GENETIC ALGORITHMS FOR SCHEDULING GENERATION AND MAINTENANCE IN POWER SYSTEMS

C.J. ALDRIDGE, K.P. DAHAL, J.R. MCDONALD

Centre for Electrical Power Engineering
University of Strathclyde, Glasgow, UK

1. Genetic algorithms

1.1 INTRODUCTION

Genetic algorithms (GAs) are search and optimisation methods based on a model of evolutionary adaptation in nature. Unlike traditional 'hill-climbing' methods involving iterative changes to a single solution, GAs work with a population of solutions, which is 'evolved' in a manner analogous to natural selection. Candidate solutions to an optimisation problem are represented by chromosomes, which for example encode the solution parameters as a numeric string. The 'fitness' of each solution is calculated using an evaluation function which measures its worth with respect to the objective and constraints of the optimisation problem.

Successive 'generations' of the population are created by several simple 'genetic' operators, as illustrated in Figure 1. In each generation, solutions are selected stochastically according to their fitness in order to be recombined to form the next generation. Relatively 'fit' solutions survive, 'unfit' solutions tend to be be discarded. A new generation is created by stochastic operators - typically 'crossover', which swaps parts of binary-encoded solution strings, and 'mutation', which changes random bits in the strings. Successive generations yield fitter solutions which approach the optimal solution to the problem.

Genetic algorithms were first developed by John Holland at MIT and described in his 1975 book 'Adaptation in Natural and Artificial Systems' [1]. More recent introductory texts include those by Davis [2], Goldberg [3], Michalewicz [4] and Mitchell [5]. GAs are inherently simple, naturally parallelisable, and can generate a set of near-optimal solutions for evaluation. They provide a powerful technique to resolve complicated multi-dimensional optimisation problems, such as resource allocation and scheduling. A plethora of information and public domain GA programs are available from sites on the World Wide Web, for example [6].

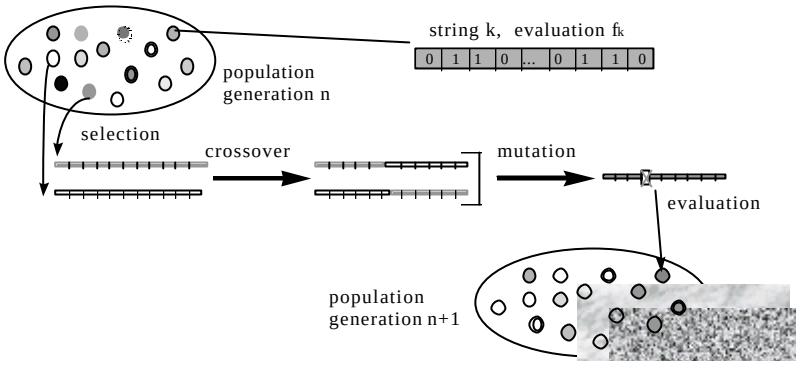

Figure 1: A generation of a basic genetic algorithm.

There are three important issues in the application of a GA to solve an optimisation problem. Firstly, how are the candidate solutions to the optimisation problem represented, in order to allow the genetic adaptation to be easily applied? Secondly, how is the optimality (quality) of the solutions assessed? Finally, how is the 'genetic adaptation' applied to existing solutions to yield new solutions? These issues are addressed in the following sections.

1.2 SOLUTION REPRESENTATION

GAs were initially developed using binary strings to encode the parameters of an optimisation problem. Binary encoding is a standard GA representation that can be employed for many problems: a string of bits can encode integers, real values, sets or whatever is appropriate. Furthermore, the genetic manipulation of binary chromosomes can be done by simple and universal crossover and mutation operators. However, a binary representation is often not appropriate for particular problems, and a problem-specific representation, using strings of integers or floating point numbers, character strings to represent sets, etc. may give a more coherent algorithm. Such representations require appropriately designed genetic operators. Ideally, the solution representation should be such that it represents only the feasible search space, though often this is not possible in practice.

1.3 EVALUATION

An evaluation function is required to assign a figure of merit (fitness) to each new solution, which should reflect the quality of the solution that the chromosome represents. During the GA 'reproduction' process the selection of individuals is done according to their fitness. If the structure of a good solution is well known it is easy to construct a suitable evaluation function. For constrained optimisation problems, the evaluation function typically comprises a weighted sum of the objective (or a simple function of it) and penalty functions to consider the constraints. This approach allows

constraints to be violated, but a penalty depending on the magnitude of the violation is incurred which degrades the fitness. A highly infeasible individual has a high penalty value and will rarely be selected for reproduction, allowing the GA to concentrate on feasible or near-feasible solutions.

Multiple objectives may be included in a weighted sum in the evaluation function. However for more advanced problems, it may be useful to consider Pareto optimality [7] or fuzzy logic [8], which are outside the scope of this introduction.

1.4 SELECTION

In order to mimic the principle of 'survival of the fittest', GAs introduce selection pressure through choosing relatively good solutions for recombination and replacing inferior solutions in the population with new individuals. Selection is a method that stochastically picks individuals from the population according to their fitness: the higher the fitness, the more chance an individual has to be a parent. The selection pressure defines the degree to which better individuals are favoured, which drives the GA to improve the population fitness over successive generations. In general, if the selection pressure is too high, then a superior individual strongly dominates the less fit individuals and this may lead the GA to converge prematurely to a local optimum.

There are three main types of selection methods: fitness-proportionate, ranking and tournament. In fitness-proportionate selection the probability that a solution is selected is directly based on its evaluation value. In order to prevent a highly fit individual dominating the population, the evaluation values are typically scaled linearly. The 'roulette-wheel' method is the simplest and classical fitness-proportionate method. Each individual is assigned a sector of a wheel whose size is proportional to its (scaled) evaluation. A position on the wheel is chosen at random, and the individual to which that position is assigned is selected. Stochastic universal sampling (SUS) selection is similar to the roulette-wheel method, except that a set of individuals are picked simultaneously, based on a random choice of a given number of positions spaced equally around the wheel.

Ranking selection methods take account of the relative ordering of individuals with respect to their evaluation measures. The probability of selecting an individual is then given by a linear function of its rank in the population rather than its evaluation measure. This approach reduces the dominance of highly fit solutions

The basic mechanism of tournament selection involves picking a subset of individuals at random and then selecting one according to their fitness. Selection pressure is applied in choosing from the subset of individuals - for example, the best is selected with a given probability, otherwise the second best is chosen with that probability, and so on.

1.5 RECOMBINATION

Following their selection, 'parent' individuals are recombined to create 'offspring'. This is usually achieved using crossover and mutation operators as below, but other domain-specific operators may also be used during this process.

Crossover exploits the current solutions by exchanging elements of selected parents. This is done with a given probability, typically in the range of 0.6-1.0, otherwise parents are unchanged. One-point crossover is the simplest crossover operator, which breaks the two selected parent strings at a random position and swaps the two substrings to create two offspring which contain information from each of the parent strings. Two-point crossover is commonly used, as illustrated in Figure 2. As an alternative to such 'N-point crossover', the uniform crossover operator copies the value at each position in the off-spring from one or the other parent at random. Off-spring therefore contain a greater mixture of genetic materials from each parent.

parent strings	crossed-over strings	mutated string
101010110101	101110010101	101110010001
001110011110	001010111110	

Figure 2: Two-point crossover and mutation of binary strings. Two crossover positions are chosen randomly (here 3 and 7) and the enclosed bits are exchanged. One of the resulting strings is randomly chosen and each bit is changed with given probability (here mutation is applied to the first string, and the tenth bit is flipped).

A mutation operator is applied to the crossed-over solutions to introduce random changes. This enables further exploration of the search space. Mutation is often seen as a background operator to maintain the genetic diversity in the population. There are many forms of mutation for different types of representation. A simple mutation operator changes the bit/value at each position in the solution string with a given small mutation probability, e.g. 0.01, as shown in Figure 2.

Mutation operators may employ hill climbing mechanisms and only apply mutation to a solution if its evaluation is improved. Such an operator can accelerate the search, but might reduce the diversity in the population and cause the algorithm converge towards some local optima.

1.6 POPULATION UPDATING

There are two basic population updating approaches, known as generational and steady state. The generational approach is as follows. In each generation, the population is replaced by off-spring produced by selection and recombination of parents from the population of the previous generation. The best individual in the population pool is generally retained (elitism). In this case individuals can only recombined with those from

the same generation. In the alternative steady state approach, new offspring are introduced immediately into the population, replacing an existing solution, which is selected for example as the least fit or by tournament selection. Hence parents and off-springs co-exist in the population.

The recombination of two individuals is effective provided there is a sufficient diversity in the population. Ideally, the population size should be as large as possible to enhance the exploration of the search space. However, the computational time and memory required by a GA become costly as the population size increases. A population size of around 100 is typical in practice. The genetic algorithms described in the case studies below use a fixed population size, however in general this may be adapted during the course of a GA run.

An initial population of a given size must be created to begin the GA search process. The simplest way of creating the initial population is to sample the search space at random. However, heuristic methods can be used to generate some or all of the initial population. If some reasonable meaningful solutions are known or can be generated, then their inclusion in the initial population can improve the performance of the GA. An example is given in section 2 below. However the initial population should not lack diversity in order to avoid exploration of a small part of the search space.

The simplest stopping criterion is to run the GA for a fixed number of generations or iterations. Alternatively the algorithm may be continued for as long as the best solution in the population is improving or halted when the solution reaches a required quality.

Instead of a single population, a GA may use a number of smaller populations, known as ‘islands’. Evolution proceeds on each island as for a single population GA, but with a regular exchange of a limited number of individuals between islands. This approach naturally lends itself to implementation on a parallel computer, with different islands allocated to individual processors [9].

1.7 IMPLEMENTATION

GAs are straightforward to implement for practical optimisation problems, typically requiring only the solution representation and evaluation function to be chosen. The evaluation function and genetic operators can be easily modified. GAs also yield multiple solutions which may be subsequently judged. The creation and evaluation of large number of solutions can be computationally costly, though the generational GA is naturally parallelisable.

The performance of a GA may be improved by hybridization with other solution techniques. For example, a heuristic technique may be applied to produce a meaningful initial population, as we describe below. Simulated annealing, an alternative stochastic search technique, may be combined with a GA to improve the search process [10,11].

Solutions in the final population of a GA may also be refined by an appropriate local search method.

A number of GA programs are available in the public domain, such as GENESIS [12], GENITOR [13], and RPL2 [9], which have been employed for the case studies described below. For a given application, the choice of the genetic operators and the values of parameters such as population size, crossover and mutation probabilities must generally be guided empirically.

Genetic algorithms have been applied to a range of search and optimisation problems arising in planning, scheduling and operation of power sytems. A useful bibliography is given in [14], and a recent comprehensive survey of applications of GAs and other evolutionary computing techniques in this area is given in [15]. Problems tackled include unit commitment, economic dispatch, maintenance scheduling, network expansion, alarm processing and parameter estimation. In the remainder of this chapter, we describe two case studies, in which GAs are applied to unit commitment and generator maintenance scheduling.

2. A Knowledge-Based Genetic Algorithm for Unit Commitment

2.1 INTRODUCTION

In order to meet the customer demand in a power system, the generating units must be scheduled to minimise the total cost and satisfy operating constraints. Calculating the optimal commitments (on/off) and dispatched generation for each thermal unit at a sequence of times in the scheduling period is known as the unit commitment & economic dispatch problem. This is a highly constrained combinatorial problem and continues to present a challenge for efficient solution techniques.

The constraints of the problem involve the individual units, groups of units and the entire network. Each unit is generally constrained by minimum and maximum generation, ramp rates which limit the rate of change of the generation, and minimum times that the unit can remain on or off. There may also be specified bounds on the total generation of local groups of units. The predicted demand must be met by the sum of the generation of all the units; in addition the on-line units must together maintain a specified reserve capacity. We seek the solution that satisfies these constraints and minimises the total cost, typically given by the start-up costs and running costs incurred by each unit.

The unit commitment/economic dispatch problem has been tackled using a range of solution methods. Sheble & Fahd [16] review the development of different heuristic, mathematical programming and expert system techniques over the last 30 years. Initially the commitment and dispatch problems were decoupled; indeed, the first to consider the coupled problem was Garver [17] (1963). Prior to this, and even today, priority listing is

employed. In the priority list method, units are ordered according to a measure of cost and committed in this order so that their cumulative generation satisfies the required level; subsequently the dispatch of the committed units is calculated. This simple approach can however give solutions far from the optimum. The chief mathematical programming methods have been Dynamic Programming and Lagrangian Relaxation. The main drawback of Dynamic Programming is that the number of combinations of states which must be searched grows exponentially and becomes computationally prohibitive, hence methods have included for example a priority list to reduce the search space. Recent work has favoured Lagrangian Relaxation [18], in which the global constraints of demand and reserve are admitted into the objective function, and the problem decomposed into master problem and unit subproblems. This natural algorithmic decomposition admits parallelisation [19]. The method provides bounds on the original optimum but a heuristic must be employed to construct a feasible solution for the original problem.

2.2 GENETIC ALGORITHMS

The combinatorial aspect of the commitment problem is a natural target for the application of genetic algorithms, and in the last few years GAs have been used to solve the unit commitment/economic dispatch problem. A review of different GA approaches and results is given in [20]. Most studies have employed a single GA for the entire scheduling period. In general, the commitments are represented in the solution string as a binary array and the dispatch variables are calculated as part of the fitness function evaluation [21-30]. Representations satisfying minimum on and off times have been introduced using integers [31] and binary substrings [32,33], while in [34] both the commitment and dispatch variables were encoded in the solution string. These single GAs have included various problem-specific operators alongside the standard mutation and crossover operators. Alternatively the solution may be calculated sequentially by using a GA for each time interval in turn [35-38].

2.3 TEST PROBLEM FORMULATION

We consider a test problem involving 10 generating units over 24 hourly scheduling points, though our approach may easily be extended for larger problems. A minimum cost schedule is sought subject to the unit and system constraints described below. We use the following notation:

D^t	demand at time t
F_i	no-load cost for unit i
l^t	transmission constraint limit
N	number of units
R	reserve level
T	number of time intervals
U_i	start-up cost

$V_{i,1}, V_{i,2}$	incremental cost gradients
W_i	incremental cost function
x_i^t	generation of unit i at time t
$x_i^{\min}$	minimum generation
$x_i^{\max}$	maximum generation
x_i^*	breakpoint for piecewise linear incremental cost
a_i^t	commitment (binary)
b_i^t	start-up indictator (binary)
g_i^t	shutdown indicator (binary)
r_i	ramp rate
t_i^{on}	minimum on time
t_i^{off}	minimum shutdown time.

Start-up and shutdown indicators are defined by

$$b_i^t = \begin{cases} 1 & \text{if } a_i^{t-1} = 0,\ a_i^t = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

$$g_i^t = \begin{cases} 1 & \text{if } a_i^{t-1} = 1,\ a_i^t = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

A commitment and dispatch schedule is given by the arrays $\{a_i^t\}$ and $\{x_i^t\}$, which we denote by a and x.

The total cost, composed of constant start-up costs and piecewise linear generating costs, is minimised,

$$\min_{a,x} C(a, x), \tag{3}$$

where

$$C(a, x) = \sum_{t=1}^{T} \sum_{i=1}^{N} b_i^t U_i + a_i^t F_i + W_i(x_i^t), \tag{4}$$

$$W_i = \begin{cases} x_i^t V_{i,1}, & \text{if } x_i^t \le x_i^*, \\ x_i^* V_{i,1} + (x_i^t - x_i^*)\, V_{i,2}, & \text{otherwise.} \end{cases} \tag{5}$$

subject to the following constraints:

Generation limits:

$$a_i^t x_i^{\min} \le x_i^t \le a_i^t x_i^{\max} \quad \text{for } i = 1, \ldots, N,\ t = 1, \ldots, T. \tag{6}$$

Ramp rates:

$$\left.\begin{array}{c}\text{if } a_i^{t-1} = a_i^t = 1 \text{ then } -r_i \le x_i^t - x_i^{t-1} \le r_i, \\ \text{if } b_i^t = 1 \text{ then } x_i^t = x_i^{\min}, \\ \text{if } g_i^t = 1 \text{ then } x_i^{t-1} \le x_i^{\min} + r_i, \end{array}\right\} \text{ for } i \in I_{RR},\ t = 1,\ldots,T. \quad (7)$$

Minimum on times:

$$b_i^t + \sum_{t'=t+1}^{\min(t+t_i^{on}-1,T)} g_i^t \le 1 \quad \text{for } i \in I_{MOT},\ t = 1,\ldots,T-1. \quad (8)$$

Minimum shutdown times:

$$g_i^t + \sum_{t'=t+1}^{\min(t+t_i^{off}-1,T)} b_i^t \le 1 \quad \text{for } i \in I_{MST},\ t = 1,\ldots,T-1. \quad (9)$$

Demand:

$$\sum_{i=1}^{N} x_i^t = D^t \quad \text{for } t = 1,\ldots,T. \quad (10)$$

Reserve:

$$\sum_{i=1}^{N} a_i^t x_i^{\max} \ge D^t + R \quad \text{for } t = 1,\ldots,T. \quad (11)$$

Transmission constraint:

$$\sum_{i \in I_{TC}} x_i^t \ge l^t \quad \text{for } t \in T_{TC}. \quad (12)$$

Initial conditions:

$$a_i^0,\ x_i^0 \text{ given for } i = 1,\ldots,N. \quad (13)$$

In the above I_{RR}, I_{MOT}, I_{MST} and I_{TC} denote particular subsets of units associated with the constraints, and T_{TC} is a subset of times. Equations (1)-(13) define a mixed integer programming problem. This may be made linear by formulating (1),(2) and (7) as inequalities and introducing extra variables to reformulate (5). In this form the problem is amenable to Lagrangian relaxation.

The parameter values for the units and demand profile are given in Tables 1 and 2. In addition we take

$$R = 200 \text{ MW},\ l^t = 1600 \text{ MW},\ I_{TC} = \{1,4,7\},\ T_{TC} = \{20,21,22\},$$

and initial conditions

$$x_1^0 = 300 \text{ MW},\ x_2^0 = 700 \text{ MW},\ x_3^0 = 900 \text{ MW},\ a_4^0 = \ldots = a_{10}^0 = 0.$$

In order to gauge the computation required to calculate the optimal solution to this problem, a series of similar problems with fewer constraints and time intervals were solved using branch-and-bound. These problems are given by objective function (3)-(5), initial conditions (13) and generation limits (6), plus (a) demand constraint (10), (b)

demand and reserve constraints (10) and (11), and (c) all constraints (7)-(12), Branch-and-bound was applied to these problems over the first T time intervals, for T=2,4,8,12,24, using standard OSL [39] subroutines on a Sun Sparc workstation. The CPU time to calculate the optimal solution to problem (a) with T=24 was over 10 hours; problem (b) with T=12 required over 6 hours. The solution to problem (b) with T=24 and problem (c) with T=8 was not found within 12 hours of CPU time.

Table 1: Generating unit data, where `-' indicates that the corresponding constraint is not specified for that unit, and units other than 2 and 5 have $x_i^* = x_i^{\max}$.

i	$x_i^{\min}$ (MW)	$x_i^{\max}$ (MW)	r_i (MW/h)	t_i^{on} (h)	t_i^{on} (h)	U_i (£)	F_i (£/h)	$V_{i,1}, V_{i,2}$; x_i^* (£/MWh);
1	300	1000	40	6	2	14,000	5000	10
2	300	1000	180	6	-	14,000	7875	3.75,15; 700
3	400	1000	600	6	-	20,000	9500	5
4	150	500	60	2	2	10,000	3750	15
5	150	500	240	2	-	10,000	5906.25	5.625,22.5;
6	200	500	300	2	-	25,000	7125	7.5
7	200	200	-	-	2	2000	2000	30
8	200	200	-	-	-	1200	1200	31
9	100	200	-	-	-	800	800	35
10	100	200	-	-	-	0	0	40

Table 2: Demand profile.

t	D^t (MW)	0700	3000	1400	3500	2100	3500
0100	2400	0800	4100	1500	3200	2200	2700
0200	2200	0900	4150	1600	3700	2300	2200
0300	2000	1000	4200	1700	4500	2400	1900
0400	1850	1100	4250	1800	5050		
0500	1750	1200	4300	1900	4700		
0600	1700	1300	4000	2000	4200		

2.4 GENETIC ALGORITHM DESIGN

2.4.1 *Solution Representation and Evaluation*

We consider the unit commitment/economic dispatch problem (3)-(13) in the decomposed form:

$$\min_{a} F(a), \tag{14}$$

subject to (8),(9), (11), and the following problem being feasible:

$$F(\alpha) = \min_{x} \{C(\alpha, x):\ x \text{ satisfies } (6), (7), (10) \text{ and } (12)\}. \tag{15}$$

A GA is applied to the combinatorial minimisation unit commitment problem (14). The continuous economic dispatch problem (15) parameterised by α is solved in the evaluation function of the GA, and this may be done by linear programming.

The solution representation in the GA is therefore the binary commitment matrix α. In the implementation of the GA this was stored as a binary string consisting of the commitments ordered by time periods first and units second,

$$(\alpha_1^1, \ldots, \alpha_N^1, \ldots, \alpha_1^T, \ldots, \alpha_N^T). \tag{16}$$

Each string is evaluated by first solving (15) to give *x*, and then summing the total cost given by (4) and penalty functions for violations of constraints (6)-(12). From (6) and (10) a necessary condition on α is

$$\sum_{i=1}^{N} \alpha_i^t x_i^{\min} \geq D^t \quad \text{for } t = 1, \ldots, T. \tag{17}$$

The evaluation function is taken as

$$f(\alpha) = C(\alpha, x) + w_a P_a(\alpha) + w_b P_b(\alpha) + w_c P_c(\alpha) + w_d P_d(\alpha) + w_e P_e(\alpha, x). \tag{18}$$

Here P_a is a penalty function associated with constraints (8) and (9), P_b with (11), P_c with (17), P_d with (10) and (12), P_e with (6) and (7), and the w_a etc. are weights. The penalty functions increase linearly with the constraint violations, and are chosen with the weights so that the penalty terms are typically larger than the cost terms.

2.4.2 *Population Updating*

An initial population of K solutions $\{\alpha^{(1)}, \ldots, \alpha^{(K)}\}$ is created. These are chosen randomly, or else the initial population is 'seeded' using the method described below. The evaluation value $f^{(k)}$ of each solution is then calculated. This may be done by linear programming, but as explained below a heuristic method is used to approximate $f^{(k)}$. A new population is created in the following steps.

1. The lowest evaluation solution $\alpha^{(k^*)}$ is copied to the new population (elitism).
2. A set of 2(*m*-1) parent solutions are selected from the old population by stochastic universal sampling [4]. This is done by ranking the solutions in order of increasing $f^{(K)}$, so $\alpha^{(k^*)}$ has rank 1. Solutions are then selected in proportion to a decreasing linear function of their rank.
3. A pair of parent solutions are combined by one-point crossover with probability p_c to create a new solution. Bit-wise mutation is then applied to the new solution with

mutation rate p_m. The mutated solution is then evaluated and placed in the new population. This crossover, mutation and evaluation is done *K*-1 times to complete the population in the next generation of the GA.

The population updating is repeated for *J* generations, using the heuristic method for solution evaluation. The best solution in the final population is then re-evaluated by calculating $f^{(k^*)}$ exactly by linear programming.

2.4.3 *Selection of Initial Population*

A method to identify the likely structure of the unit commitments was derived following knowledge elicitation with scheduling experts, and the construction and validation of a knowledge model. This was done using the KADS (Knowledge Acquisition and Design Structuring) methodology [40].

Typically a number of units are committed throughout the scheduling period, while others remain uncommitted. These groups of units may be heuristically determined, largely by operating cost; however, the inflexibility of certain units and the transmission constraints must also be taken into account.

The units are initially placed in merit order (here in order of increasing running cost/MW at full output), and their cumulative total generations calculated. Units which lie sufficiently (say $\geq m$ MW) below the minimum demand are classified as 'must-run'; units which lie sufficiently above the maximum demand are classified as 'can't-run'; and those remaining are 'can-run' units. This classification is then revised at each time interval to take account of unit inflexibilities and transmission constraints - in this case, constraint (12) - and the can-run band subsequently narrowed to a margin of around the demand curve. Here we use a margin of width *m*=500 MW.

The resulting 'partition' may then be used to initialise the population of the GA. For each solution in the initial population, the commitments are then set as $a_i^t = 1$ (must-run), $a_i^t = 0$ (can't-run), or chosen randomly (can-run).

2.4.4 *Heuristic Evaluation*

The evaluation of each commitment string requires the solution of the economic dispatch problem (15). In order to realise an efficient algorithm, a fast heuristic method was used to solve (15) rather than a standard linear programming solver. An existing rule-based method for generation scheduling (commitment and dispatch) was identified and described in a knowledge model, again using the KADS methodology. From this model a heuristic method was derived for economic dispatch with commitments given.

In this method the x_i^t are calculated at a sequence of time intervals, ordered according to the maxima and minima of the demand profile. At each time interval the x_i^t are

successively decreased and increased, using merit order, in order to satisfy the group constraints and demand. This is done taking into account the values of x_i^t at previously set times, the given commitments, and the unit capacities and ramp-rates. This approximate method proved to a fast and sufficiently accurate alternative to an exact linear programming method.

To gauge the effectiveness of these knowledge-based methods, the GA was initially applied to a smaller, simple problem with a known optimum solution. Results showed that choosing the initial population based on the derived partition and using the heuristic dispatch method in the evaluation function significantly reduced the computational time of the GA to find the optimum, compared to a GA with random initial population and exact LP evaluation [21]. A schematic of the augmented GA is shown in Figure 3.

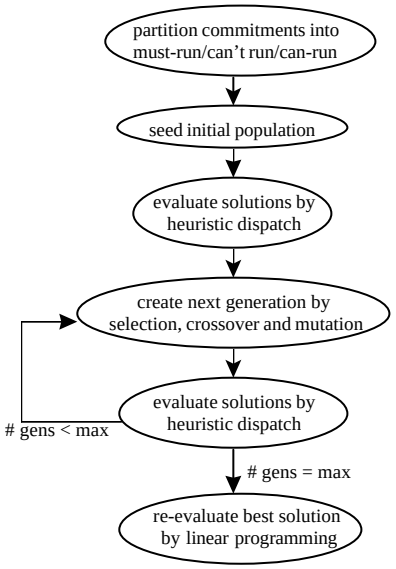

Figure 3: Schematic of knowledge-based GA.

2.5 RESULTS

The GA was implemented using RPL2 [9] and the LP re-evaluation was done using AMPL [41] with OSL solver routines on a Sun Sparc5 workstation. The results presented in this section were obtained with the GA parameters J=1000, K=100, $p_c = 0.9$, $p_m = 0.0015$. These values were found to give the best results in a study of the sensitivity of the GA performance on the above parameters.

The GA was applied to the test problem and the results compared with those obtained using Lagrangian relaxation (LR). Table 3 shows the cost of the solution obtained using LR with 10 sub-gradient iterations, the average cost over ten independent runs of the GA solution, and the associated CPU times. A lower bound, given by the dual cost after 200 iterations, is also shown, which is useful in assessing solutions. The GA cost is 5.7% higher than the lower bound, compared to 6.6% for LR.

Table 3: Comparison of results given by knowledge-based genetic algorithm and Lagrangian relaxation.

LR	solution cost	$1.5603 * 10^6$
	CPU time	145 s
	lower bound	$1.4638 * 10^6$
GA	solution cost	$1.5470 * 10^6$
	CPU time	257 s
	improvement on	0.38%

2.6 CONCLUSIONS

A knowledge-based genetic algorithm has been developed for the unit commitment/ economic dispatch problem. In the GA each binary string is a complete commitment schedule, and the corresponding dispatched generations are calculated in the evaluation of each string. Expert knowledge of generation scheduling has been modelled and used to define the initial conditions of the GA. This has been shown to significantly improve the convergence. Scheduling rules have been incorporated in a fast approximate method of evaluating solutions, accelerating the computational time of the GA to competitive levels. The knowledge-based genetic algorithm has been applied to a representative test problem and shown to obtain better solutions than Lagrangian relaxation (LR) in similar computational times.

3. Generator Maintenance Scheduling using a Genetic Algorithm

3.1 INTRODUCTION

It is vital for a utility to determine when its generators should be taken off-line for preventive maintenance. This is primarily because other short-term and long-term planning activities such as unit commitment, generation dispatch, import/export of power and generation expansion planning are directly affected by such decisions. In modern power systems the demand for electricity has greatly increased with related expansions in system size, which has resulted in higher numbers of generators and lower reserve margins making the generator maintenance scheduling (GMS) problem more complicated. The goal of GMS is to calculate a maintenance timetable for generators in order for example to maintain a high system reliability, reduce total operation cost, and extend generator life time, while satisfying constraints on the individual generators and the power system.

There are generally two categories of objectives in GMS, based on reliability [8,42-48] and economic cost [10,11,45,47-50]. The levelling of the reserve generation over the entire operational planning period is the most common reliability criterion. This can be realised by maximising the minimum net reserve of the system during any time period [45,46,48]. In the case of a large variation of reserve, minimising the sum of squares of the reserves can be an effective approach [46]. Alternatively, the quality of reserve is considered, whereby the risk of exceeding the available capacity is levelled over the entire period by using the equivalent load carrying capacity for each unit and an equivalent load for each interval [46,48]. Minimising the sum of the individual loss of load probabilities for each interval can also be a reliability objective under the conditions of load uncertainty and random forced outages of units [46].

The most common economic objective is to minimise the total operating cost, which includes the costs of energy production and maintenance. If outage durations are allowed to vary, this results in a trade-off solution between the energy production cost and the maintenance cost. Shorter outage durations lead to higher maintenance costs but reduce the load of expensive generation and possible energy purchases, resulting in lower energy production costs [47]. The production cost alone could also be chosen as the objective function by minimising the total energy replacement cost due to preventive maintenance scheduling. However, this is an insensitive objective as it requires many approximations [47,48].

Any maintenance timetable must satisfy a given set of constraints. Typical constraints of the GMS problem are:

- Maintenance window constraints, which define the possible times and the duration of maintenance for each unit.
- Crew constraints, which consider the manpower availability for maintenance work.
- Resource constraints, which specify the limits on the resources needed for maintenance at each period.
- Exclusion constraints, which prevent the simultaneous maintenance of a set of units.
- Sequence constraints, which restrict the initiation of maintenance of some units after a period of maintenance of some other units.
- Load constraints, which consider the demand on the power system during the scheduling period.
- Reliability constraints, which consider the risk level of a given maintenance schedule.
- Transmission capacity constraints, which specify the limit of transmission capacity in an interconnected power system.
- Geographical constraints, which limit the number of generators under maintenance in a region.

In general GMS is a multi-criterion constrained combinatorial optimisation problem, with nonlinear objective and constraint functions. Several deterministic mathematical methods

and simple heuristic techniques are reported in the literature for solving particular GMS problems [45,46,47,48]. Mathematical methods are based on integer programming, branch-and-bound and dynamic programming. However these methods are unsuitable for the nonlinear objectives and constraints of GMS and their computational time grows prohibitively with problem size. The heuristic methods use a trial-and-error method to evaluate the maintenance objective function in the time interval under examination. They require significant operator input and may even fail to find feasible solutions [46,47].

In order to overcome the above limitations a number of artificial intelligence approaches for GMS have been studied [43]. Genetic algorithms (GAs) offer an effective alternative method to solve complex combinatorial optimisation problems, and have recently been applied to GMS using binary strings to represent the maintenance timetable [10,11,42,50] and integer representation [8,42,43]. In all cases, penalty functions were used in the formulation of the evaluation function to take account of violations of problem constraints. GAs have been hybridized with other techniques in order to include scheduling heuristics and improve the performance of the solution algorithm. In [8,49] fuzzy logic was used in the evaluation of each candidate solution, in order to model flexibilities in scheduling using expert knowledge. A knowledge-based technique was employed in [49] for load flow calculation within the evaluation function to improve the speed of the algorithm. GAs have been applied to GMS using the acceptance probability of the simulated annealing (SA) method for the survival of a candidate solution during the evolution process [10,11]. If a newly created solution is an improvement, it is accepted, otherwise it is accepted with a defined probability. The hybridization improved the convergence of the algorithms. In [10] a tabu search (TS) technique was also coupled with the GA/SA hybrid method. In each generation, the best solution was selected as the new trial solution for the TS to improve the search in the neighbourhood of the solution.

In the following sections we describe an application of GAs to a GMS test problem. Both steady state and generational GAs are employed using an integer representation. The GMS test problem and its mathematical model are described in section 4.2. Section 4.3 details the implementation of the genetic algorithm technique to the problem. The test results and the performances of the GAs are discussed in section 4.4, and our conclusions follow in section 4.5.

3.2 TEST GMS PROBLEM

A test problem of scheduling the maintenance of 21 units over a planning period of 52 weeks is considered, which is loosely derived from the example presented in [47] with some simplifications and additional constraints. The problem involves the reliability criterion of minimising the sum of squares of the reserves in each week. Each unit must be maintained (without interruption) for a given duration within a specified window, and the available manpower is limited. Table 4 gives the capacities, allowed periods and duration of maintenance and the manpower required for each unit. The system's peak load is 4739 MW, and there are 20 people available for maintenance work in each week.

Table 4: Data for the test system.

Unit	Capacity (MW)	Allowed period	Outage (weeks)	Manpower required for each week
1	555	1-26	7	10+10+5+5+5+5+3
2	555	27-52	5	10+10+10+5+5
3	180	1-26	2	15+15
4	180	1-26	1	20
5	640	27-52	5	10+10+10+10+10
6	640	1-26	3	15+15+15
7	640	1-26	3	15+15+15
8	555	27-52	6	10+10+10+5+5+5
9	276	1-26	10	3+2+2+2+2+2+2+2+2+3
10	140	1-26	4	10+10+5+5
11	90	1-26	1	20
12	76	27-52	3	10+15+15
13	76	1-26	2	15+15
14	94	1-26	4	10+10+10+10
15	39	1-26	2	15+15
16	188	1-26	2	15+15
17	58	27-52	1	20
18	48	27-52	2	15+15
19	137	27-52	1	15
20	469	27-52	4	10+10+10+10
21	52	1-26	3	10+10+10

The GMS problem can be formulated as an integer programming problem by using binary variables, either indicating the period in which maintenance of each unit starts [8,42-45,50] or representing the maintenance status of each unit at each time [10,11,46-48]. The variables in the first formulation are bounded by the maintenance window constraints and hence the search space is reduced. The test problem is formulated below using these variables. We introduce the following notation:

i	index of generating units
I	set of generating unit indices
N	total number of generating units
t	index of periods
T	set of indices of periods in planning horizon
e_i	earliest period for maintenance of unit i to begin
l_i	latest period for maintenance of unit i to end
d_i	duration of maintenance for unit i
P_{it}	generating capacity of unit i in period t
L_t	anticipated load demand for period t
M_{it}	manpower needed by unit i at period t
AM_t	available manpower at period t

Suppose $T_i \subset T$ is the set of periods when maintenance of unit i may start, so $T_i=\{t \in T: e_i \leq t \leq l_i-d_i+1\}$ for each i. We define

$$X_{it}=\begin{cases}1 & \text{if unit i starts maintenance in period t,}\\ 0 & \text{otherwise,}\end{cases}$$

to be the maintenance start indicator for unit $i \in I$ in period $t \in T_i$. It is convenient to introduce two further sets. Firstly let S_{it} be the set of start time periods k such that if the maintenance of unit i starts at period k that unit will be in maintenance at period t, so $S_{it}=\{k \in T_i: t-d_i+1 \leq k \leq t\}$. Secondly, let I_t be the set of units which are allowed to be in maintenance in period t, so $I_t=\{i: t \in T_i\}$. Then the problem can be formulated as a quadratic 0-1 programming problem as below.

The objective is to minimise the sum of squares of the reserve generation

$$\min_{X_{it}} \left\{ \sum_{t \in T} \left(\sum_{i \in I} P_{it} - \sum_{i \in I_t} \sum_{k \in S_{it}} X_{ik} P_{ik} - L_t \right)^2 \right\}, \qquad (19)$$

subject to the maintenance window constraint

$$\sum_{t \in T_i} X_{it} = 1 \qquad \text{for all } i \in I, \qquad (20)$$

the manpower constraint

$$\sum_{i \in I_t} \sum_{k \in S_{it}} X_{ik} M_{ik} \leq AM_t \qquad \text{for all } t \in T, \qquad (21)$$

the load constraint

$$\sum_{i} P_{it} - \sum_{i \in I_t} \sum_{k \in S_{it}} X_{ik} P_{ik} \geq L_t \qquad \text{for all } t \in T. \qquad (22)$$

3.3 GA IMPLEMENTATION

A solution to the test problem may be represented as a one-dimensional binary string which consists of sub-strings X_{i,e_i}, X_{i,e_i+1}, ..., X_{i,l_i-d_i+1} for each unit i. The size of the GA search space for this type of representation is

$$2^{\sum_{i=1}^{N}(l_i-d_i-e_i+2)}.$$

For each unit i=1,2,...,N, the maintenance window constraint (20) forces exactly one variable in $\{X_{it}: t \in T_i\}$ to be one and the rest to be zero. The solution of the problem

thus amounts to finding the correct choice of positive variable from each variable set $\{X_{it}: t \in T_i\}$, for i=1,2,...,N. The index t_i^* of this positive variable indicates the period when maintenance for unit i starts. In order to reduce the number of variables the t_i^*, i=1,2,...,N, can be taken as new variables. These can be expressed as binary numbers, in a 'binary for integer' representation. However, a direct integer representation automatically considers the maintenance window constraint (20) and greatly reduces the size of the GA search space to

$$\prod_{i=1}^{N} (l_i - d_i - e_i + 2).$$

We present results obtained using the integer representation, which has been found to give significantly better results than the binary representation or binary for integer representation [44].

The merit of the solution represented by the GA string is calculated by an evaluation function, given by a weighted sum of the objective and penalty functions for violations of the constraints, which we seek to minimise. The penalty value for each constraint violation is proportional to the amount by which the constraint is violated, hence

$$\text{evaluation} = \omega_O \text{ SSR} + \omega_M \text{ TMV} + \omega_L \text{ TLV}, \qquad (23)$$

where SSR is the sum of squares of reserves as in (19), TMV is the total manpower violation of (21), and TLV is the total load violation of (22). The weighting coefficients ω_O, ω_M and ω_L are chosen so that the penalty values for the constraint violations dominate over the objective function, and the violation of the relatively hard load constraint (22) gives a greater penalty value than for the relatively soft crew constraint. This is because a solution with a high reliability but requiring more manpower may well be accepted for a power utility as the unavailable manpower may be hired. In fact, there is a trade-off between the level of reliability and the required extra manpower. This flexibility of the problem can be modelled using a fuzzy logic approach within the evaluation function [8].

3.4 TEST RESULTS AND DISCUSSION

Both generational (GN) and steady state (SS) GAs were implemented for the test problem using tournament selection, two-point crossover, random mutation and elitism. A tournament replacement operator was employed for the SS GA. GAs were implemented for the test problem using the RPL2 program [9] on a Sun Sparcstation 1000.

The performance of a GA is generally dependent on the GA parameters used, in particular the crossover and mutation probabilities. The sensitivity of both GAs to the variation of crossover and mutation probabilities CP and MP in the range of 0.6-1.0 and

0.001-0.1 respectively was therefore established. The results are depicted in Figure 4, which shows for each case the average evaluation value of the best solutions obtained from ten independent GA runs. A different initial population was randomly created for each run but the same ten initial populations were used for each case. The total number of iterations (solutions created) in each run was fixed as 30,000. The population size was taken to be 100.

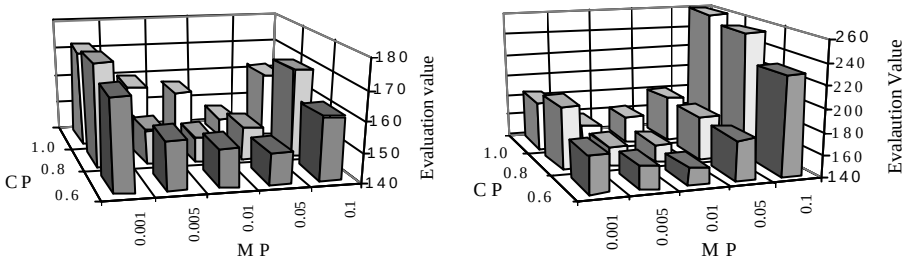

Figure 4: Effect of variations of CP and MP on the performance of SS GA (left) and GN GA (right).

Table 5: Best results obtained from SS GA and GN GA.

	SS GA	GN GA
CP, MP	1.0, 0.05	0.6, 0.01
average evaluation value (over ten runs)	146.71	155.05
best evaluation value (over ten runs)	137.91	148.31
CPU time (one run)	34s	25s

As Figure 4 shows, the average results of both GAs do not vary greatly for varying CP, but are more sensitive to variations of MP, particularly for the GN GA. The SS GA gives the best performance at higher crossover and mutation probabilities than the GN GA. The best results of both GAs are given in Table 5. Hence the SS GA finds better schedules (with lower evaluation values) than the GN GA. However the CPU time (which increases as MP increases) for the GN GA is smaller than that for the SS GA. For both GAs, the best solution (over ten runs) is feasible, so the values shown in Table 5 represent the objective value (SSR multiplied by weighting coefficient ω_O).

The best solution found by the SS GA, whose evaluation measure is 137.91, is illustrated in Figure 5 (left). The schedule represented by the solution is set out in the top portion of the figure, in which the horizontal bars indicate the maintenance of a generating unit. The middle portion of the figure shows the reserve margins in each week for the schedule, which are non-negative since the schedule satisfies the load constraint. The manpower requirements in each week for the solution are depicted in the bottom portion of the figure, which are within the available level.

In order to compare with the best GA solution, we developed a solution heuristically by timetabling the maintenance outages of generators in order of decreasing capacity, to

level the reserve generation while considering the maintenance window and load constraints. The schedule, reserve margins and manpower requirements for each week given by the heuristic solution is illustrated in Figure 5 (right). The solution respects the load constraints but violates the manpower constraints in three time periods. The evaluation value of the solution is 222.61, which is the weighted sum of the objective value (134.61) and the amount of the violation of the constraints. Hence the objective value of the heuristic solution is better than that of the best GA solution, but the solution is infeasible.

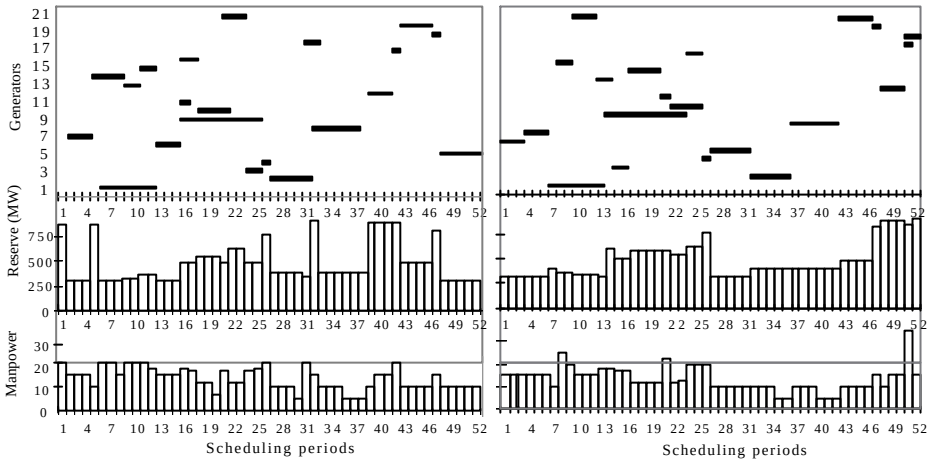

Figure 5: The schedule given by the best GA solution (left) and the heuristically developed schedule (right).

The convergence of the SS GA in finding the best solution (with CP=1.0, MP=0.05) is depicted in Figure 6, which shows the evaluation value of the best solution found so far and the mean evaluation value of the solutions in the population against the number of iterations.

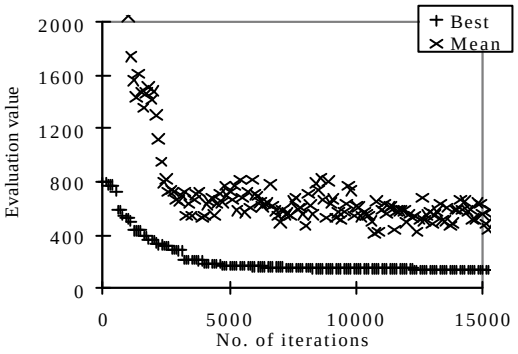

Figure 6: Performance of the SS GA in finding the best solution.

The reduction of the mean evaluation value and the evaluation value of the best solution is very quick in the initial stage, up to 5000 iterations, of the GA. During this stage, the GA mainly concentrates on finding feasible solutions to the problem. The population does not converge to the best solution even after a large number of iterations as the high mutation probability (0.05) maintains the diversity in population. The convergence of the algorithm can be improved even with a high mutation probability if the probabilistic acceptance criteria of a SA technique is incorporated into the GA method. A further improvement can be gained by initialising the population using a heuristic schedule. This results will be reported elsewhere.

3.5 CONCLUSIONS

A GA technique using an integer representation has been demonstrated for a test problem of generator maintenance scheduling. The use of an integer rather than binary representation greatly reduces the GA search space and is straightforward to implement. A penalty function approach has been employed to consider the constraints of the problem. Two GAs with steady state and generational design were tested and the effect of varying crossover and mutation probabilities were studied. The test results show that both the GAs are stable to variation in crossover probability in the expected range. The GN GA is found more sensitive to variation in mutation probability than the SS GA. The integer SS GA gives better performance than the integer GN GA in term of finding better solution in a fixed number of iterations, but the latter is found to be faster.

The results presented above show that the GA is a robust and stable technique for the solution of GMS problems. Good solutions of the problem can be found if an appropriate problem encoding, GA approach, evaluation function and GA parameters are selected for the problem.

Acknowledgements
This work was carried out in the Rolls-Royce University Technology Centre in Power Engineering at the University of Strathclyde. The first author was supported by the Engineering and Physical Science Research Council and The National Grid Company, and the second author was supported by Rolls-Royce plc. The authors also acknowledge the use of the Reproductive Plan Language, RPL2, produced by Quadstone Limited, in the production of this work.

4. References

1. Holland, J.H (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press.
2. Davis, L. (1991) *Handbook of Genetic Algorithms*, Van Nostrand Reinhold.
3. Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimisation, and Machine Learning*, Addison-Wesley.

4. Michalewicz, Z. (1994) *Genetic Algorithms + Data Structures =Evolution Programs*, Springer-Verlag.
5. Mitchell, M. (1996) *An Introduction to Genetic Algorithms.* MIT Press.
6. *GA Archive*, web site http://www.aic.nrl.navy.mil/galist/
7. Fonseca, C.M. and Fleming, P.J. (1993) Genetic algorithms for multiobjective optimization: formulation, discussion and generalization, *Proceeding of the 5th International Conference on Genetic Algorithms (ICGA'93)*, Morgan Kaufmann Publishers, 416-423.
8. Dahal, K.P., Aldridge, C.J. and McDonald, J.R. (in press) Generator maintenance scheduling using a genetic algorithm with a fuzzy evaluation function, *Fuzzy Sets and Systems.*
9. Quadstone Limited (1997) Reproductive Plan Language (RPL2), User manual.
10. Kim, H., Hayashi, Y. and Nara, K. (1997) An algorithm for thermal unit maintenance scheduling through combined use of GA, SA and TS, *IEEE Transactions on Power Systems* **12,** 329-335.
11. Kim, H., Nara, K. and Gen, M. (1994) A method for maintenance scheduling using GA combined with SA, *Computers and Industrial Engineering* **27,** 477-480.
12. Grefenstette, J. (1990) *A user's guide to GENESIS*. ftp.aic.nrl.navy.mil/pub/galist/src/ga.
13. Whitley, D.L. (1990) *GENITOR.* Ftp.cs.colostate.edu/pub/GENITOR.tar.
14. Alander, J.T. (1996) *An indexed bibliography of genetic algorithms in power engineering*. Report 94-1-POWER, University of Vaasa, ftp.uwasa.fi,/cs/report94-1/gaPOWERbib.ps.Z.
15. Miranda, V., Srinivasan, D. and Proença, L.M. (1998) Evolutionary computation in power systems, *Electrical Power & Energy Systems* **19**, 45-55.
16. Sheble, G.B. and Fahd, G.N. (1994) Unit commitment literature synopsis, *IEEE Transactions on Power Systems* **9**, 128-135.
17. Garver, L. (1963) Power generation scheduling by integer programming --- development of theory, *AIEE Transactions* **81,** 1212-1218.
18. Oliveira, P., McKee, S., and Coles, C. (1992) Lagrangian relaxation and its application to the unit-commitment-economic-dispatch problem, *IMA Journal of Mathematics Applied in Business and Industry* **4,** 261-272.
19. Oliveira, P., Blair-Fish, J., McKee, S., and Coles, C. (1992) Parallel Lagrangian relaxation in power scheduling, *Computer Systems in Engineering* **3**, 609-612.
20. Aldridge, C.J., McKee, S. and McDonald, J.R. (1997) Genetic algorithm methodologies for scheduling electricity distribution, in M. Brøns, M.P. Bendsøe and M.P. Sørensen (eds.), *Progress in Industrial Mathematics at ECMI'96*, Teubner, 364-371.
21. Aldridge, C.J., McDonald, J.R. and McKee, S. (1997) Unit commitment for power systems using a heuristically augmented genetic algorithm, *Proceedings of 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'97),* IEE Conference Publication 446, 433-438.
22. Cai, X.-Q. and Lo, K.-M. (1997) Unit commitment by a genetic algorithm, *Nonlinear Analysis, Theory, Methods & Applications* **30**, 4289-4299.
23. Hassoun, M.H. and Watta, P. (1994) Optimization of the unit commitment problem by a coupled gradient network and by a genetic algorithm, *report no. TR-103697*, Electric Power Research Institute.
24. Kazarlis, S.A., Bakirtzis, A.G. and Petridis, V. (1996) A genetic algorithm solution to the unit commitment problem, *IEEE Transactions on Power Systems* **11,** 83-90.
25. Ma, X., El-Keib, A.A., Smith, R.E. and Ma, H. (1995) A genetic algorithm based approach to thermal unit commitment of electrical power systems, *Electrical Power Systems Research* **34**, 29-36.
26. Maifeld, T.T. and Sheble, G.B. (1996) Genetic-based unit commitment algorithm, *IEEE Transactions on Power Systems* **11**, 1359-1370.
27. Numnonda, T., Annakkage, U.D. and Pahalawaththa, N.C. (1996) Unit commitment using stochastic optimisation, *Proceedings of Intelligent Systems Applications in Power Systems (ISAP'96),* 429-433.

28. Orero, S.O. and Irving, M.R. (1997) A combination of the genetic algorithm and Lagrangian relaxation decomposition techniques for the generation unit commitment problem, *Electrical Power Systems Research* **43**, 149-156.
29. Sheble, G.B. and Maifeld, T.T. (1994) Unit commitment by geneticalgorithm and expert system, *Electrical Power Systems Research* **30**, 115-121.
30. Shebleé, G.B., Maifeld, T.T., Brittig, K., Fahd, G. and Fukurozaki-Coppinger, S. (1996) Unit commitment by genetic algorithm with penalty methods and a comparison of Lagrangian search and genetic algorithm - economic dispatch example, *Electrical Power & Energy Systems* **18**, 339-346.
31. Saitoh, H., Inoue, K. and Toyoda, J. (1994) Genetic algorithm approach to unit commitment, in A. Hertz, A.T. Holen and J.C. Rault (eds.) *Proceedings of the International Conference on Intelligent System Application to Power Systems*, 583-589.
32. Yang, P-C., Yang, H-T. and Huang, C-L. (1996) Solving the unit commitment problem with a genetic algorithm through a constraint satisfaction technique, *Electrical Power Systems Research* **37**, 55-65.
33. Yang, H.-T., Yang, P.-C. and Huang, C.-L. (1997) A parallel genetic algorithm approach to solving the unit commitment problem: implementation on the transputer networks, *IEEE Transactions on Power Systems* **12**, 661-668.
34. Oliveira, P., McKee, S., and Coles, C. (1994) Genetic algorithms and optimising large nonlinear systems, in J.H. Johnson, S. McKee and A. Vella (eds.) *Artificial Intelligence in Mathematics*, OUP, 305-312.
35. Dasguptar, D. and McGregor, D.R. (1994) Thermal unit commitment using genetic algorithms, *IEE Proceedings C - Generation, Transmission and Distribution* **141,** 459-465.
36. Orero, S.O. and Irving, M.R. (1995) Scheduling of generators with a hybrid genetic algorithm, *Proceedings of 1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, IEE Conference Publication no. 414, 200-206.
37. Orero, S.O. and Irving, M.R. (1996) A genetic algorithm for generator scheduling in power system, *International Journal of Electrical Power and Energy Systems* **18**, 19-26.
38. Orero, S.O. and Irving, M.R. (1997) Large scale unit commitment using a hybrid genetic algorithm, *International Journal of Electrical Power and Energy Systems* **19**, 45-55.
39. IBM (1992) *Optimisation Subroutine Library (OSL) Guide and Reference*, Release 2.
40. Wielinga, B.J., Schreiber, A.Th. and Breuker J.A. (1992) KADS: A modelling approach to knowledge engineering, *Knowledge Acquisition* **4**, 5-53.
41. Fourer, R., Gay, D.M. and Kernighan, B.W. (1993) *AMPL - A Modeling Language for Mathematical Programming*, Boyd & Fraser.
42. Dahal, K.P., and McDonald, J.R. (1998) Generational and steady state genetic algorithms for generator maintenance scheduling problems, in G.D. Smith, N.C. Steele and R. Albrecht (eds.), *Artificial Neural Nets and Genetic Algorithms, Proceedings of Third International Conference in Norwich (ICANNGA'97)*, Springer-Verlag, Vienna, 260-264.
43. Dahal, K.P., and McDonald, J.R. (1997) A review of generator maintenance scheduling using artificial intelligence techniques, *Proceedings of 32nd Universities Power Engineering Conference (UPEC'97)*, 787-790.
44. Dahal, K.P. and McDonald, J.R. (1997) Generator maintenance scheduling of electric power systems using genetic algorithms with integer representation, *Proceedings of 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'97)*, IEE Conference Publication 446, 456-461.
45. Dopazo, J.F. and Merrill, H.M. (1975) Optimal generator maintenance scheduling using integer programming, *IEEE Transactions on Power Apparatus and Systems* **94,** 1537-1545.
46. Egan, G.T., Dillon, T.S. and Morsztyn, K. (1976) An experimental method of determination of optimal maintenance schedules in power systems using branch-and-bound technique, *IEEE Transactions on Systems, Man and Cybernetics* **6,** 538-547.

47. Yamayee, Z. and Sidenblad, K. (1983) A computationally efficient optimal maintenance scheduling method, *IEEE Transactions on Power Apparatus and Systems* **102**, 330-338.
48. Yang, S. (1994) Maintenance scheduling of generating units in a power system, in X. Wang and J.R. McDonald (eds.), *Modern Power System Planning*, McGraw-Hill, London, 247-307.
49. Bretthauer, G., Gamaleja, T., Handschin, E., Neumann U. and Hoffmann, W. (1998) Integrated maintenance scheduling system for electrical energy systems, *IEEE Transactions on Power Delivery* **13**, 655-660.
50. Burke, K.B., Clarke, J.A. and Smith, A.J. (1998) Four methods for maintenance scheduling, in G.D. Smith, N.C. Steele and R. Albrecht (eds.), *Artificial Neural Nets and Genetic Algorithms, Proceedings of Third International Conference in Norwich (ICANNGA'97)*, Springer-Verlag, Vienna, 265-270.