. 6. Evolution of the GA (30 bus).

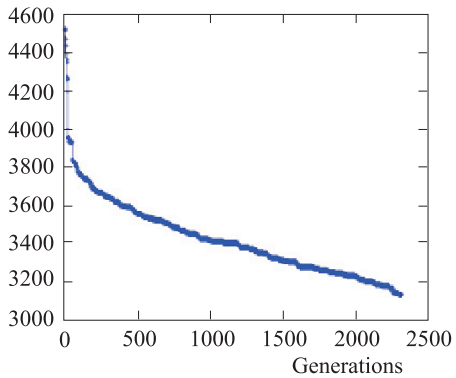

Fig. 7. Evolution of the GA (57 bus).

Table 6. Result of the study

methods	GA	Matpower
P_{G1}^{opt} [MW]	43.204228	41.53
P_{G2}^{opt} [MW]	58.642223	55.39
P_{G3}^{opt} [MW]	21.551666	16.19
P_{G4}^{opt} [MW]	37.439453	22.74
P_{G5}^{opt} [MW]	17.576327	16.25
P_{G6}^{opt} [MW]	13.684809	39.96
Q_{G1}^{opt} [MWAR]	–5.420000	–5.42
Q_{G2}^{opt} [MWAR]	1.730000	1.73
Q_{G3}^{opt} [MWAR]	35.920000	35.92
Q_{G4}^{opt} [MWAR]	34.200000	34.20
Q_{G5}^{opt} [MWAR]	6.960000	6.96
Q_{G6}^{opt} [MWAR]	31.660000	31.66
Coût [$/h]	576.629286	576.83

Characteristic values of the IEEE 57 bus test system are as follows:

Table 7. Generator operating limits and quadratic cost function coefficients.

Bus	**Pmin**	**Pmax**	**Qmin**	**Qmax**	**a**	**b**	**c**
1	0.00	575.88	–200	300	0.01	0.30	0.20
2	0.00	100.00	–17	50	0.01	0.30	0.20
3	0.00	140.00	–10	60	0.01	0.30	0.20
6	0.00	100.00	–8	25	0.01	0.30	0.20
8	0.00	550.00	–140	200	0.01	0.30	0.20
9	0.00	100.00	–3	9	0.01	0.30	0.20
12	0.00	410.00	–150	155	0.01	0.30	0.20

Losses are respectively

$$P_L = 2.86\,\text{MW}, \quad Q_L = 31.13\,\text{MVAR} \tag{20}$$

The problem is minimize the objective function

$$\text{Min}\Big\{f(P_G) = \sum f_i(P_G)\Big\} \tag{21}$$

Subject to constraints

$$P_{G1} + P_{G2} + P_{G3} + P_{G4} + P_{G5} + P_{G6} = 192.1\,\text{MW} \tag{22}$$

$$Q_{G1} + Q_{G2} + Q_{G3} + Q_{G4} + Q_{G5} + Q_{G6} = 105.1\,\text{MW} \tag{23}$$

Testing parameters for the IEEE 57 bus test system are as follows:

Table 8. Parameter values for GA.

Population size	60
Number of generations	2500
Crossover probability	0.8
Mutation probability	0.05
Generation-elitism	10
Number of bit for encode active power generation	32

Table 9. Result of the study

methods	GA	Matpower
P_{G1}^{opt} [MW]	296.94	265.33
P_{G2}^{opt} [MW]	100.00	100.00
P_{G3}^{opt} [MW]	140.00	140.00
P_{G4}^{opt} [MW]	99.99	100.00
P_{G5}^{opt} [MW]	343.88	276.97
P_{G6}^{opt} [MW]	100.00	100.00
P_{G7}^{opt} [MW]	189.11	287.56
Q_{G1}^{opt} [MWAR]	176.22	72.73
Q_{G2}^{opt} [MWAR]	49.99	50.00
Q_{G3}^{opt} [MWAR]	59.72	36.74
Q_{G4}^{opt} [MWAR]	24.99	6.37
Q_{G5}^{opt} [MWAR]	199.98	55.78
Q_{G6}^{opt} [MWAR]	–2.99	9.00
Q_{G7}^{opt} [MWAR]	–90.57	48.25
Coût [$/h]	3120.87	3176.39

6 CONCLUSION

GA as a solution to the economic dispatch problem of the IEEE 9 bus, 30 bus and 57 bus test system have been presented. Although genetic algorithms are generally considered to be offline optimisation algorithms, owing to the large amount of CPU time that need to converge to an optimal solution, they can exhibit very good online performance, when a suitable combination of operators is employed. The basic advantage of GA is that they can be very effectively coded to work on parallel machines. Potential application of GA include unit commitment and optimal power flow. With the recent advances in parallel computing, the on line solution optimal power flow with nonconvex generator cost functions may soon be possible.

References

[1] HOGAN, W. W.: Contract Networks for Electric Power Transmission, Journal of Regulatory Economics **4** (1992), 211–242.

[2] DOMMEL, H. W.: Optimal Power Dispatch, IEEE Transactions on Power Apparatus and Systems **PAS93** No. 3 (May/june 1974), 820–830.

[3] ALSAC, O.—BRIGHT, J.—PARIS, M.—STOTT: Further Developments in LP-Based Optimal Power Flow, IEEE Transaction of Power Systems **5** No. 3 (August 1990), 697–711.

[4] NANDA, J.—KOTHARI, D. P.—SRIVATAVA, S. C.: New Ptimal Power-Dispatch Algorithm Using Fletcher's Quadratic Programming Method, Procedings of the IEE **136** No. 3 (May 1989), 153–161.

[5] BJORNDAL, M.—JORNSTEN, K.: Zonal Pricing in a Deregulated Electricity Market, The Energy Journal **22** No. 1 (2001).

[6] ZIMMERMAN, R. D.—GAN, D.: MATPOWER — A MATLAB Power System Simulation Package, User's Manual, School of Electrical Engineering, Cornell University, 1997, available: http://www.pserc.cornell.edu/matpower/manual.pdf.

[7] HIMMELBLAU, D. M.: Applied Linear and Nonlinear programming, McGraw-Hill, New York, 1972.

[8] RAHLI, M.: Applied Linear and Nonlinear Programming to Economic Dispatch, PhD Thesis, Electrical Institute, Usto, Oran, Algeria, 1996.

[9] STAGG, G. W.—ABIADH, A. H. El: Computer Methods in Power System Analysis, New York, 1968.

[10] GOLDBERG, D. E.: Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley, Reading, 1989.

[11] GOLDBERG, D. E.: Sizing Populations for Serial and Parallel Genetic Algorithms, In J.D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, 1989, pp. 70–79.

[12] MIKAC, B.—INKRET, R.: Application of a Genetic Algorithm to the Availability-Cost Optimisation of a Transmission Network Topology, Proceedings ICANNGA'97 Third International Conference on Artificial Neural Networks and Genetic Algorithms, Norwich, U.K., Springer Verlag Wien, New York, 1997, pp. 306–310.

[13] HOLLAND, J. H.: Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, Michigan, 1975.

[14] MÜHLENBEIN, H.—SCHLIERKAMP-VOOSEN, D.: Predictive Models for the Breeder Genetic Algorithm, Evolutionary Computation **1** No. 1 (1993), 25–49.

[15] KAYNAK, O.—ZADEH, L. A.—TURKSEN, B.—RUDAS, I. J.: Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, 1998.

[16] TINNEY, W. F.—HART, C. E.: Power Flow Solution by Newton's Method, IEEE Transactions on Power Apparatus and Systems **PAS-86** No. 11 (Nov. 1967), 1449–1460.

[17] STOTT, B.—ALSAC, O.: Fast Decoupled Load Flow, IEEE Transactions on Power Apparatus and Systems **PAS-93** (June 1974), 859–869.

[18] van AMERONGEN, R.: A General-Purpose Version of the Fast Decoupled Loadflow, IEEE Transactions on Power Systems **4** No. 2 (May 1989), 760–770.

[19] MÉSZÁROS, C.: The Efficient Implementation of Interior Point Methods for Linear Programming and their Applications, PhD Thesis, Eötvös Loránd University of Sciences, 1996.

Received 19 September 2005

Mimoun Younes was born in 1965 in Sidi Belabbes, Algeria. He received his BS degree in electrical engineering from the Electrical Engineering Institute of The University of Sidi Belabbes (Algeria) in 1990, the MS degree from the Electrical Engineering Institute of The University of Sidi Belabbes (Algeria) in 2003. He is currently Professor of electrical engineering at The University of Sidi Belabbes (Algeria). His research interests include operations, planning and economics of electric energy systems, as well as optimization theory and its applications.

Mostefa Rahli was born in 1949 in Mocta Douz, Mascara, Algeria. He received his BS degree in electrical engineer-ing from the Electrical Engineering Institute of The University of Sciences and Technology of Oran (USTO) in 1979, the MS degree from the Electrical Engineering Institute of The University of Sciences and Technology of Oran (USTO) in 1985, and the PhD degree from the Electrical Engineering Institute of The University of Sciences and Technology of Oran (USTO) in 1996. From 1987 to 1991, he was a visiting professor at the University of Liege (Monte ore's Electrical Institute) Liege (Belgium) where he worked on Power Systems Analysis analysis under Professors Pol Pirotte and Jean Louis Lilien. He is currently Professor of electrical engineering at The University of Sciences and Technology of Oran (USTO), Oran, Algeria. His research interests include operations, planning and economics of electric energy systems, as well as optimization theory and its applications.

Lahouari Abdelhakem Koridak was born in 1966 in Oran, Algeria. He received his BS degree in electrical engineering from the Electrical Engineering Institute of The University of oran (Algeria) in 1993, the MS degree from the Electrical Engineering Institute of The University of Sidi Be-labbes (Algeria) in 2003. He is currently Professor of electrical engineering at The University of Sidi Belabbes (Algeria). His research interests include operations, planning and economics of electric energy systems, as well as optimization theory and its applications.