

# Four Methods for Maintenance Scheduling

Edmund K. Burke, University of Nottingham, *ekb@cs.nott.ac.uk*  
John A. Clark, University of York, *jac@minster.york.ac.uk*  
Alistair J. Smith, University of Nottingham, *ajs@cs.nott.ac.uk*

## Abstract

We had a problem to be solved: the thermal generator maintenance scheduling problem [Yam82]. We wanted to look at stochastic methods and this paper will present three methods and discuss the pros and cons of each. We will also present evidence that strongly suggests that for this problem, tabu search was the most effective and efficient technique.

The problem is concerned with scheduling essential maintenance over a fixed length repeated planning horizon for a number of thermal generator units while minimising the maintenance costs and providing enough capacity to meet the anticipated demand.

Traditional optimisation based techniques such as integer programming [DM75], dynamic programming [ZQ75, YSY83] and branch-and-bound [EDM76] have been proposed to solve this problem. For small problems these methods give an exact optimal solution. However, as the size of the problem increases, the size of the solution space increases exponentially and hence also the running time of these algorithms.

To overcome this difficulty, modern techniques such as simulated annealing [Č85, KGV83], stochastic evolution [SR91], genetic algorithms [Gol89] and tabu search [Ree93] have been proposed as an alternative where the problem size precludes traditional techniques.

The method explored in this paper is tabu search and a comparison is made with simulated annealing (the application of simulated annealing to this problem is given in [SN91]), genetic algorithms and a hybrid algorithm composed with elements of tabu search and simulated annealing.

## 1 Problem Description

Consider  $I$  generating units producing output over a planning horizon of  $J$  periods. Each unit  $1 \leq i \leq I$  must be maintained for  $M_i$  contiguous periods during the horizon. However the starting period denoted by  $x_i$  for each unit  $i$  is unconstrained even in the case that  $x_i = J$  and  $M_i > 1$  for some  $i$ . Since we are considering a rolling plan, the maintenance period would wrap around to the start of our planning horizon.

The operating capacity of each unit is denoted by  $C_i$ . Under no circumstances is it possible for a unit to exceed this limit.

In order to avoid random factors in the problem, such as unit random outages, a reserve capacity variable proportional to the demand is incorporated into the problem description. This problem is classified as a deterministic cost-minimisation problem and can be solved using an optimisation-based technique.

Therefore in period  $j$  where  $1 \leq j \leq J$ , the anticipated demand for the system as a whole will be denoted by  $D_j$  and the reserve capacity required by  $R_j$ . Fuel costs can also be estimated for each period as a constant,  $f_j$  per unit output.

Finally, let  $p_{ij}$  represent the generator output of unit- $i$  at period- $j$ ,  $c_i(j)$  be the maintenance cost of unit- $i$  if committed at period- $j$  and let  $y_{ij}$  be a state variable, equal to one if unit- $i$  is being maintained in period- $j$  and otherwise zero.

The objective of the problem is to minimise the sum of the overall fuel cost and the overall cost of maintenance:

$$\text{Minimise } \sum_{j=1}^J (f_j \cdot \sum_{i=1}^I p_{ij}) + \sum_{i=1}^I c_i(x_i)$$

Once the maintenance of unit- $i$  starts, the unit must be in the maintenance state for  $M_i$  contiguous periods.

$$y_{ij} = \begin{cases} 0 & \text{if } j = 1, 2, \dots, x_i - 1 \\ 1 & \text{if } j = x_i, \dots, x_i + M_i - 1 \\ 0 & \text{if } j = x_i + M_i, \dots, J \end{cases}$$

If  $x_i + M_i > J$  then the maintenance wraps around to the next repetition of the planning horizon. This formulation captures the notion of continual maintenance. It would be possible to arrange matters so that overlap never happens; the difference is minor.

The generator output must not exceed the upper limit; the output of the generator is set to zero during maintenance.

$$0 \leq p_{ij} \leq C_i(1 - y_{ij})$$

The total output must equal the demand in each period,

$$\sum_{i=1}^I p_{ij} = D_j \text{ where } j = 1, 2, \dots, J$$

and the total capacity must not be less than the required reserve.

$$\sum_{i=1}^I (1 - y_{ij}) C_i \geq (D_j + R_j)$$

Our formulation of the problem is based closely on that of [SN91].

To simplify the operation of the algorithm, all solutions in the solution space are considered valid. A solution could be infeasible if the demand and reserve constraints cannot be met. In this case the solution is penalised by the addition of a penalty function:

$$\alpha \sum_{j=1}^J u_j + \beta \sum_{j=1}^J v_j$$

Where  $\alpha$  and  $\beta$  are tunable parameters and  $u_j$  and  $v_j$  are derived from the shortfall in output:

$$\left( \sum_{i=1}^I p_{ij} \right) + u_j = D_j$$

and the shortfall in capacity.

$$\left( \sum_{i=1}^I C_i (1 - y_{ij}) \right) + v_j = D_j + R_j$$

$u_j$  and  $v_j$  are not permitted to be negative, thus a feasible solution incurs no penalty function.

Thus any initial solution can be chosen and the optimisation algorithm will be directed towards feasible solutions through the choice of sufficiently high  $\alpha$  and  $\beta$ .

## 2 Problem Solution and Discussion

Simulated annealing [KGV83], genetic algorithms [Hol75], tabu search [Ree93] and a hybrid algorithm composed from elements of simulated annealing and tabu search were implemented to solve this problem.

The problem as described in the previous sections can be reduced to that of finding the optimal values of  $x_i$ .

### 2.1 Implementation of the methods

#### 2.1.1 Simulated Annealing

Each iteration consists of generating  $n$  states in the neighbourhood of the current state. A solution is accepted with probability  $\exp^{-\Delta/T_k}$  where  $T_k$  is the current “temperature” and if accepted, it replaces the current state from which the new states are derived during the current iteration and so the current solution can

change several times during one iteration.  $T_k$  is initialised to some value  $T_0$  and varies according to a cooling schedule  $T_{k+1} = pT_k$ ,  $0 \leq p \leq 1$ .

Once  $n$  states have been generated, if the number of solutions which were *accepted* during the last iteration is less than some value then the algorithm terminates.

#### 2.1.2 Genetic Algorithms

The encoding used is a binary representation of each unit’s starting period concatenated together. This gives a suitable representation for use with a genetic algorithm.

After generating a starting population consisting of individuals generated at random, for a predetermined number of iterations the genetic algorithm performs roulette-wheel selection to choose a pair of individuals. Single-point crossover is then applied, followed by the application of a mutation operator.

Selection favours the better individuals by ensuring that each individual’s probability of being selected is proportional to the fitness of that individual.

The probability of crossover taking place is predetermined and if crossover does occur, single-point crossover is applied from a random position in the individual.

Finally, each bit in a new individual is flipped, again with a predetermined probability.

#### 2.1.3 Tabu search

During each iteration the solutions in the neighbourhood of the current solution are considered as candidates for the next solution. Two different move functions are considered for use by the algorithm.

The first move function returns solutions where for each generating unit the start period  $x_i$  is moved to each possible starting period. In each case, only one start period is moved.  $(J - 1) * I$  solutions are considered.

The second move function is similar to the first, except for each unit only the periods adjoining the current start period are considered.  $2 * I$  solutions are considered.

For each function, only one generating unit is selected and its maintenance start period is changed. Thus it is possible to implement the cost function calculation by means of computing the change caused by the move rather than by evaluating the whole cost function.

Note that these move functions are deterministic, in that the neighbourhood is searched in a deterministic way. Non-deterministic (random) move functions were used in an initial implementation of the simulated annealing algorithm but it was found that the use of a deterministic search improved the algorithm’s effectiveness.

Two different methods of tabu classification were implemented. The first method compared state of the trial solution to that of the last  $N$  solutions, thus the algorithm cannot revisit a previous solution for  $N$  iterations. This is referred to as “solution tabu”.

The second method compared the move that took the current solution to the trial solution with those made in the last  $N$  iterations, thus the algorithm cannot perform a move more frequently than once in  $N$  iterations.  $N$  is called the tabu list size. This is referred to as “move tabu”.

When no improving solution has been found over the last  $n$  iterations the algorithm terminates. Typically this value is between 200 and 1000 depending on the desired quality of the solution and the size of the problem since for a larger problem, improving moves are likely to be found with a lower frequency.

An alternative termination criterion was considered: the algorithm stops after computing  $n$  iterations. This was found to be more dependant upon the problem size.

### 2.1.4 Tabu search / Simulated annealing hybrid

The addition of a tabu list to the simulated annealing algorithm ensures that recently visited states are not candidates for selection during each iteration.

The only difference between this algorithm and the simulated annealing algorithm described in section 2.1.1 is that once a neighbourhood solution is generated, it only becomes a candidate to be accepted if it does not appear in the list of recently accepted solutions.

## 2.2 Numerical Examples

To find out the optimal values for the problem parameters, a number of tests were performed. It was found that for this particular problem, best combination of search algorithm and tabu criteria was the first search algorithm and using ‘solution tabu’ where a solution is in tabu if it was visited during the last  $N$  iterations.

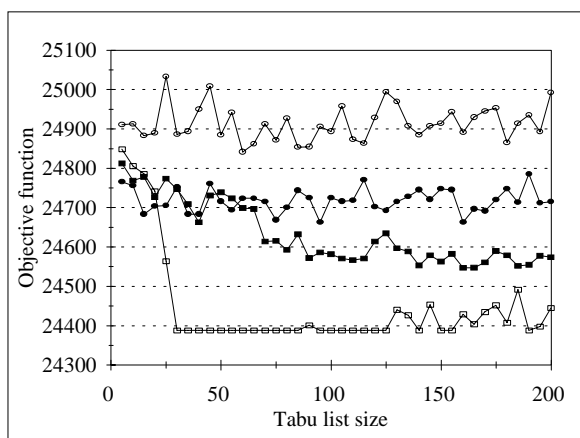


Figure 1: Tabu Search

This is illustrated in figure 1 where hollow markers indicate that the neighbourhood search function tried each move that could be made by moving the start period to every other possible period for each unit (search algorithm 1). Solid markers indicate that the neighbourhood search function tried to adjust the start period by

$\pm$  one period for each unit (search algorithm 2). Square markers indicate that the tabu criterion used was solution comparison and round markers indicate that the tabu criterion used was move comparison.

This figure also indicates that the ideal tabu list size is between 30 and 125. Below this range the solution is stopped from cycling and above this range, ideal solutions are not reached because too many solutions are in tabu and they cannot be reached. Solution aspiration may occur but it will still pick the solution whose cost value is greater than the other tabu solutions.

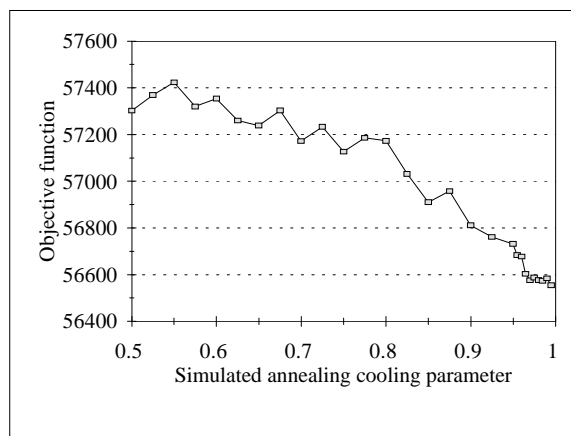


Figure 2: Simulated Annealing

Simulated annealing has only one tunable parameter, namely the cooling parameter. In figure 2 it can be seen that the quality of the solution increases as the cooling parameter approaches 1. There is a tradeoff between the quality of the solution and the length of time it takes for the algorithm to terminate since as the cooling parameter increases towards 1 it takes longer for the temperature to drop enough to cause the algorithm to terminate.

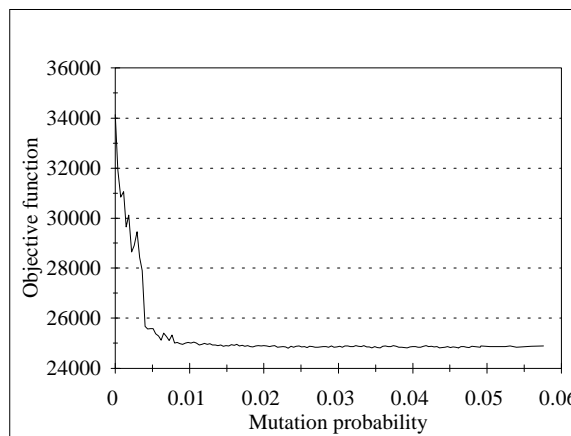


Figure 3: Mutation Probability

To perform tests with the parameter greater than 0.99 would take a very long time, also there are theoretical convergence results for this algorithm. It can be seen that there is little improvement to be gained by setting the

parameter higher than 0.97 so for this particular problem the results strongly suggest that this is the best value to use for the cooling parameter.

The genetic algorithm was the worst performing algorithm for solutions with large numbers of feasible solutions but performs slightly better than simulated annealing for problems with a small number of feasible solutions.

Tests were carried out to determine the optimum value for the parameters *GA\_crossover\_prob* and *GA\_mutation\_prob* for the hard 5-generator problem. These parameters were changed between ranges 0.0 to 1.0 and 0.0 and 0.05 respectively.

In figure 3 the mutation results are sorted into ascending order, and plotted against the objective function.

As can be seen, if the mutation parameter is close to zero the quality of the results is severely affected. It is not clear how the solution quality varies as the mutation parameter varies above 0.01. To rectify this, further tests were carried out where the mutation probability was varied in the range 0.01 to 0.2. The results are shown in figure 4.

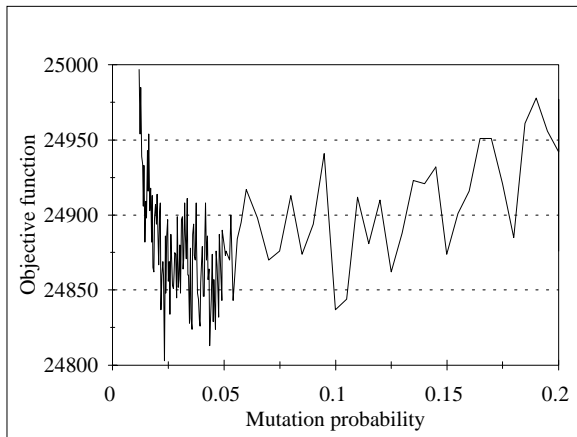


Figure 4: Mutation Probability

Although the variance of the results is quite large, the trend of the results is clearly increasing after the mutation parameter passes 0.05. The optimum mutation probability therefore is in the neighbourhood of 0.04 which is consistent with DeJong's recommendation [DJ75] that the ideal mutation parameter is 0.0333. Figure 5 plots the results from varying the *GA\_crossover\_prob* parameter. Solid markers indicate that the values were obtained by averaging all results regardless of the mutation parameter. Hollow markers indicate the result corresponding to where the mutation parameter was set to 0.04, i.e. the optimum value.

The optimum crossover parameter occurs at 0.1 where both the average and the individual result is better than for the other results. However the variation in the objective function as the crossover parameter is varied is significantly less than that observed as the mutation pa-

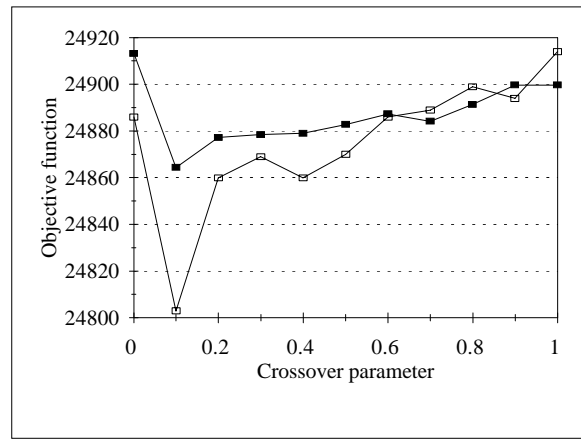


Figure 5: Crossover Probability

parameter is varied.

In fact it is suggested [SCED89] that the optimum probability for mutation is much more critical than that for crossover. In [SE91] this hypothesis was investigated further and it was found that crossover algorithms evolve much faster than mutation alone but mutation generally finds better solutions than algorithms that utilise only crossover. This is certainly true for the maintenance scheduling problem as seen above.

Mutation is important otherwise there is no escape (ever) from certain patterns, i.e. if a bit is 0 in all the population then crossover will not change it.

To conclude, the optimum parameters for the maintenance scheduling problem are 0.1 for the crossover parameter and 0.04 for the mutation parameter, however even the most optimal genetic algorithm performance does not approach the performance of either simulated annealing or tabu search.

The time analysis for genetic algorithms is less complex than that for simulated annealing since the algorithm terminates after a predefined number of iterations according to our implementation, although another stopping criterion could have been chosen. For a population size of  $n$  the objective function is evaluated  $n + 1$  times per iteration. Running genetic algorithms over 1000 iterations with a population size of 15 takes 3.94 seconds.

A number of improvements could be made to this algorithm, such as using an element of local search to improve each individual. Starting with a strong initial population could also prove useful. Finally, using dynamic crossover and mutation probabilities might also yield an improvement. For example, if the best solution has not improved for a number of generations then increasing the probability of mutation would ensure that the diversity of the population is maintained.

In order to compare the performance of the algorithm, the problem was also implemented using simulated annealing and a hybrid simulated annealing and tabu search algorithm where a tabu list is used as an additional move acceptance criteria. These algorithms were

System		TS 1		TS 2	
I	J	Cost	Time	Cost	Time
15	25	199157	58	199846	15
30	40	698020	110	699629	19
60	52	1350328	334	1357315	48

System		SA/TS		SA	
I	J	Cost	Time	Cost	Time
15	25	200255	12	200801	11
30	40	705450	20	706426	16
60	52	1359006	33	1359499	30

System		GA	
I	J	Cost	Time
15	25	201252	24
30	40	711543	31
60	52	1362343	72

Table 1: Comparison of the algorithms

implemented in C and run on an IBM RS/6000-530.

For very small problems it is the case that simulated annealing, the hybrid algorithm and tabu search all find the same solution. By enumeration the same solution was found, therefore the solutions obtained by using the three algorithms were optimal.

In the paper by Satoh and Nara [SN91] three problems of varying complexity are given and a comparison performed between simulated annealing and integer programming. Here a comparison is performed between tabu search, the hybrid simulated annealing/tabu search algorithm and simulated annealing. Table 1 shows the results obtained by using the four methods, where TS 1 uses search algorithm 1 in combination with ‘solution tabu’ and TS 2 uses search algorithm 2 in combination with ‘move tabu’. Times are shown in minutes.

Clearly tabu search generates the best results, at the cost of longer execution time. This can be reduced by using the second search algorithm since less of the neighbourhood is searched on each iteration. There is a tradeoff between two conflicting factors: execution time and solution quality. The combination of the first search algorithm and ‘solution tabu’ (TS 1) produces better results but at the cost of increased computation time whereas the second search algorithm and ‘move tabu’ (TS 2) yields slightly worse results but in a much faster time. The increase in time overhead involved with using TS 2 instead of one of the simulated annealing algorithms is very small, even for the large problem.

The utility of the tabu concept can also be seen in the simulated annealing algorithms where the addition of a tabu list to the simulated annealing algorithm yields an improvement in the quality of the result at no cost in terms of the execution time. This tabu list is an additional stage during trial solution generation where the trial solution is accepted with probability

$$\exp^{-\Delta/T_k}$$

only if it does not appear in the tabu list (or solution aspiration occurs).

This paper proposes to use tabu search to solve the problem of maintenance scheduling. Although tabu search does not guarantee to find a global optimum, with a suit-

able choice of search algorithm and tabu criteria, it is expected that the solution found will be sufficiently close to the global optimum. The numerical results show that the proposed method can be applicable to problems in real systems.

## References

- [DJ75] K. A. De Jong. An analysis of the behaviour of a class of genetic adaptive systems. *Dissertation Abstracts International*, 36(10):5140B, 1975.
- [DM75] J. F. Dopazo and H. M. Merrill. Optimal generator maintenance scheduling using integer programming. *IEEE Transactions on Power Apparatus and Systems*, PAS-94(5):1537–1545, 1975.
- [EDM76] G. T. Egan, T. S. Dillon, and K. Morsztyn. An experimental method of determination of optimal maintenance schedules in power systems using the branch-and-bound technique. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(8):538–547, 1976.
- [Gol89] David E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.
- [Hol75] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan, 1975.
- [KGV83] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimisation by simulated annealing. *Science*, 220:671–680, 1983.
- [Ree93] C. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell, 1993.
- [SCED89] J. D. Schaffer, R. A. Caruna, L. J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 51–60. Morgan Kaufmann, 1989.
- [SE91] J. D. Schaffer and L. J. Eshelman. On crossover as an evolutionary viable strategy. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann, 1991.
- [SN91] T. Satoh and K. Nara. Maintenance scheduling by using the simulated annealing method. *IEEE Transactions on Power Systems*, 6:850–857, 1991.
- [SR91] Youssef G. Saab and Vasant B. Rao. Combinatorial optimisation by stochastic evolution. *IEEE Transactions on Computer-Aided Design*, 10:525–535, 1991.
- [Č85] V. Černý. Thermodynamic approach to the travelling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–52, 1985.
- [Yam82] Z. A. Yamayee. Maintenance scheduling: Description, literature survey and interface with overall operations scheduling. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(8):2770–2779, 1982.
- [YSY83] Z. A. Yamayee, K. Sidenblad, and M. Yoshimura. A computationally efficient optimal maintenance scheduling method. *IEEE Transactions on Power Apparatus and Systems*, 102(2):330–338, 1983.
- [ZQ75] H. H. Zürn and V. H. Quintana. Generator maintenance scheduling via successive approximations dynamic programming. *IEEE Transactions on Power Apparatus and Systems*, 94(2):665–671, 1975.