Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

GA Tuned Differential Evolution for Economic Load Dispatch with Non-convex Cost Function

Nidul Sinha[1], *MIEEE*; Y Ma[2], *IEEE Student Member* and Loi Lei Lai[2], *FIEEE*

Abstract — This paper proposes a genetic algorithm (GA) tuned differential evolution (DE) method for solving economic dispatch (ED) problem with non-smooth cost curves. The tuning of the weights in differential evolution is the key issue in designing an efficient differential evolution algorithm. Their values are dependent on nature and characteristic of objective function. As there is no explicit rule or guideline in determining these parameters, they are generally determined after a number of experimentations. In this paper floating point GA is used in tuning these parameters. The developed algorithm is experimented on a medium size of 40 units. The performance of the proposed algorithm is compared with standard Improved Fast Evolutionary Programming (IFEP) techniques. The simulation results demonstrate that GA tuned DE method is very efficient in finding higher quality solutions in high order non-convex ED problems.

Index Terms—Genetic Algorithm, Differential Evolution, Self-adaptive Evolutionary Programming, Economic Load Dispatch, Non-smooth Cost Function.

I. INTRODUCTION

Economic load dispatch (ELD) in electric power system is the task of allocating generation among the committed units such that the cost of production is minimized and constraints are satisfied at the same time. Improvements in scheduling the unit outputs can lead to significant savings in the cost of producing the required energy. Most of the conventional classical dispatch algorithms, employing the lambda-iteration method, the base point and participation factors method, and the gradient method [1], [2] consider the incremental cost curves to be of monotonically increasing or piece-wise linear nature. But most of the modern practical thermal units do have highly non-linear input-output characteristics because of valve point loadings, prohibiting operating zones etc resulting in multiple local minima in the cost function. Classical dispatch algorithms usually approximate the characteristics as quadratic ones to meet their requirements. However, such approximations may lead to huge loss of revenue over the time.

[1]Nidul Sinha is with the Department of Electrical Engineering, NIT, Silchar< Assam, India-788010, (email:nidulsinha@rediffmail.com).
[2]Y. Ma and Loi Lei Lai are with the Energy Systems Group, City University, London, UK. (emails: y.ma@city.ac.uk, l.l.lai@city.ac.uk).

The trend of considering more and more realistic features is on the rise especially after deregulation. Utilities are giving more stress on any efforts resulting into more savings. Consideration of highly nonlinear characteristics of the units demands for solution techniques having no restrictions on the shape of the fuel cost curves. The classical gradient-based techniques fail in solving these types of problems. Though dynamic programming (DP) [1] is capable of solving ED problems with inherently nonlinear and discontinuous cost curves but proves to suffer from intensive mathematical computations.

With nonlinear and non-differentiable objective functions, direct search approaches are the methods of choice. The best known of these are genetic algorithm (GA) [3]-[11], evolutionary strategy (ES) [12], [13], evolutionary programming (EP) [13]-[25] and differential evolution (DE) [26]-[31]. At the heart of every direct search method is a strategy that creates new solutions and some criterion to accept or reject the new solutions. While doing this all basic direct search methods use some greedy criteria. One of the greedy criteria is to accept a new solution if and only if it reduces the value of the objective function (in case of minimization) and the other may be forcing to create more new solutions nearer to already found better solutions. Although the greedy decision process converges fairly fast, it runs the risk of getting trapped in a local minimum. Inherently all parallel search techniques like genetic and evolutionary algorithms have some built-in safeguards like exploration to forestall misconvergence. EAs are more flexible and robust than conventional calculus based methods. Due to its high potential for global optimization, GA has received great attention in solving ELD problems. Walters and Sheble [3] reported a GA model that employed units' output as the encoded parameter of chromosome to solve an ELD problem with valve-point discontinuities. To enhance the performance of GA, Yalcinoz et al. [10] have proposed the real-coded representation scheme, arithmetic crossover, mutation, and elitism in the GA to solve more efficiently the ELD problem, and it can obtain a high-quality solution with less computation time. Though the GA methods have been employed successfully to solve complex optimization problems, recent research has identified some deficiencies in GA performance. EP differs from GA in two aspects: EP uses the control parameters (real values), but not their codings as in GAs; the generation selection procedure in EP are mutation and competition, but not reproduction, mutation and crossover as in GAs. Hence considerable computation time may be saved in EP. It has been reported

[8] that EP outperforms GAs. EP has seen a lot of developments for over four decades in terms of faster convergence rate and solution quality. Mutation in EP is often implemented by adding a random number or a vector from a certain distribution [e.g., a Gaussian distribution in the case of classical EP (CEP)] to a parent. The degree of variation of the Gaussian mutation is controlled by its standard deviation, which is also known as a 'strategy parameter' in evolutionary search. In the self-adaptation scheme of EP, this parameter is not prefixed; rather, it is evolved along with the objective variables. Experiments with self-adaptive EP have indicated efficient convergence to quality solutions [14]-[25]. Sinha et al. [21] and [22] have reported enhanced and better performance of improved fast EP (IFEP) algorithms, both cost based and self-adaptive ones, on non-convex power dispatch problems.

As most utilities demand that a practical optimization technique should fulfill three requirements. First, the method should be capable of finding near global minimum, regardless of the initial system parameter values. Second, it should converge very fast. And the third, the program should have minimum number of control parameters so that it will be easy to use. There is a method, which is not only astonishingly simple, but also reported to have performed extremely well on a wide variety of benchmark test problems. It is inherently parallel and hence lends itself to computation via a network of computers or processors. The basic strategy employs the difference of two randomly selected parameter vectors as the source of random variations for a third parameter vector. The method is called Differential Evolution.

The highlights of Differential Evolution (DE) are:

- A population of solution vectors are successively updated by addition, subtraction, and component swapping, until the population converges, hopefully to the optimum.
- No derivatives are used.
- Very few parameters to set.
- A simple and apparently very reliable method.

DE is reported [26]-[31] to be the only algorithm, which consistently found the optimal solution, and often with fewer function evaluations than the other direct search methods on benchmark nonlinear functions. The attributes for better performance of DE are:

- Simple vector subtraction to generate 'random' direction.
- More variation in population (because solution has not converged yet) leads to more varied search over solution space.
- Size and direction
- Annealing versus 'self-annealing'.

Very few works are reported on the performance of DE algorithm on highly nonlinear power system problems. The number of reported impressive performances of DE on benchmark mathematical functions has induced us in applying this method on highly nonlinear ELD problems. The key factor behind DE is a new scheme for generating trial parameter vectors. DE generates new parameter vectors by adding the weighted difference vector between two population members to a third member. The values of the weights play vital rule in the success of the algorithm. The weights are dependent on the nature and size of the objective functions. There is no explicit rule or guideline for proper choice of their values. Usually they are determined through experimentations for a particular problem. This has urged us to exploit GA approach in finding the better, if not the best, weights for DE approach. In view of the above, the main objectives of the present work are:

(i) To develop a program based on cost based improved fast EP (IFEP)[21] technique and study its performance in solving the same above problem.
(ii) To develop a program based on self-adaptive improved fast EP (IFEP$'$)[21] technique and study its performance in solving the non-convex ELD problem;
(iii) To develop a program based on floating point GA for tuning weights of DE algorithm and also to develop a program based on DE and study its performance in solving the above problem. Finally draw comparisons amongst the performances of the all the programs developed in steps (i), (ii), and (iii) above.

The rest of the paper is organized as follows: In Section II the mathematical problem formulation is briefly reviewed. DE approach is described in Section III. Algorithm on DE based ELD is outlined in Section VI. Section V presents the experimental results together with comparisons with other methods and discussions. Conclusions are drawn in Section VI.

II. PROBLEM FORMULATION

In this section the optimization problem is formulated as a minimization of summation of fuel costs of individual generators.

ELD Problem Formulation

The economic load dispatch problem can be described as an optimization (minimization) process with the following objective function:

$$\text{minimize}\, F = \sum_{j=1}^{n} FC_j(P_j) \tag{1}$$

where $FCj(Pj)$ is the fuel cost function of the jth unit and Pj is the power generated by the jth unit.

Subject to power balance constraints:

$$D = \sum_{j=1}^{n} Pj - P_L \tag{2}$$

where D is the system load demand and P_L is the transmission loss, and generating capacity constrains:

$$P_{j\,\min} \le P_j \le P_{j\,\max} \quad for\, j = 1,2,\, \ldots\, n \tag{3}$$

where P_{jmin} and P_{jmax} are the minimum and maximum power outputs of the *j*th unit .

The fuel cost function considering valve point loadings of the generating units are given as

$$FC_j(P_j)=a_jP_j^2+b_jP_j+c_j+|e_j\times\sin(f_j\times(P_{jmin}-P_j))| \quad (4)$$

where a_j,b_j,c_j are the fuel cost coefficients of the *j*th unit and e_j and f_j are the fuel cost coefficients of the *j*th unit with valve point effects.

The generating units with multi-valve steam turbines exhibit a greater variation in the fuel cost functions. The valve-point effects introduce ripples in the heat rate curves.

Now the fitness function, which is the sum of production cost and penalty for constraint violation, can be calculated for each individual of the parent population as

$$FIT_i = F + \sum_{z=1}^{N_c} PF_z \quad (5)$$

and $PF_z = \lambda_z \times [VIOL_z]^2$

where $VIOL_z$ is the violation of constraint z and λ_z is the penalty multiplier. N_c is the number of constraints. In the present case the limits of power output of the dependent unit is the constraint.

III. DIFFERENTIAL EVOLUTION (DE)

The DE algorithm is a population based algorithm like GAs using the similar operators; crossover, mutation and selection. The main difference between GA and DE is that GAs rely mostly on crossover while DE relies on mutation operation. The algorithm uses mutation operation as a search mechanism and selection operation to direct the search toward the prospective regions in the search space. Mutation in DE uses differences of randomly sampled pairs of solutions in the population and greediness may be embedded in it. The DE algorithm also uses a non-uniform crossover that can take child vector parameters from one parent more often than it does from others. By using the components of the existing population members to construct trial vectors, the recombination (crossover) operator efficiently shuffles information about successful combinations, enabling the search for a better solution space.

An optimization task consisting of n parameters can be represented by an n-dimensional vector. In DE, a population of N_p solution vectors is randomly created at the start. This population is successfully improved by applying mutation, crossover and selection operators.

3.1 DE BASED ECONOMIC DISPATCH (ED)

Let $p_i = [P_1, P_2, ...P_n]$ be a trial vector denoting the individual of a population to be evolved. The elements of the p_i are the real power outputs of the committed n generating units which are subjected to their respective capacity constraints in (3). To meet exactly the load demand in (2), a dependent unit is arbitrarily selected from among the committed n units. Let P_d be the power output of the dependent unit, then P_d is calculated by

$$p_d = D + P_L - \sum_{\substack{j=1 \\ \neq d}}^{n} P_j \quad (6)$$

In this work the power loss is not considered. However, it may be calculated by an iterative algorithm or by using directly B-loss matrix of the power system.

3.1.1 Initialization:

An initial population of N_p individuals having the form: $p_{iG} = [P_{1,i,G}, P_{2,i,G}, ...P_{n,i,G}]$, $i = 1, 2, \ldots N_p$ is generated. Where N_p, G, and n are the number of individuals, generation number and committed units respectively. Each individual is determined by setting the jth component $P_j \sim U(P_{jmin}, P_{jmax})$, for j=1,2,…n. $U(P_{jmin}, P_{jmax})$ denotes a uniform random variable ranging over $[P_{jmin}, P_{jmax}]$.

3.1.2 Creation of offspring

New solutions (offspring) are created based on both mutation and crossover as given below:

3.1.2.1 by mutation

Though there are number of variant mutation schemes in DE, the most promising one is considered here. In each generation, for individual $p_{i,G}$, the donor vector v_i is obtained using the mutation operation through

$$v_i = p_{i,G} + \lambda.(p_{best,G} - p_{i,G}) + F.(p_{r2,G} - p_{r3,G}) \quad (7)$$

Where the mutation factor F is a constant from [0, 2] and the vectors $p_{r2,G}$, and $p_{r3,G}$ are selected randomly such that the indices i, r_2 and r_3 are distinct from (1, 2, … N_p).

The idea behind the introduction of the additional control variable λ is to provide a means to enhance the greediness of the scheme by incorporating the current best vector $p_{best,G}$.

3.1.2.2 by crossover

Secondly, the trial vector u_i is obtained by crossover through

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } rand_{j,i} \le C_R \text{ or } j = I_{rand} \\ P_{j,i,G+1} & \text{if } rand_{j,i} > C_R \text{ and } j \ne I_{rand} \end{cases} \quad (8)$$

i= 1, 2, …..N_p, j= 1, 2, …..n

$rand_{j,i} \sim U[0,1]$, I_{raand} is a random integer from [1, 2, …. n], C_R is user defined number between [0,1] which is called crossing factor.

3.1.3 Selection

All solutions in the population have the same chance of being selected as parents without dependence of their fitness values. The child produced, after the mutation and crossover operations, is evaluated. Then, the performance of the child vector and its parent is compared and the better one is selected. If the parent is still better, it is retained in the population. The target vector, $p_{i,G+1}$ is selected after comparison of parents $p_{i,G}$ with their offspring $u_{i,G+1}$ as given below:

$$p_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } f(u_{i,G+1}) \le f(p_{i,G}) \\ p_{i,G} & \text{otherwise} \end{cases} \quad i = 1,2,...N_p \quad (9)$$

Mutation, crossing and selection continue until the stopping criterion is satisfied.

3.1.4 Stopping Rule

The iterative procedure of generating new trials by selecting those with minimum function values from the competing pool is terminated when there is no significant improvement in the solution. It can also be terminated when a given maximum number of generations (iterations) are reached. In the present work the latter method is employed.

VI ALGORITHM OF DE FOR THE TEST CASE

Step-1: The problem variables to be determined are represented as a n-dimensional trial vector, where each vector is an individual of the population to be evolved.
Step-2: An initial population of parent vectors, Q_i, for $i=1,2,\ldots N_p$, is selected at random from the feasible range in each dimension. The distribution of these initial parent vectors is uniform.
Step-3: Calculate the fitness value of each individual in the population using the evaluation function given by (5).
Step-4: An offspring ($Q_i^{'}$) is generated from the parent by using Eqs. (7) & (8) through mutation and crossover.
Step-5: Fitness function, FIT_i is evaluated for each individual of both parent and child populations. Update each individual's fitness value p_{best}.
Step-6: Comparison is made between N_p parents and N_p offspring and better ones are chosen as the target vector (new parent vectors) in the next generation.
Step-7: If current generation is greater than or equal to the maximum generation, print the result and stop; otherwise repeat the steps 3 to 6.

The algorithms are implemented in Matlab command line for solution of two test cases of non-convex economic load dispatch. The programs were run on a 3.2 GHz, Pentium-IV, with 512 MB RAM PC.

V. NUMERICAL TESTS

The performance of the proposed algorithms are experimented on a 40-unit case which has been adopted from [7] with modifications to incorporate the effects of valve point loadings. Data is shown in Table 1.

Table 1: Units data for the test case with load 10500MW (with valve point loadings)

Generator	P_{min}(MW)	P_{max}(MW)	a	b	c	e	f
1	36	114	0.00690	6.73	94.705	100	0.084
2	36	114	0.00690	6.73	94.705	100	0.084
3	60	120	0.02028	7.07	309.54	100	0.084
4	80	190	0.00942	8.18	369.03	150	0.063
5	47	97	0.0114	5.35	148.89	120	0.077
6	68	140	0.01142	8.05	222.33	100	0.084
7	110	300	0.00357	8.03	287.71	200	0.042
8	135	300	0.00492	6.99	391.98	200	0.042
9	135	300	0.00573	6.60	455.76	200	0.042
10	130	300	0.00605	12.9	722.82	200	0.042
11	94	375	0.00515	12.9	635.20	200	0.042
12	94	375	0.00569	12.8	654.69	200	0.042
13	125	500	0.00421	12.5	913.40	300	0.035
14	125	500	0.00752	8.84	1760.4	300	0.035
15	125	500	0.00708	9.15	1728.3	300	0.035
16	125	500	0.00708	9.15	1728.3	300	0.035
17	220	500	0.00313	7.97	647.85	300	0.035
18	220	500	0.00313	7.95	649.69	300	0.035
19	242	550	0.00313	7.97	647.83	300	0.035
20	242	550	0.00313	7.97	647.81	300	0.035
21	254	550	0.00298	6.63	785.96	300	0.035
22	254	550	0.00298	6.63	785.96	300	0.035
23	254	550	0.00284	6.66	794.53	300	0.035
24	254	550	0.00284	6.66	794.53	300	0.035
25	254	550	0.00277	7.10	801.32	300	0.035
26	254	550	0.00277	7.10	801.32	300	0.035
27	10	150	0.52124	3.33	1055.1	120	0.077
28	10	150	0.52124	3.33	1055.1	120	0.077
29	10	150	0.52124	3.33	1055.1	120	0.077
30	47	97	0.01140	5.35	148.89	120	0.077
31	60	190	0.00160	6.43	222.92	150	0.063
32	60	190	0.00160	6.43	222.92	150	0.063
33	60	190	0.00160	6.43	222.92	150	0.063
34	90	200	0.0001	8.95	107.87	200	0.042
35	90	200	0.0001	8.62	116.58	200	0.042
36	90	200	0.0001	8.62	116.58	200	0.042
37	25	110	0.0161	5.88	307.45	80	0.098
38	25	110	0.0161	5.88	307.45	80	0.098
39	25	110	0.0161	5.88	307.45	80	0.098
40	242	550	0.00313	7.97	647.83	300	0.035

Parameters for the algorithms:
Population Size = 60
Maximum Iterations = 1000
Penalty multiplier = 100.

Parameters for DE

$$\left.\begin{array}{l}\lambda = 0.7398 \\ F = 1.2000\end{array}\right\} \textit{GA optimized}$$

Floating point GA (GAF) features used:
(i) Heuristic crossover
(ii) Uniform mutation
(iii) Normalized geometric select function

The GA optimization toolbox GAOT in Matlab proposed by C.R. Houck et al. [11] is used after minor modification for tuning the weights of the DE algorithm Uniform

mutation [11] randomly selects one variable, j, and sets it equal to an uniform random number U(P_{jmin} , P_{jmax}).
Heuristic crossover [11] performs extrapolation along the line formed by the two parents outward in the direction of the better fit parent. It utilizes the fitness information. A new individual is created using Eq. (10) where r = U(0,1) and $\overline{X}$ is better than $\overline{Y}$ in terms of fitness.

$$\overline{X}' = \overline{X} + r(\overline{X} - \overline{Y}) \quad (10)$$

$$\overline{Y}' = \overline{X} \quad (11)$$

The convergence characteristics of the algorithms are shown in Fig.1.

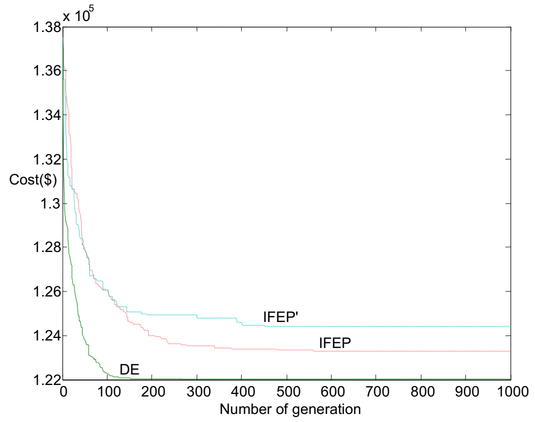

Fig.1: The convergence performance of IFEP, IFEP′ and DE algorithms.

DE has the higher capability to escape local minima and converge to better quality solutions.

To investigate the effects of initial trial solutions all the algorithms were run with 50 different initial trial solutions for 1000 iterations per run and the results are reported in Tables 2. In these results the average cost is the prime indicator of competence of an algorithm in finding better quality solutions. It is observed that DE is more competent.

Table 2: Statistical test results of 50 runs with different initial solutions (with non-smooth cost curves)

Method	Average cost (Rs.)	Maximum cost (Rs,)	Minimum cost (Rs.)
IFEP	123326.03	124787.42	122483.81
IFEP’	124313.66	125291.56	123387.83
DE	122569.28	124255.83	121808.06

DE appears to be the most efficient in terms of faster convergence rate and quality of solution, which makes it to be much efficient in finding the global optimum.

VII. CONCLUSIONS

Algorithms based on IFEP, IFEP′, and DE are developed and their performances are tested on non-convex economic load dispatch problems with valve point loading effects; The weights, that determine the right blending of random direction with greediness in DE, are tuned using floating point GA. Experimental results reveal that all the algorithms are competent to provide better quality solutions.

VIII.REFERENCES

1. Liang, Zi-Xiong and Glover, J. Duncan, “A zoom feature for a dynamic programming solution to economic dispatch including transmission losses,” *IEEE Trans. on Power Systems*, Vol.7, No.2, May 1992, pp.544-549.
2. Wood, A.J., Wollenberg, B.F., *Power Generation, Operation and Control*, second edition, Wiley, New York, USA, 1996.
3. Wong, K.P. and Wong, Y.W., “Thermal generator scheduling using hybrid genetic/simulated annealing approach,” *IEE Proc. Part-C*, Vol.142, No.4, July 1995, pp.372-380.
4. Walter, D.C. and Sheble, G.B., “Genetic algorithm solution of economic dispatch with valve point loading,” *IEEE Trans. on Power Systems*, Vol.8, No.3, August, 1993, pp.1325-1332.
5. Bakirtzis, A. Petridis, V. and Kazarlis, S., “Genetic algorithm solution to the economic dispatch problem,” *IEE Proc. Part-D*, Vol. 141, No. 4, July 1994, pp. 377–382.
6. Sheble, G.B. and Brittig, K., “Refined genetic algorithm — Economic dispatch example,” *IEEE Trans. on. Power Systems,* Vol. 10, Feb. 1995, pp. 117–124.
7. Chen, P.H. and Chang, H.C., “Large-scale economic dispatch by genetic algorithm,” *IEEE Trans. on Power Systems*, Vol.10, No.4, November 1995, pp.1919-1926.
8. Fogel, D.B., “A comparison of evolutionary programming and genetic algorithms on selected constrained optimization problems,” *Simulation*, June 1995, pp.397-404.
9. Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, MA, 1989.
10. Yalcionoz, T., Altun, H., and Uzam, M., “Economic dispatch solution using a genetic algorithm based on arithmetic crossover,” *Proc. Power Tech. Conf.*, IEEE, Portugal, Sept. 2001.
11. Houck, C.R., Joines, J.A., and Kay, M.G., “A genetic algorithm for function optimization: A Matlab implementation,” Technical Report NCSU-IE TR 95-09, North Carolina State University, 1995.
12. Bäck, Th. and Schwefel, H.P., “An overview of evolutionary algorithms for parameter optimization,” *Evolutionary Computation*, Vol.1, No.1, 1993, pp.1-23.
13. Fogel, D.B., *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ, 1995.
14. Fogel, D.B., “An introduction to simulated evolutionary optimization,” *IEEE Trans. on Neural Networks*, Vol.5, No.1, 1994, pp.3-14.
15. Chellapilla, K., “Combining mutation operators in evolutionary programming,” *IEEE Trans. on Evolutionary Computation*, Vol.2, No.3, 1998, pp.91-96.
16. Wolpert, D.H. and Macready, W.G., “No free lunch theorems for optimization,” *IEEE Trans. on Evolutionary Computation*, Vol.1, No.1, 1997, pp.67-82.
17. Fogel L.J., Fogel D.B. and Angeline P.J., “A preliminary investigation on extending evolutionary programming to include self-adaptation on finite state machines,” *Informatica*, 18, 1994, pp.387-398.
18. Ma, J.T. and Lai, L.L., “Evolutionary programming approach to reactive power planning,” *IEE Proc. Part-C*, Vol.143, No.4, July 1996, pp. 365-370.
19. Yang, H.T., Yang, P.C. and Huang, C.L., “Evolutionary programming based economic dispatch for units with non-smooth fuel cost functions,” *IEEE Trans. on Power Systems*, Vol.11, No.1, February 1996, pp.112-118.

20. Yao, X. , Liu, Y. and Lin, G., "Evolutionary programming made faster," *IEEE Trans. Evolutionary Computation*, Vol. 3, July 1999, pp. 82–102.
21. Sinha, Nidul, Chakrabarti, R. and Chattopadhyay, P.K., "Evolutionary programming techniques for economic load dispatch," *IEEE Trans. on Evolutionary Computation*, Vol.7, No.1, February, 2003, pp.83-94.
22. Sinha, Nidul, Chakrabarti, R. and Chattopadhyay, P.K., "Fast Evolutionary programming techniques for short-term hydrothermal scheduling," *IEEE Trans. on Power Systems*, Vol.18, No.1, February 2003, pp.214-220.
23. Fogel L.J., Owens A.J. and Walsh M.J., *Artificial Intelligence through Simulated Evolution*, John Wiley, NY, 1966.
24. Lai, L.L., "Evolutionary programming for economic load dispatch of units with non-smooth input-output characteristic functions," Chapter 8 of *Intelligent System Applications in Power Engineering – Evolutionary Programming and Neural Networks*, John Wiley & sons, UK, July, 1998.
25. Yao, X. and Liu, Y., "Fast evolutionary programming," *Proc. 5th Annu.Conf. Evolutionary Programmin*g, L. J. Fogel, T. Bäck, and P. J. Ange-line, Eds. Cambridge, MA, 1996, pp. 451–460.
26. Storn, R., "System design by constraint adaptation and differential evolution," *IEEE Trans. on Evolutionary Computation*, Vol. 3, No. 1, 1999, pp. 22-34.
27. Storn, R. and Price K., "Minimizing the real functions of the ICEC'96 contest by differential evolution," *Int. Conf. Evolutionary Computation*, IEEE, 1996, pp.842-844.
28. Price K., "Differential Evolution: A Fast and Simple Numerical Optimizer," NAFIPS 1996, Berkeley, pp.524-527.
29. Ursem, R.K., and Vadstrup, P., "Parameter identification of induction motors using differential evolution," *Proceedings of the Fifth Congress on Evol. Comp*. CEC-03, IEEE, 2003, pp.790 –796.
30. Paterlini, S., and Krink, T., "High performance clustering using differential evolution," *Proceedings of the Six Congress on Evol. Comp.*, CEC-04, IEEE, 2004.
31. Huse E.S., "Power generation scheduling- a free market based procedure with reserve constraints included," Ph.D. thesis, Norwegian University of Science and Technology, Norway, 1998.