Generational and Steady State Genetic Algorithms for Generator Maintenance Scheduling Problems

K.P. Dahal, J.R. McDonald
Centre for Electrical Power Engineering, University of Strathclyde, UK
(presented at International Conference on Artificial Neural Networks and Genetic Algorithms, ICANNGA'97)

Abstract

The aim of generator maintenance scheduling (GMS) in an electric power system is to allocate a proper maintenance timetable for generators while maintaining a high system reliability, reducing total production cost, extending generator life time etc. In order to solve this complex problem a genetic algorithm technique is proposed here. The paper discusses the implementation of GAs to GMS problems with two approaches: generational and steady state. The results of applying these GAs to a test GMS problem based on a practical power system scenario are presented and analysed. The effect of different GA parameters is also studied.

1 Problem description

Generator maintenance scheduling (GMS) is an essential part of the problem of economic operation and control of power systems, and involves finding the optimum timing of the outages of generating units. This is important primarily because other planning activities are directly affected by such decisions. In modern power systems the demand for electricity has increased with related expansions in system size, which has resulted in higher numbers of generators and lower reserve margins making the GMS problem more complicated. A good maintenance schedule increases system operating reliability, reduces generation cost, extends equipment life time and relaxes new installation pressure.

The GMS problem is a complex combinatorial constrained optimisation problem. There are generally two categories of objective functions in GMS based on reliability or economic cost criteria. The problems have the following general constraints to be satisfied.

- Maintenance window - defines the limitation on the earliest and latest times and the duration of maintenance for a unit.
- Sequence constraints - the maintenance of certain units is allowed only after the maintenance of other specified units.
- Non-simultaneous constraints - the simultaneous maintenance of certain units is not allowed.
- Crew and resource constraints - consider the availability of manpower and resources.
- Load constraints - consider the demand and the reliability of power supply.

Mathematically, GMS problems can be formulated as integer programming problems using binary variables associated with answers to "When does maintenance start?". The use of these variables instead of the variables associated with answers to "When does maintenance occur?" reduces the number of variables [1]. The first formulation satisfy the constraints on the periods and duration of maintenance. The answer to the first question automatically provides the answer to the second.

Several deterministic mathematical methods and heuristic techniques are reported in the literature for solving these problems [1,2,3]. General solution methods are based on integer programming, branch-and-bound techniques, dynamic programming, etc. However, such approaches are severely limited by the 'curse of dimensionality' and are poor in handling the non-linear objective and constraint functions that characterise the GMS problem. The heuristic approach uses a trial-and-error method to evaluate the maintenance objective function in the time interval under examination. This requires significant operator input and in some situations it fails to produce even feasible solutions [3].

Genetic algorithms (GAs) provide a new approach to the solution of complex combinatorial optimisation problems. This paper describes the procedure for implementing GAs for solving the GMS problem. Two GA approaches, namely generational and steady state, have been applied to test GMS problems which include features of real systems. The paper discusses the application of GAs to GMS problems using a reliability criteria based on levelling reserve generation [3]. This is achieved by minimising the sum of squares of the reserves over the entire operational planning period.

2 GA approach

Two basic approaches, known as the generational approach and the steady state approach, may be implemented in the realisation of a genetic algorithm (GA). In each iteration step, called a 'generation', a generational genetic algorithm (GN GA)

replaces the population of the previous generation by off-spring which are reproduced by applying genetic manipulation to parents selected from the population of the previous generation according to some selection procedure. The iteration is continued until a termination criteria has been reached. A widely available GA package GENESIS [4] has been used to carry out the numerical tests for GMS problems using this GN GA approach.

A steady state genetic algorithm (SS GA) selects two individuals from the population pool in each iteration step according to some selection procedure. A new off-spring is created by applying genetic manipulation to the selected individuals, and is inserted into the population pool replacing a less fit individual. Hence, the parents and off-spring can co-exist in the population pool for the next iteration step. A SS GA software package GENITOR [5] which uses this steady state structure has been applied to GMS problems

The ranking selection method, where parents are selected according to their ranked fitness score has been used for both algorithms.

In order to tackle GMS problems using a GA, a candidate solution of a GMS problem is encoded as a one dimensional binary array as follows.

$$[X_{1,e1}, X_{1,(e1+1)}, \cdots, X_{1,(l1-d1+1)}, X_{2,e2}, X_{2,(e2+1)}, \cdots \cdots, X_{2,(l2-d2+1)}, \cdots, X_{N,eN}, X_{N,(eN+1)}, \cdots X_{N,(lN-dN+1)}]$$

where

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

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,
N = total number of generating units.

This binary string (chromosome) consists of sub-strings which each contain the variables over the whole scheduling period for a particular unit. The size of the GA search space for this type of representation is

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

To take into account the various constraints of GMS problems, we have taken a penalty function approach. The penalty value for each constraint violation increases linearly with the amount by which the constraint is violated. The evaluation function is a weighted sum of penalty values for each constraint violation and the objective function itself, hence

$$\text{evaluation} = \sum_c \omega_c V_c + \omega_o F,$$

where ω_c and ω_o are the weighting coefficients, V_c is the violation of constraint c and F is the objective value. The coefficients are chosen in such a way that the violation of harder constraints gives a greater penalty value than for the soft constraints. In general the penalty value for the constraint violations dominates over the objective function. Feasible solutions with low objective values have high fitness values while unfeasible solutions with high objective values take low fitness measures.

In the test problem described below the crew constraint was assigned a low penalty coefficient. 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.

3 Test Results

Table 1: 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

A number of small problems have been tested with the proposed GAs with different objectives and constraints. GAs with both generational and steady state approaches yield the optimum solution for small

problems when appropriate GA parameters are chosen. Here we present the results of applying a steady state GA to a larger test problem comprising 21 units over a planning period of 52 weeks, which was loosely derived from the example presented in [2] with some simplifications and additional constraints. The data for the test problem is given in Table 1.

The objective is to schedule the maintenance outages of generators to minimise the sum of the squares of the reserve generation. Each unit must be maintained exactly once and the maintenance for each unit must occupy the required time duration without interruption. The system's peak load is 4739 MW. There are only 20 people available for the maintenance work each week. Due to its complexity the optimum solution for this problem is unknown.

Table 2 presents the results of a number of runs of the GN GA and SS GA taking different values of the GA parameters. The total number of trials for each run was fixed at 100000. The crossover operator used for both GAs is a simple two-point crossover. In GENITOR crossover is applied in each iteration of the SS GA when the exchanged information is unique to each parent. In the GN GA the crossover probability was similarly set to be 1.

The first part of Table 2 demonstrates the test results obtained with varying values of the mutation probability (MP), while taking other GA parameters as constant. The selection bias (SB) and population size (PS) are taken as 2.0 and 50 respectively. Each case presents the outcome of 5 GA runs, using a different random seed.

Table 2: Effect of GA parameters for 100000 trials.

		GN GA			SS GA		
	Value	min	avg	max	min	avg	max
MP SB=2 PS=50	.001	*278*	*488*	*885*	2987	4441	8051
	.005	342	616	971	*679*	*1495*	*2105*
	.01	7870	1.2e5	3.0e6	1285	1851	2701
	.05	7.1e6	7.6e6	8.1e6	3.5e6	4.6e6	5.2e6
SB PS=50	1.01	2.0e5	4.5e5	7.0e5	*818*	*878*	*950*
	1.25	*217*	*305*	*378*	914	1627	2758
	1.5	227	585	807	703	1056	1925
	2.0	278	488	885	679	1495	2105
PS	25	*229*	*268*	*339*	433	1162	2388
	50	278	488	885	818	878	950
	100	355	642	1132	363	1107	2954
	200	253	349	543	412	1041	2011
	500	6.0e5	6.6e6	7.0e5	*166*	*408*	*918*
With seeding		*163*	*172*	*184*	*163*	*183*	*203*
Best solution		163.62			163.62		
CPU time		1m1.59s			1m53.28s		

The results show that the GN GA is more sensitive than the SS GA to variations in MP. For both GAs, lower values of MP are recommended to achieve a better solution.

For each unit i=1,2,...,N, the maintenance window constraint (2) forces exactly one variable in $\{X_{it}: e_i \le t \le l_i - d_i + 1\}$ to be one and the rest to be zero. Therefore, a maintenance window feasible genetic structure contains many more '0' bits than '1' bits. For our test problem, only 21 out of 496 bits in the string must be '1' and the rest '0'. Hence the most of the search space represents unfeasible solutions. A high mutation probability increases the chance of changing these '0's into '1's and has the potential to disrupt and degrade the search process. With higher mutation probabilities the GA could not find a maintenance window feasible solution even in 100000 trials. However, with lower mutation probabilities the GA found maintenance feasible solutions.

The selection bias value specifies the amount of preference to be given to the superior individuals in selection of parents. If SB=2, for example, then the selection probability for the best individual is twice that of the mean individual. When SB is close to 1, the distribution of selection probability becomes nearly uniform. In general, if the selection bias is too high, then a superior solution strongly dominates the less fit solutions and this may lead the GA to converge prematurely to a local minimum. Low values of the selection bias cause less preference to be given to the good genetic structures previously found. The second part of Table 2 presents results found using 4 different bias values for each of the two GAs, with MP chosen to give the best performance from above. Taking selection bias 1.01 for the SS GA and 1.25 for the GN GA gives the best solution. Thus the SS GA gives good results when all individuals in the pool have virtually uniform probability of selection. For the GN GA selection pressure towards fitter individuals leads to better performance.

Table 2 also presents the outcomes of 5 GA runs for different population sizes with other GA parameters chosen to give the best performance from above. For a fixed number of trials, the number of generations in the GN GA decreases as the population size increases. Hence the GN GA performance is poor for the large population size. The SS GA performs better with larger population size.

In order to enhance the performance of the GAs, one of the individuals in the initial population was

created meaningfully and the remainder chosen randomly as before. The 'seeded' solution was developed heuristically by ranking the generating units in order of decreasing capacity to level the reserve generation while considering the maintenance window constraints. The results of 5 runs of both GAs are shown in Table 2. The seeding significantly improves the performance of both GAs. During the early iterations, the GAs with random initial population spend most of their time on finding maintenance window feasible solutions. Therefore, the inclusion of a maintenance window feasible solution incorporating domain knowledge in the initial population leads to the improvement of the GA performance. In the real system problem, some previously used solutions may be available for initialisation.

In the GN GA the reproduction of individuals within a generation is independent of the offspring produced in that generation as parents and children do not co-exist in a genetic pool. The influence of new offspring in the reproduction procedure can only occur in the next generation. For each generation a number of individuals equal to the population size are selected for genetic manipulation which helps to preserve the diversity of population in the genetic pool. This reduces the chance of premature convergence.

In the steady state GA, there is no concept of 'generation' and the produced offspring enter to the genetic pool before the next trial. Hence, the influence of the offspring in the reproduction procedure is immediate. As a fitter solution generally replaces a less fit solution in the pool at every trial there is a chance of filling the genetic pool with individuals converging towards the top individual of the pool during the course of GA run. The crossover operator acts to improve the solutions in the initial trials. However, in the later trials the improvements to the offspring are expected to be due to the mutation operator only, as the crossover operator does not change any information between identical parents. Therefore, the improvement of candidate solutions in the pool is faster for initial trials.

The CPU times on a DEC Utirx 5000/260 workstation for both GAs are shown in Table 2. With GA parameters chosen to give the best performance for each GA, the GN GA takes about a half of the computation time than that of the SS GA to obtain the same result.

4 Conclusions

Two GAs with generational and steady state design were tested for a GMS problem. The effects of varying the mutation probability, selection bias and population size were studied. The test results show that both GAs are sensitive to variation in these parameters and appropriate values must be chosen in order to obtain good solutions. In both cases a low value of mutation probability must be chosen. The GN GA gives better results with a small population. For the SS GA large population size and virtually uniform selection gives the best results. The effect of seeding a heuristically derived individual in the initial pool was investigated for both GAs. Seeding greatly enhances the performance of both GAs in finding better solutions. The obtained results show that the GN GA gives better performance than the SS GA in terms of the speed and the average quality of the solution.

Binary values were used for representing the maintenance start period during the encoding of the GMS problem. However, the problem variables are numeric so representing them directly as a integers rather than bit strings can reduce the size of the GA search space greatly. The use of problem specific knowledge in the formulation of the evaluation function and in the design of the GA operator could reduce the computational time of the GA. Furthermore, domain knowledge may be used to prevent obviously unfit chromosomes, or those which would violate problem constraints from being created within the GA. Further research is in progress to investigate all these and other issues and the results will be reported else where.

5 References

[1] J.F. Dopazo, H. M. Merrill, "Optimal generator maintenance scheduling using integer programming", IEEE Transactions on Power Apparatus and Systems, Vol.PAS-94, No. 5, September/October 1975, pp1537-1545.

[2] Z. Yamayee, S. Kathleen, "A computationally efficient optimal maintenance scheduling method", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, No. 2, February 1983, pp 330-338.

[3] X. Wang, J.R. McDonald, "Modern Power System Planning", McGraw-Hill, London, 1994, pp247-307.

[4] J. J. Grefenstette, "A User's Guide to GENESIS Version 5.0", available at ftp site: ftp.aic.nrl.navy.mil:/pub/galist/src/ga/genesis.tar.z. 1990.

[5] Darrel L. Whitley, "GENITOR", available at ftp site: ftp.cs.colostate.edu/pub/GENITOR.tar, Colorado State University, 1990.