A Multi-Stage Approach for the Thermal Generator Maintenance Scheduling Problem

E. K. Burke
Automated Scheduling and Planning Group
University of Nottingham
ekb@cs.nott.ac.uk

A. J. Smith
Automated Scheduling and Planning Group
University of Nottingham
ajs@cs.nott.ac.uk

Abstract- The thermal generator maintenance scheduling problem has been tackled by a variety of traditional optimisation techniques over the years. While these methods can give an optimal solution to small scale problems, they are often inefficient and impractical when applied to larger problems.

In this paper, we employ a multi-stage approach where the problem is decomposed into smaller sub-problems, each of which can be solved much more efficiently by existing algorithms. After each part of the problem has been solved, the results are then recombined to form the solution to the whole problem.

Both tabu search and a memetic algorithm have been observed to produce very good results but they take a significant amount of time to run. In this paper we utilise both techniques to form the basis of a multi-stage approach to solve the thermal generator maintenance scheduling problem.

The results will demonstrate that the multi-stage methodology is just as effective for this problem while achieving a significant reduction in run-time.

1 Introduction

The thermal generator maintenance scheduling problem presented by Yamayee [13] is concerned with scheduling essential maintenance over a fixed 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 [5], dynamic programming [15, 14] and branch-and-bound [6] have been proposed to solve this problem. For small problems these methods can 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.

Several modern heuristic methods have been applied to the problem. Examples are simulated annealing [12, 8], stochastic evolution [11], genetic algorithms [7] and tabu search [10]. In addition, the authors have successfully applied tabu search and a memetic algorithm to this problem and compared it to a selection of these methods [2], with the result that a memetic algorithm alone can produce quality solutions at the expense of extended run-time.

In this paper we will address the problem of extended run-times through the use of a multi-stage approach. Rather than attempting to solve the whole problem in one step, this approach separates the problem into a number of sub-problems, each of which can be solved consecutively and the results recombined to form the solution to the problem as a whole. It has been noted that for timetabling problems [1], the application of an algorithm to a collection of sub-problems can not only reduce the run-time required, but also arrive at higher quality solutions. This multi-stage techinque is similar to rolling-horizon methods, which have been applied to problems such as multipurpose plant scheduling [4], commercial fishing fleet scheduling [9] and tyre production scheduling [3].

2 Problem Description

A brief description of the problem is given below. A more detailed description of the formulation is given in [2].

Consider a collection of generating units producing output over a fixed planning horizon. Each unit must be maintained for a predetermined number of contiguous periods during the horizon. The demand forecast throughout the planning horizon is known, as well as the anticipated fuel costs.

In order to avoid random factors in the problem, such as unit random outages, a reserve capacity variable which is 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.

The objective function takes into account the running costs and maintenance costs of the plan, together with penalty functions to penalise infeasible or undesirable situations. An example of an infeasible solution would be a plan which attempts to exceed the operating capacity of the generating units.

3 The Multi-Stage Approach

3.1 Introduction

As mentioned above, Burke and Smith [2] have shown that both tabu search and a memetic algorithm can produce very good results on this problem, at the expense of increased run-time when compared to other techniques such as hill climbing or simulated annealing.

Following a run-time analysis of the algorithm, it was noted that the fitness function evaluation consumed the majority of the execution time. This was attributed to the fact that the fitness function evaluation requires the fitness for the entire solution to be recalculated at nearly every step. Data from previous evaluations was ignored. The only exception to this is where a neighbouring solution is being considered in which case the difference between two neighbouring solutions was calculated rather than recomputing the whole function.

It was further noted that each unit's contribution to the cost function is relatively static given that the unit's start position is constant. This separation of the objective function into the components contributed by each unit is crucial to the successful partitioning of the problem in that it becomes possible to cache partially-evaluated solution data for re-use. In view of this, a multi-stage approach was made feasible.

There is a potential danger with this technique in that there is a 'horizon' of problem data over which the algorithm cannot see. This may have the effect of causing a unit to be fixed into place too early, since information which might suggest that a better position could be found is unavailable to the algorithm. This danger can be addressed to a certain extent by choosing the sub-problems according to some measure of difficulty and by the introduction of a look-ahead set which expands the horizon of the algorithm.

Consider a list of generating units which has been ordered according to some measure of difficulty. The ordering would take the most difficult first. An example of the difficulty measure we might use is that we could order the units so that those with the least number of possible maintenance starting periods are placed first in the ordering. Another example might be that the units with the highest operating capacity are scheduled first. The intuitive idea is that when we split the problem up into a series of subproblems then we put all the most difficult units to schedule into earlier subproblems. Thus the danger of creating difficulties later on in the process is reduced.

The aim of the multi-staging approach is to pick the first N units and schedule them using some algorithm. All other units are left unscheduled. The next N units are then placed in the schedule, and so on, until all units have been scheduled. Therefore, each fitness function evaluation can re-use data acquired from previous evaluations very effectively. This approach differs from other rolling horizon approaches in that the problem is divided into subproblems which contain a reduced number of units to be scheduled, rather than sub-problems with a

reduced planning horizon (i.e. where the number of time periods is reduced).

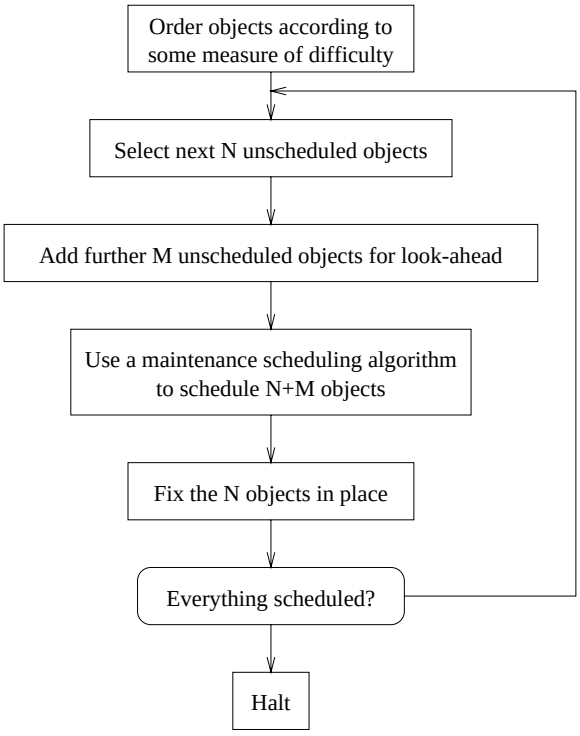

Figure 1: The Multi-Stage Algorithm

A look-ahead set is the addition of a further M units from the list to the units currently being scheduled. However at the end of the scheduling, only the first N units are fixed into the schedule. This enables the algorithm to schedule the current set of N units based not only on the units already scheduled (if any), but also to utilise additional information based from looking also at the next M units.

This combined approach is presented in Figure 1.

The multi-stage algorithm depends critically on two features of the problem. Firstly the problem must be divisible into sub-problems where each sub-problem can be solved individually with minimal reference to the remainder of the problem. This is to say that the amount of time taken to evaluate the objective function for a sub-problem of $\frac{1}{n}$ of the whole problem should be approximately $\frac{1}{n}$ of the time taken to evaluate the objective function for the whole problem.

Also, there should be a continuum of significance between the results obtained from a sub-problem and the application of those results to the problem as a whole. That is to say that the results for each sub-problem should be relevant with regard to the context of the problem as a whole.

Two maintainance scheduling algorithms are considered in this paper, a memetic algorithm and a tabu search algorithm (both briefly described below). As has been noted, they have both previously been found to perform well on this problem. However, it is important to point out that the crux of the method involves the successful partitioning of the problem into sub-problems which can be tackled with minimal interactions. In principle, any maintenance scheduling algorithm could be used to solve the sub-problems.

3.2 The Tabu Search Algorithm

The tabu search algorithm has been described in detail in [2]. A brief description is given here.

In each iteration, a neighbourhood of local solutions is defined. Each of these solutions is evaluated and the best is chosen to be the starting point to define the next local neighbourhood.

For each solution in the neighbourhood, 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.

The choice of the best solution is subject to the condition that previously visited solutions are "in tabu" or forbidden. This technique avoids both backtracking and oscillation between solutions close to a local optimum.

To determine whether a solution is in tabu, it is compared with the previous n solutions. If a match is found then that solution is forbidden, thus the algorithm cannot revisit a previous solution for n iterations.

3.3 The Memetic Algorithm

The underlying memetic algorithm for this approach has been described in detail in [2]. A brief description is given below.

A heuristic initialisation technique was used, which attempted to produce a solution where each

unit was scheduled in sequence, such that the number of units simultaneously in maintenance for each period is minimised.

The algorithm uses tournament selection, single-point crossover, and a combination of two mutation operators.

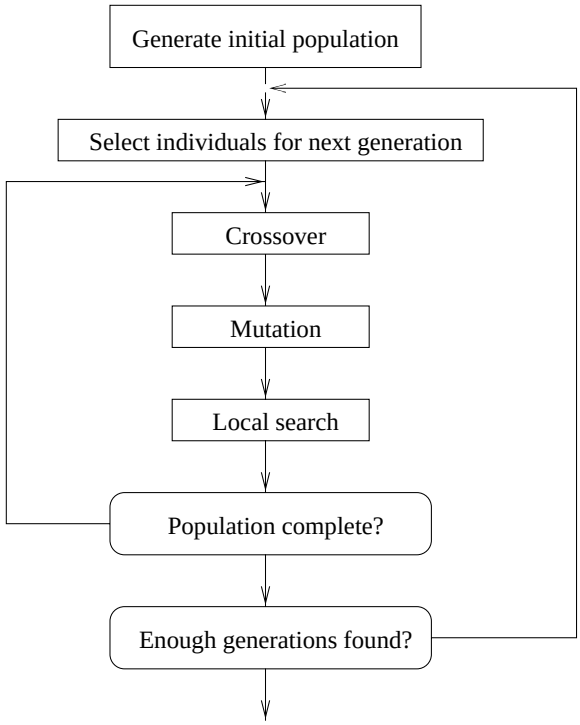

Figure 2: The Memetic Algorithm

A light mutation operator probabilistically flips bits in the chromosome. With a sufficiently small probability, this has the effect of increasing diversity while still retaining the potentially beneficial effects of the crossover operator.

This is followed by heavy mutation which attempts to target periods in which a certain penalty is incurred due to capacity being below the desired level. Each unit which is being maintained during these periods is rescheduled to a random starting period.

So far we have only described a genetic algorithm. A memetic algorithm differs from a genetic algorithm in that an individual is able to improve upon itself independantly of the remainder of the population.

A local search is employed as the process of an individual improving itself. A detailed series of experiments [2] found that a memetic algorithm employing tabu search as a local operator was the most effective approach. Thus tabu search is again employed in this implementation.

The steps of the memetic algorithm can be seen in Figure 2. This whole algorithm can be directly plugged in to the "maintenance scheduling" step of Figure 1.

3.4 Applying the Multi-Stage Approach

The problem as briefly described in Section 2 can be reduced to that of finding the optimal starting period of each generating unit such that the objective function is minimised. In this section we discuss how the multi-stage approach described above can be applied to the thermal generator maintenance scheduling problem.

The first task of the multi-stage algorithm is to order the units according to some measure of difficulty. In each application of the scheduling algorithm, it is operating with reduced information so it is important that the most difficult units be scheduled first. During the later stages of the multi-stage algorithm, it is expected that the easier units can fit into the remaining gaps.

Units which have to be maintained for a relatively long time within a relatively short period are given priority. Units which must be scheduled during higher demand periods are also given priority, as are units involved in timing constraints. These measures of difficulty can be used to derive an ordering of the units.

Both the scheduling algorithms described in Sections 3.2 and 3.3 can be applied to a subset of units in the problem.

The remainder of the units are either unscheduled, or are fixed in place by a previous application of the scheduling algorithm. Unscheduled units do not contribute towards the objective function.

Fixed units contribute towards the power output in a particular period. They also consume fuel and have an associated maintenance cost. By keeping a summary of this information for each period for the fixed units, it is possible to significantly reduce the objective function evaluation time for those units.

Furthermore, because each algorithm run is only considering a small number of units, a local optimum can be reached sooner and the runtime of the algorithm as a whole can be reduced.

3.5 Experimental results

To demonstrate the utility of the multi-stage approach, we will consider a parametrically generated problem, similar to those described in [2]. This problem concerns the scheduling of 60 generating units over 52 weeks.

To investigate the performance of the multi-stage tabu search and multi-stage memetic algorithm, each algorithm was run with varying set-size and look-ahead parameters. The effect on both solution quality and elapsed time can be seen in the following plots, where the set size is varied between 1 and 55, and the look-ahead between 0 and 54.

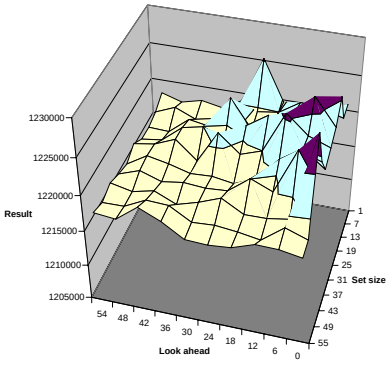

Figure 3: Tabu Search cost

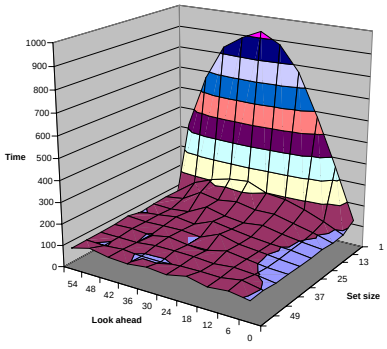

Figure 4: Tabu Search time

Figures 3 and 4 show the solution quality and elapsed time where the tabu search ran for 200 iterations during which the best solution was not improved upon. In the case that the set size plus look ahead encompasses the total number of units, the algorithm is able to consider the whole solution space at once. However, a reduced set size imposes a horizon that the algorithm is unable to see over. Thus, there is a corresponding drop in solution quality as the horizon shrinks.

A reduced set size together with a reasonable choice of look ahead leads to a reduction in running time.

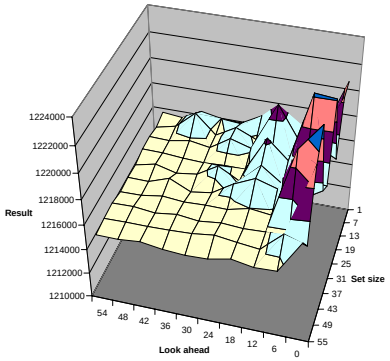

Figure 5: Memetic Algorithm Cost

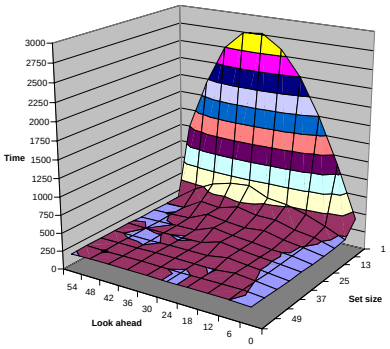

Figure 6: Memetic Algorithm Time

A similar result can be seen for the memetic algorithm in Figures 5 and 6. However the algorithm's stability is demonstrated on the left hand side of the graph where both the time taken and the resulting cost is close to constant. In this area, no multi-staging takes place, since the set size and look ahead together mean that the entire solution is being considered at once.

On reflection, it was observed that when the algorithm is considering a reduced problem, both scheduling algorithms are able to find a local optimum much faster than if they consider the whole problem at once. It was felt that both scheduling algorithms can be run for fewer iterations when considering a reduced problem size. To investigate this hypothesis, the number of iterations was reduced so that the number of iterations performed by each scheduling algorithm is proportional to the number of units currently being scheduled. This caused a large reduction in search time while the solution quality remained mostly static. This is illustrated in Figures 7 and 8 for the tabu search algorithm, and in Figures 9 and 10 for the memetic algorithm.

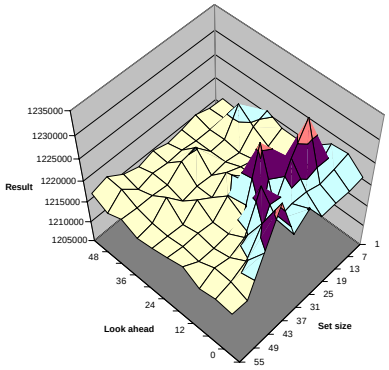

Figure 7: Tabu Search cost

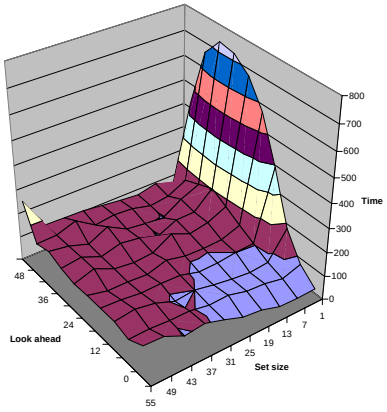

Figure 8: Tabu Search time

For the tabu search algorithm, there was a small decrease in solution quality in the region where the set size plus look ahead was equal to roughly half the total number of units. To compensate for this,

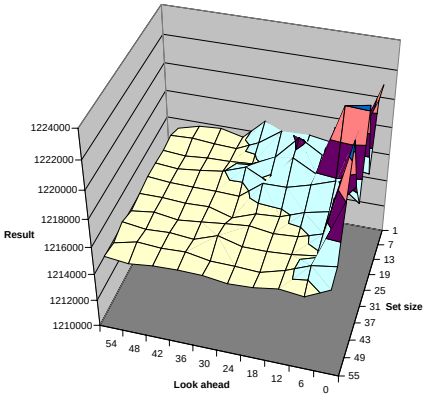

Figure 9: Memetic Algorithm Cost

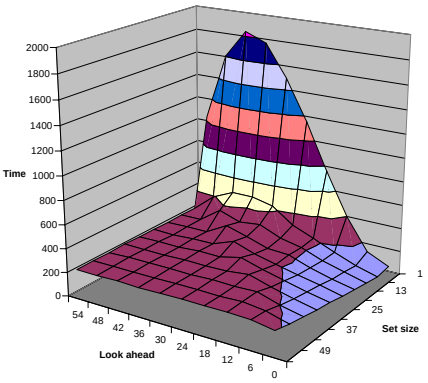

Figure 10: Memetic Algorithm Time

the number of iterations was amended by the addition of an extra component based on the distance away from this area. The results obtained from this method are illustrated in Figures 11 and 12.

A similar approach was attempted for the memetic algorithm, as can be seen in Figures 13 and 14 but the results were less encouraging. Additional iterations merely slowed the algorithm and resulted in little improvement in solution quality.

In Table 1, selected numerical results are shown.

Tabu Search	Cost	Time	Set size	Look ahead
Best one-pass	1217593	102	-	-
Least cost	1216137	138	11	30
Least time	1220430	11	1	0
Best result	1220077	12	6	0
Memetic Algorithm	**Cost**	**Time**	**Set size**	**Look ahead**
Best one-pass	1215243	249	-	-
Least cost	1215308	318	21	25
Least time	1220873	28	1	0
Best result	1215698	82	11	0

Table 1: Numerical results

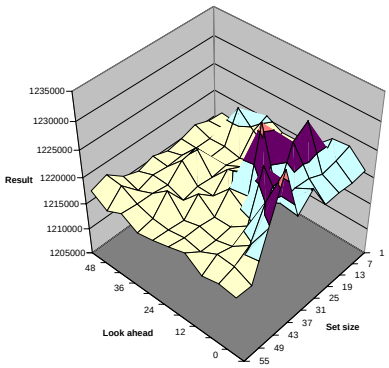

Figure 11: Tabu Search cost

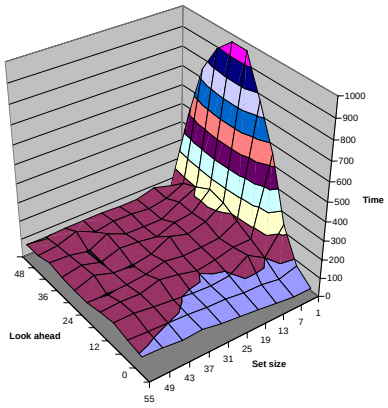

Figure 12: Tabu Search time

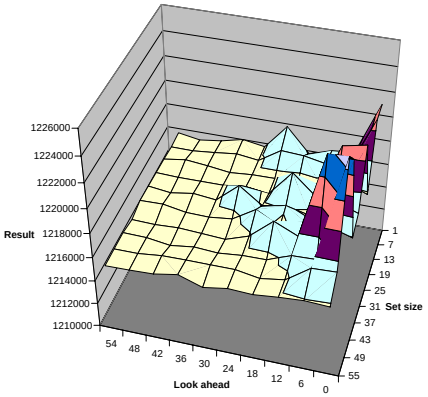

Figure 13: Memetic Algorithm Cost

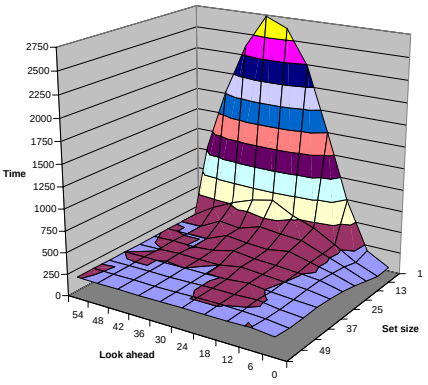

Figure 14: Memetic Algorithm Time

For each scheduling algorithm the best one-pass result is shown, along with the least cost and least time multi-stage results. Finally the best multi-stage result is shown, where the criteria is finding the least cost in least time.

4 Conclusions

We can see that there is a clear advantage in setting the number of iterations to be proportional to the number of units being considered at each stage. The most pronounced effect is where the set size is relatively small. The runtime of the algorithms is reduced to a tenth, while solution quality is unchanged.

The results as a whole reflect the fact that both algorithms are hampered due to the limited number of units that they can schedule at once. Where the set size and look ahead means that the algorithm can schedule every unit in one pass, the results are reasonably uniform and the runtime is also reasonably constant. The consistency of this result is greater in the memetic algorithm than the tabu search algorithm.

As the amount of information the algorithm has at each stage is reduced, the solution quality falls, although this is less dramatic as one might expect. The decrease is rarely more than 1%.

Another feature to note is the rapid increase of time when units are present in the look ahead set for many stages of the algorithm before being fixed into place. This can be clearly seen where the set size is small and the look ahead is about half. The elapsed time rises remarkably when the set size approaches 1, especially where the look ahead set is non-trivial. This can be seen intuitively - the algorithm repeatedly finds a good schedule for a large number of units, but only fixes in place a small number, hence the majority of units are rescheduled many times.

To compare the memetic algorithm against the

tabu search algorithm as the scheduling operator, it can be seen that the solution quality achieved by the memetic algorithm is much more consistent where the set-size and look-ahead mean that the solution is fixed in one pass. It can also be seen that as the set size is reduced, the decrease in solution quality when the memetic algorithm is employed can be much less than the decrease observed when using the tabu search as the scheduling operator.

From Table 1, the memetic algorithm clearly outperforms the tabu search algorithm in terms of solution quality, however tabu search can often find a result in a shorter time.

In conclusion, with a small set size and little or no look ahead, almost a ten-fold decrease in run-time can be achieved with minimal increase in solution cost. In these experiments, the elapsed time could be reduced from just under five minutes down to just under half a minute on a Pentium 200 workstation. In a problem where the problem size is much greater (and in practice they usually are), the application of even a meta-heuristic technique which attempts to schedule every unit in just one pass is restricted. However, a multi-stage approach could still be usefully applied to such a problem. For example, a problem for which a one-pass technique would require a ten week run-time could be completed in just over one week. This could render previously infeasible problems practical.

Future work could investigate whether the multi-stage approach proposed in this paper can be successfully parallelised. Currently the scheduling algorithm in each 'stage' is performed sequentially. On a multi-processor system it should be possible to schedule multiple stages simultaneously and achieve a further reduction in run-time.

Bibliography

[1] E. K. Burke and J. P. Newall. A multi-stage evolutionary algorithm for the timetable problem. To appear in *IEEE Transactions on Evolutionary Computation*, 3(1), 1999.

[2] E. K. Burke and A. J. Smith. Hybrid evolutionary techniques for the maintenance scheduling problem. To appear in *IEEE Power Engineering Society Transactions*, 1999.

[3] Z. Degraeve and L. Schrage. A tire production scheduling system for Bridgestone/Firestone Off-The-Road. *Operations Research*, 45(6):789–796, 1997.

[4] A. D. Dimitriadis, N. Shah, and C. C. Pantelides. Rtn-based rolling horizon algorithms for medium term scheduling of multipurpose plants. *Computers and Chemical Engineering*, 21(Suppl.):1061–1066, 1997.

[5] 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.

[6] 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.

[7] David E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

[8] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimisation by simulated annealing. *Science*, 220:671–680, 1983.

[9] H. H. Millar. The impact of rolling horizon planning on the cost of industrial fishing activity. *Computers and Operations Research*, 25(10):825–837, 1998.

[10] C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell, 1993.

[11] Youssef G. Saab and Vasant B. Rao. Combinatorial optimisation by stochastic evolution. *IEEE Transactions on Computer-Aided Design*, 10:525–535, 1991.

[12] V. Černy. Thermodynamic approach to the travelling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–52, 1985.

[13] Z. A. Yamayee. Maintanance scheduling: Description, literature survey and interface with overall operations scheduling. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(8):2770–2779, 1982.

[14] 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.

[15] 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.