International Journal of Electrical and Power Engineering 2 (2): 85-91, 2008
ISSN: 1990-7958

Genetic Algorithm with Improved Lambda-Iteration Technique to Solve the Unit Commitment Problem

V. Senthil Kumar and M.R. Mohan
Department of Electrical and Electronics Engineering, College of Engineering,
Anna University, Chennai 600025, Tamil Nadu, India

Abstract: This study presents a two -layer approach to solve the Unit Commitment (UC) problem. The first layer uses a Genetic Algorithm (GA) to decide the on/off status of the units. The second layer uses an Improved Lambda -Iteration (ILI) technique to solve the Economic Dispatch (ED) problem while satisfying all the plant and system constraints. GA's are general -purpose optimization technique based on principle of natural selection and natural genetics. In order to deal effectively with the constraints involved in the UC problem, a repair and approximate genetic operators were introduced. The proposed method is tested and compared with Lagrangian Relaxation (LR) and GA on the systems with the number of generating units in the range of 10-100. The simulation results reveal that the features of easy implementation, convergence within an acceptable execution time and highly optimal solution in solving UC problem can be achieved.

Key words: Unit commitment, repair genetic algorithm, economic dispatch and lambda iteration

INTRODUCTION

A thermal generation scheduling problem is the tasks of finding an optimal schedule for each thermal unit over a time horizon. This calculated schedule helps to determine when to start and when to shut down units such that the total operating cost could be minimized. A standard thermal generation scheduling problem is often formulated subjected to several constraints that includes power balance constraint, spinning reserve constraint, generation limit constraint and minimum up -time and down-time constraints.

This problem is quite difficult due to its inherent high dimensional, large scale, mixed -integer combinatorial optimization problem with constraints. The exact solution to the problem can be obtained only by complete enumeration (Wood and Wollenberg, 1996). A survey of literature on the UC methods reveals that various numerical optimization techniques have been employed to approach the UC problem. Specifically: priority list (Sheble, 1990), dynamic programming (Snyder *et al*., 1987; Lowery, 1983; Su and Hsu, 1991; Ouyang and Shahidehpour, 1991), lagrangian relaxation (Merlin and Sandrin, 1983; Zhuang and Galiana, 1988), mixed-integer programming (Muckstadt and Wilson, 1968) and branch-and -bound (Cohen and Yoshimura, 1983).

Recently simulated annealing (Zhuang and Galiana, 1990), expert systems (Wang and Shahidehpour, 1992), artificial neural networks (Ouyang and Shahidehpour, 1992; Sasaki *et al*., 1992) and Genetic Algorithms (GAs) (Goldberg, 1989; Kazarlis *et al*., 1996; Rudoll and Bayrleithner, 1999; Xing and Wu, 2002; Arroyo and Conejo, 2002; Cheng and Liu, 2000) have also been used for the solution of the UC problem. These methods can accommodate more complicated constraints and are claimed to have better solution quality. GA's are a general-purpose stochastic and parallel search method based on the mechanics of natural selection and natural genetics. GA's are a search method to have potential of obtaining near-global minimum.

Six basic methods have been reported in the literatures that enable GA's to be applied to constrained optimization problem (Vassilio *et al*., 1998). This study uses method 3 as a first layer in which an invalid solution is approximated and repaired to become a valid one. In GA the dynamic penalty method improves the convergence criteria but as the number of iteration increases the penalty term included may be very high. As a result, the best solutions may be discarded from the population. The operator, which is used for repair and approximations, reduces the time as well as brings the solutions to the near optimal solutions.

A repair and approximate algorithm presents three advantages compared to penalty based genetic algorithms: It does not work on a broad search space full of infeasible solutions, but on bounded search spaces (consisting of feasible solutions), thus reducing the search burden and increasing the efficiency of the

Corresponding Author: V. Senthil Kumar, Department of Electrical and Electronics Engineering, College of Engineering, Anna University, Chennai 600025, Tamil Nadu, India

algorithm, the problem of choosing penalties of different nature for each of the constraints disappears (Arroyo and Conejo, 2002) and possibility of neglecting the best solutions which are infeasible regarding the violation of constraints can be avoided. The second layer uses a novel ILI technique to solve the Economic Dispatch (ED) problem. ILI technique avoids premature convergence involved in solving the ED problem using lambda-iteration technique.

PROBLEM FORMULATION

The objective of the UC problem is to minimize the total production cost over the scheduling period. Therefore, the objective function is expressed as the sum of fuel and start-up costs of the generating units. The UC problem can be mathematically formulated as:

$$\min \ TPC = \sum_{i=1}^{N} \sum_{i=1}^{H} [F_i (P_{ih}) + ST_i (1-U_{i(h-1)})] U_{ih} \quad (1)$$

$$F_i (P_{ih}) = a_i P^2_{ih} + b_i P_{ih} + c_i \quad (2)$$

Due to the operational requirements, the minimization of the objective function is subjected to the following constraints:

- Power balance constraints

$$\sum_{i=1}^{N} P_{ih} U_{ih} \geq D_h \quad (3)$$

- Spinning reserve constraints

$$\sum_{i=1}^{N} U_{ih} P_{i(max)} D_h + R_h \quad (4)$$

- Generation limit constraints

$$P_{i(min)} \leq P_{ih} \leq P_{i(max)} \quad (5)$$

- Minimum up-time constraints

$$U_{ih} = 1 \quad \text{for} \sum_{t=h-down_i}^{h-1} U_{it} < up_i \quad (6)$$

- Minimum down -time constraints

$$U_{ih} = 1 \quad \text{for} \sum_{t=h-down_i}^{h-1} (1-U_{it}) < down_i \quad (7)$$

- Start-up/shut-down cost

$$ST_i = \begin{cases} HSC_i, & \text{if} \quad down_i \leq CSH_i \\ CSC_i, & \text{otherwise} \end{cases}$$

$$Sd_i = 0 \quad \text{For all units}$$

GENETIC ALGORITHMS

GA's are inspired by the study of genetics. They are conceptually based on natural evolution mechanisms working on populations of solutions in contrast to other search techniques that work on a single solution. The most interesting aspect of GA's is that all though they do not require any prior knowledge or space limitations, such as smoothness or convexity of the function to be minimized they exhibit very good performance on the majority of the problems applied.

At first a population of M solutions are generated at random, encoded in strings (genotypes) of symbols resembling natural chromosomes. Each member of the population is then decoded to a real problem solution and a "fitness" value is assigned to it by a quality function that gives a measure of the solution quality. With the initial population produced and evaluated, genetic evolution takes place by means of three genetic operators:

Reproduction: Two genotypes are selected from the parent population with a probability, which is proportional to their fitness using Roulette wheel parent selection algorithm.

Crossover: If a probability test is passed, the two genotypes are combined (exchange bits) to from a new genotype, which incorporates characteristics from both parent genotypes. The produced genotype (offspring) is included to the next generation's population.

Mutation: With a small probability, random bits of the offspring genotype flip from 0 to 1 and vice versa to give characteristics that don't exist in the parent population.

When M new solution strings are produced, they are considered as a new generation and they totally replace the parents in order for the evolution to proceed. Many generations are needed for the population to converge to the optimum or a near-optimum solution, the number increases according to the problem difficulty.

GENETIC ALGORITHMS FOR UNIT COMMITMENT PROBLEM

Implementation of the Proposed Genetic Algorithm (PGA) for UC problem includes the following stages.

Initial population: A number of S_M (say M = 500) initial binary coded solutions (genotypes) are generated, by committing all the units. Then, a random decommitment is done with some probability. Each schedule is checked for minimum up/down time, demand and spinning reserve constraints. A fitness score is assigned based upon their objective function for the solutions, which satisfy all the constraints. The objective function associated with each solution is calculated by economically dispatching the hourly load to the operating units using improved lambda iteration technique as explained in this study. Based upon the fitness values each solution is ranked in descending order and the best S_k (k = 50) solutions are considered as parents. The fitness score F associated with each solution is:

$$F(S_m) = \text{Total fuel cost} + \text{start up/shut-down cost}$$

Reproduction: The reproduction operator is a prime selection operator. Two genotypes are selected using Roulette wheel parent selection algorithm that selects a genotype with a probability proportional to genotypes relative fitness within the population. Then, a new offspring genotype is produced by means of the two basic genetic operators namely crossover and mutation.

Crossover: To get the new patterns of genetic strings during the evolution process, two levels of crossover operation, i.e. unit level crossover and population level crossover are introduced. Both type of crossover is done with fixed probability of 0.7.

Unit level crossover: A good scheduling could be expected by exchanging the scheduling periods of the units within the genotype. Since the partial string of genotype has no fitness function value, the selection processes are performed randomly with certain probability. Here two types of unit level crossover is implemented, Type 1 is by keeping the first half of the strings and exchanging the second half of the strings with randomly selected units. Type 2 is keeping the second half of the strings and exchanging the first half of the strings with randomly selected units to get a better scheduling.

An example of 10 units and 24 h schedule is considered.
Type 1: Between unit 1 and unit 9
Type 2: Between unit 2 and unit 10
Before crossover

```
Unit 1    111000110110101011010010001
Unit 2    100010111101000011011010 10
                  .
                  .
Unit 9    010100010110011101011011 00
Unit 10   010111010100010101010011 00
```

After crossover

```
Unit 1    111000110110011101011011 00
Unit 2    010111111101000011011010 10
                  .
                  .
Unit 9    010100010110101011010100 01
Unit 10   100010010100010101010011 00
```

Population level crossover: This operator is applied with certain probability. When applied, the parent genotypes are combined to form two new genotypes that inherent solution characteristics from both parents. In the opposite case the offspring are identical replications of their parents. Crossover is done between the parent genotypes obtained from roulette wheel parent selection. The crossover scheme used is single -point crossover.

Mutation: Mutation introduces new genetic material into the gene at some low rate. With a small probability, randomly chosen bits of the offspring genotypes change from '0' to '1' and vice versa.

Swap mutation operator: In this operator, based upon the full -load average production cost of the generating units, units are ranked and are arranged in descending order. Here full load average production cost could be obtained by calculating the net heat rate at full load times the fuel cost. If the full -load production cost of the ith unit is lesser than the full -load production cost of the jth unit and the status of the units are OFF and ON, respectively, then the status of ith and jth units were exchanged. For each scheduled hours this procedure is followed with some probability to avoid local convergence. This helps in reducing the total operating cost of the system (Jorge and Smith, 1999).

Repair operator for up-time/down-time: This operator repairs the solutions that are infeasible regarding minimum up/down constraints. The state of a unit is evaluated starting from hour '0'. If at a given time 't' the minimum up or down time constraint is violated, the state (on/off) of the unit at that hour is reversed and updated. The process continues until the last hour.

For example if the minimum up/down time is 4 h for a particular unit than,

```
Before repair operation        After repair operation
001010101011011001011001 001111100001111000011111
```

Demand/spinning reserve approximations: This operator approximates the solutions that are infeasible regarding demand/spinning reserve constraints. For each hour

demand/spinning reserve constraint is checked. If it is not satisfied, the schedule for that particular hour is retained from the previous generation, which is again a satisfied solution.

Selection: The entire population, including parent and offspring are arranged in descending order. The best K solutions, which survive are transcribed along with their elements to form the basis of the next generation.

The above procedure is repeated until the given maximum generation count is reached. Here for all systems maximum of 500 generations is considered. The above algorithm produces only the feasible solutions, which totally avoids penalizing of solutions.

IMPROVED LAMBDA-ITERATION TECHNIQUE

In GA based unit commitment problem, populations are created randomly with binary code 1's and 0's, which shows the on -off status of the generating units. The necessary condition for the existence of a minimum cost -operating condition for the thermal power system is that the incremental cost (λ) of all the committed units be equal. Each generating unit has different fuel cost coefficients. Finding the optimum incremental cost for the randomly committed units is difficult. This study explains the procedure for finding the optimum incremental cost and the problem incurred in finding the optimum incremental cost for the randomly committed units and the solution for the same.

The total fuel cost is computed as the sum of the hourly fuel costs by economically dispatching the load demand to the operating units for every hour of the scheduling period, which is a sub-problem in the unit commitment problem. This could be solved using lambda-iteration technique. The lambda-iteration technique converges very rapidly for simple type of optimization problems. The actual computational technique is lightly more complex, since it is necessary to observe the operating limits on each of the units during the course of the computation (Wood and Wollenberg, 1996).

We assume an incremental cost rate and find the power outputs for each of the N generating units for this value of incremental cost. Our first estimate may be incorrect. If the assumed value of incremental cost is such that the total power is too low, λ is increased and try another solution. With two solutions, we can interpolate the two solutions to get closer to the desired value of total received power. By keeping track of the total demand versus the incremental cost, we can rapidly find the desired operating point.

To illustrate the lambda -iteration technique following case studies are considered:

The test system in case 1 comprises of two generating units, unit 1 and unit 2 with the demand of 700MW (Kazarlis *et al*., 1996). Using lambda-iteration technique involving interpolation, the problem is solved in six iterations as shown in Table 1. In this calculation, the value for λ on the second iteration is always set at 5% above or below the starting value depending on the sign of error, for the remaining iterations, lambda is projected as shown in Fig. 1. From Fig. 1 it is observed that a lambda value of 17.0015 with an error of -95 is considered as an initial value and the second estimate is 5% of the initial value, which is 17.8516 with an error of 210. Third lambda value is obtained by interpolating between the above two points which is 17.2663 with an error of -95. This procedure is continued until convergence is obtained.

Here λ^0 -starting lambda, total received power = Σ P and error = total received power -demand. Mathematically lambda projection can be formulated as follows:

$$\lambda^{k+1} = \lambda^{k} \pm \{[(\lambda^{k} - \lambda^{k-1})/(S^{k} - S^{k-1})] \times (D_h - S^{k})\} \; ; K = 1, 2,..., ITmax$$

Initial guess for λ^0:

$$P_i = (D_h \times P_{i\,(max)})/\Sigma P_{i\,(max)}$$

$$\lambda^0 = \Sigma(2a_iP_i + b_i)/N.$$

Where,

k = Represents the iteration count.
S = Represents the total received power.
D_h = Represents the demand for the h-th hour.
P_i = Represents the generation output for the i-th unit.

In case 2, three generating units were considered, unit 1, unit 2 and unit 5 for the demand of 850MW. Using lambda-iteration technique involving interpolation, it is observed that in the 3rd and 4th iterations the total received power is same. The reason being, for a change in the lambda value change in the total received power is zero, because of the operating limits involved in each unit. Though the number of iterations increases lambda value remains same and it falls into premature convergence (infeasible) as shown in Table 2. The lambda projection for case 2 (infeasible) is shown in Fig. 2. From Fig. 2 it is observed that, while interpolating between second lambda value of 17.3351 with an error of -220 and third lambda value of 17.9932 with an error of 85, the 4th lambda value obtained is 17.8098 with an error value of 85. Since the error value remains same for a change in the lambda value it reaches the premature convergence.

Table 1: Economic dispatch solution for case 1

Iterations	λ	Lambda ($/MWh)	Total received power (MW)	Unit1 P(MW)	Unit 2 P(MW)	Error (MW)
1	λ^0	17.0015	605	455	150	-95
2	λ^1	17.8516	910	455	455	210
3	λ^2	17.2663	605	455	150	-95
4	λ^3	17.4486	759.19	455	304.19	59.19
5	λ^4	17.3786	646.29	455	191.29	-53.71
6	λ^5	17.4119	700.00	455	245	0.0

Table 2: Economic dispatch solution for case 2 (infeasible)

Iterations	λ	Lambda ($/MWh)	Total received power (MW)	Unit1 P(MW)	Unit 2 P(MW)	Unit 5 P(MW)	Error (MW)
1	λ^0	18.2475	935	455	455	25	85
2	λ^1	17.3351	630	455	150	25	-220
3	λ^2	17.9932	935	455	455	25	85
4	λ^3	17.8098	935	455	455	25	85
5	λ^4	17.8098	935	455	455	25	85

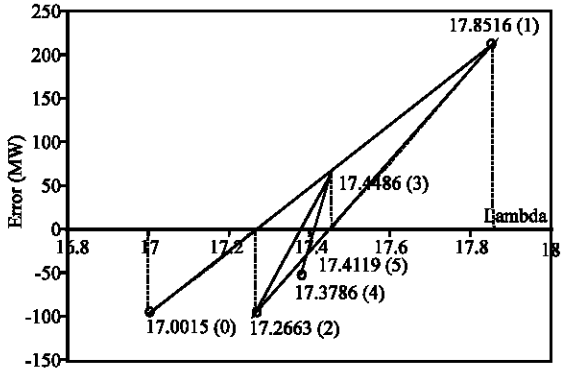

Fig. 1: Lambda projections for case 1

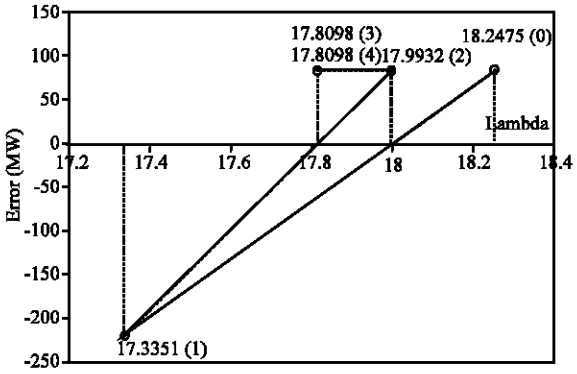

Fig. 2: Lambda projection for case 2 (infeasible)

This problem could be solved using Simple Lambda Iteration (SLI) technique without including interpolation i.e., by increasing or decreasing a small value of lambda based upon the sign of error. Which takes more number of iterations.

$$\lambda = \lambda^0 \pm \Delta\lambda$$

Table 3: Dispatch result of three-unit system with 850MW (feasible solution)

Iterations	λ	Lambda ($/MWh)	Total received power (MW)	Unit1 P(MW)	Unit 2 P(MW)	Unit 5 P(MW)	Error (MW)
1	λ^0	18.2475	935	455	455	25	85
2	λ^1	17.3351	630	455	150	25	-220
3	λ^2	17.9932	935	455	455	25	85
4	λ^3	17.8098	935	455	455	25	85
5	λ^4	17.7208	935	455	455	25	85
6	λ^5	17.6322	935	455	455	25	85
7	λ^6	17.5440	935	455	455	25	85
8	λ^7	17.4563	796.595	455	316.595	25	-53.404
9	λ^8	17.4901	851.188	455	371.189	25	1.1884
10	λ^9	17.4894	850.001	455	370.001	25	0.0009

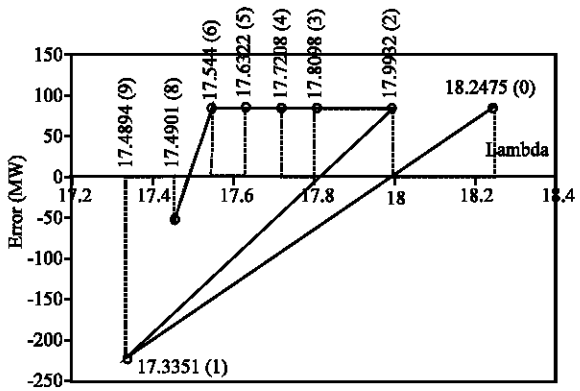

Fig. 3: Lambda projection for case 2 (Feasible)

Where:

$$\Delta\lambda = 0.005 \text{ times } \lambda^0$$

To avoid the premature convergence and increase in the number of iterations an improved lambda -iteration technique is developed, in which the lambda value is varied locally until an error value is obtained. Once an error value is obtained it reaches the convergence within few number of iterations as shown in Table 3. From 4th iteration to 7th iteration lambda value is varied locally until there is a change in error value. In 10th iteration it had obtained the convergence. The lambda projection for case 2 (feasible) is shown in Fig. 3. From Fig. 3, it is observed that the error value remains same in 3rd and 4th iterations. To avoid this, the lambda value is varied locally until there is a change in the error value. In 8th iteration, change in the error value is obtained and the convergence is obtained in 10th iteration.

RESULTS

A ten-unit system data and load demands are considered for case study (Kazarlis *et al*., 1996). The 20, 40, 60, 80 and 100 unit's data are obtained by duplicating

Table 4: Simulation results for up to 100-unit systems

	LR solution	GA solution	PGA		
			Best cost	Worst cost	
Units	Operating cost $		Operating cost $		Difference (%)
10	565,825	565,825	564,367.5	564,367.5	0.0
20	1,130,660	1,126,243	1,124,893.3	1,125,072.1	0.02
40	2,258,503	2,251,911	2,245,827.7	2,249,609.5	0.17
60	3,394,066	3,376,625	3,368,537.5	3,375,491.25	0.20
80	4,526,022	4,504,933	4,497,871.5	4,507,177.5	0.21
100	5,657,277	5,627,437	5,622,746.0	5,638,858.0	0.28

Table 5: Comparison of average number of iterations of ILI and SLI

Units	ILI (iterations)	SLI (iterations)
10	141	2968
20	144	3273
40	151	5222
60	166	8033
80	154	9059
100	153	10649

Table 6: Comparison of average CPU time (Sec) of PGA using ILI and SLI

Units	PGA with ILI (sec)	PGA with SLI (sec)
10	11.75	67.75
20	20.968	152.562
40	42.796	407.515
60	67.078	845.937
80	97.406	1165.72
100	127.343	1807.81

the base case (10 units) with a 24 h demand schedule, whereas the load demands are adjusted in proportion to the system size. In the simulation, the spinning reserve is required to be 10% of the load demand. In order to avoid misleading results due to the stochastic nature of the PGA, 20 runs were made for each problem set, with each run starting with different random populations. For a specific problem set, every one of the 20 runs was terminated at the same generation limit. The population size was 50 genotypes in all runs. The simulation was carried out on a Pentium IV, 2 GHz processor.

The test results are shown in Table 4. For the PGA, both the best and worst solutions produced are reported together with their difference as a percentage of the best solution. The results reported in PGA represents the average of the entire population across 20 runs. The difference between the average best and the average worst runs is calculated to indicate the likelihood that the PGA will reproduce the same range of solutions. In comparison with the LR and GA as reported by the references the PGA method obviously displays a satisfactory performance (Kazarlis *et al*., 1996; Cheng *et al*., 2000).

Table 5 shows the comparison of average number of iterations of ILI and SLI in solving the ED problem for the units up to 100 and 24 h schedule. Comparison reveals the effectiveness of the algorithm and also it helps in reducing the execution time for large-scale problems.

As the number of units increases the time increases linearly. Table 6 gives the average time in 20 trial runs for unit's up to 100 and 24 h schedule, while comparing PGA using ILI and PGA using SLI techniques. Here the number of iterations is fixed to 500 for all the units to show the effectiveness of the algorithm.

CONCLUSION

This study presents a two-layer approach to solve the unit commitment problem. The first layer uses a penalty less genetic algorithm and it is proved as an effective algorithm in obtaining the near global solution. The second layer uses an improved lambda-iteration technique to solve the economic dispatch problem to avoid the local convergence. Finally, it can be said that the proposed solution approach represents an interesting and promising method. The simulation results reveal that the features of easy implementation, convergence in an acceptable time and highly optimal solution in solving the unit commitment problem are achieved.

The list of symbols used in this study is as follows:

TPC : Total Production Cost.
$F_i(P_{ih})$: Fuel cost function of the i-th unit with generation output, P_{ih}, at the h-th hour.
a_i, b_i, c_i : Cost coefficients of the i-th unit.
N : The number of available generating units.
H : The number of hours.
P_{ih} : The generation output of the i-th unit at the h-th hour.
ST_i : Start-up cost of the i-th unit.
SD_i : Shut-down cost of the i-th unit.
U_{ih} : On/off status of the i-th unit at the h-th hour and $U_{ih} = 0$ when off, $U_{ih} = 1$ when on.
D_h : Load demand at the h-th hour.
R_h : Spinning reserve at the h-th hour.
$P_{i(min)}$: Minimum generation limit of i-th unit.
$P_{i(max)}$: Maximum generation limit of i-th unit.
up_i : Minimum up-time of i-th unit.
$down_i$: Minimum down-time of i-th unit.
HSC_i : Hot Start Cost of i-th unit.
CSC_i : Cold Start Cost of i-th unit.
CSH_i : Cold Start Hours of i-th unit.

REFERENCES

Arroyo, J.M. and A.J. Conejo, 2002. A parallel repair genetic algorithm to solve the unit commitment problem. IEEE Trans. Power Sys., 17: 1216-1224.

Cheng, C.P., C.W. Liu and G.C. Liu, 2000. Unit commitment by Lagrangian relaxation and genetic algorithms. IEEE. Trans. Power Sys., 15: 707-714.

Cohen, A.I. and M. Yoshimura, 1983. A Branch-and-Bound Algorithm for Unit Commitment. IEEE. Trans. Power Sys., PAS-102: 444-451.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning: Addison Wesley.

Kazarlis, S.A., A.G. Bakirtzis and V. Petridis, 1996. A genetic algorithm solution to the unit commitment problem. IEEE Trans. Power Sys., 11: 83-92.

Lowery, P.G., 1983. Generation Unit Commitment by Dynamic Programming. IEEE Trans. Power Sys., 102: 1218-1225.

Merlin, A and P. Sandrin, 1983. A new Method for Unit Commitment at Electricite De France. IEEE. Trans. Power Sys., 102: 1218-1225.

Muckstadt, J.A. and R.C. Wilson, 1968. An Application of Mixed-Integer Programming Duality to Scheduling Thermal Generating Systems. IEEE. Trans. Power Sys., pp: 1968-1978.

Ouyang, Z. and S.M. Shahidehpour, 1991. An Intelligent Dynamic Programming for Unit Commitment Application. IEEE. Trans. Power Sys., 6: 1203-1209.

Ouyang, Z. and S.M. Shahidehpour, 1993. A multi-stage intelligence system for unit commitment. IEEE. Trans. Power Sys., 7: 639-646.

Petridis, V., S. Kazarlis and A. Bakirtzis, 1998. Varying Fitness Functions in Genetic Algorithm Constrained Optimization: The Cutting Stock and Unit Commitment Problems. IEEE. Trans. Sys. Man and Cybernetics-part B: Cybernetics, 28: 629-640.

Rudolf, A. and R. Bayrleithner, 1999. A genetic algorithm for solving the unit commitment problem of a hydro-thermal power system. IEEE. Trans. Power Sys., 14: 1460-1468.

Sasaki, H., M. Watanabe and R. Yokoyama, 1992. A solution method of unit commitment by artificial neural networks. IEEE. Trans. Power Sys., 7: 974-981.

Sheble, G.B., 1990. Solution of the Unit Commitment Problem by the Method of Unit Periods. IEEE. Trans. Power Sys., 5: 257-260.

Snyder, W.L., H.D. Powell and J.C. Rayburn, 1987. Dynamic Programming Approach to Unit Commitment. IEEE. Trans. Power Sys., 2: 339-350.

Su, C.C. and Y.Y. Hsu, 1991. Fuzzy Dynamic Programming: An Application to Unit Commitment. IEEE. Trans. Power Sys., 6: 1231-1237.

Valenzuela, J. and A.E. Smith, 1999. A Seeded Mimetic Algorithm For Large Unit Commitment Problems. Paper submitted to Journal of Heuristics.

Wang, C. and S.M. Shahidehpour, 1992. A decomposition approach to nonlinear multi-area generation scheduling with tie-line constraints using expert systems. IEEE. Trans. Power Sys., 7: 1409-1418.

Wood, A.J. and B.F. Wollenberg, 1996. Power Generation, Operation and Control: John Wiley and Sons, Inc.

Xing, W.G. and F.F. Wu, 2002. Genetic algorithm based unit commitment with energy contracts. Int. J. Elect. Power, 24: 329-336.

Zhuang, F. and F.D. Galiana, 1988. Toward a more Rigorous and Practical Unit Commitment by Lagrangian Relaxation. IEEE. Trans. Power Sys., 3: 763-772.

Zhuang, F. and F.D. Galiana, 1990. Unit commitment by simulated annealing. IEEE. Trans. Power Sys., 5: 311-318.