Searching for Gas Turbine Maintenance Schedules

Markus Bohlin, Kivanc Doganay, Per Kreuger, Rebecca Steinert, and Mathias Wärja

■ *Preventive-maintenance schedules occurring in industry are often suboptimal with regard to maintenance coallocation, loss-of-production costs, and availability. We describe the implementation and deployment of a software decision support tool for the maintenance planning of gas turbines, with the goal of reducing the direct maintenance costs and the often costly production losses during maintenance down time. The optimization problem is formally defined, and we argue that the feasibility version is NP-complete. We outline a heuristic algorithm that can quickly solve the problem for practical purposes and validate the approach on a real-world scenario based on an oil production facility. We also compare the performance of our algorithm with results from using integer programming and discuss the deployment of the application. The experimental results indicate that down time reductions up to 65 percent can be achieved, compared to traditional preventive maintenance. In addition, the use of our tool is expected to improve availability by up to 1 percent and to reduce the number of planned maintenance days by 12 percent. Compared to an integer programming approach, our algorithm is not optimal but is much faster and produces results that are useful in practice. Our test results and SIT AB's estimates based on operational use both indicate that significant savings can be achieved by using our software tool, compared to maintenance plans with fixed intervals.*

Preventive maintenance can reduce breakdowns and the costs associated with them but is also costly when done frequently. That is why substantial efforts have been invested in minimizing the expected total cost due to failures and preventive maintenance of industrial equipment (Dekker 1996, Tan and Kramer 1997, Bäckert and Rippin 1985, Dekker and Scarf 1998). In industry, most preventive-maintenance approaches include the use of fixed schedules, optimized in advance for minimum cost. However, there are many situations in which maintenance replanning is necessary to operate with as low cost as possible. For example, unexpected breakdowns force the production unit to stop for emergency repair, and performing other maintenance tasks at the same time can save time and money. In addition, the increasing use of condition monitoring leads to more unpredictable maintenance events and a need for dynamic planning.

In this article, we present the ideas behind the preventive-maintenance optimizer (PMOPT) tool used to optimize gas turbine maintenance schedules. We developed PMOPT for Siemens Industrial Turbomachinery AB (SIT AB), one of the leading manufacturers of gas turbines of small and medium size. Gas turbines are used for power generation in various production facilities that often have high down time costs. A gas turbine in the form of a jet engine is depicted in figure 1. Axial flow gas turbines (including many industrial gas turbines) are constructed using a *compressor*, which produces compressed air to a *combustion chamber*, where fuel is injected. The resulting mixture is

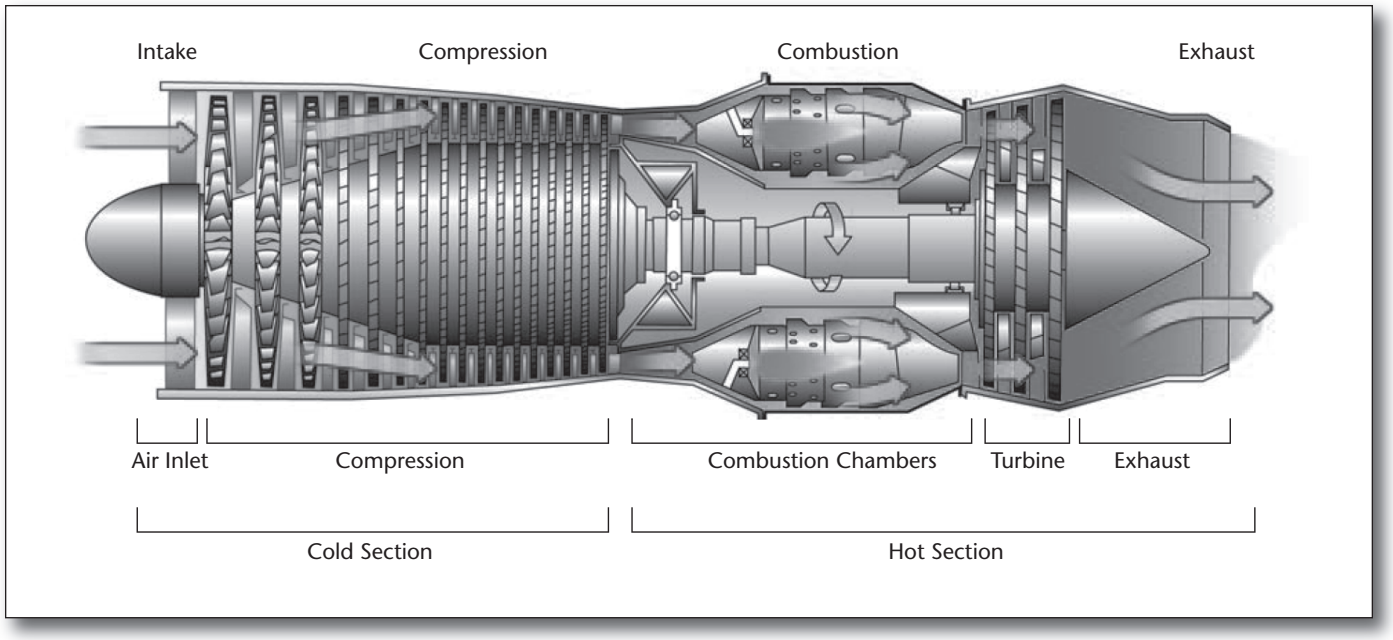

Figure 1. Schematics of a Gas Turbine.

Image taken from figure 14-1, *Airplane Flying Handbook,* U.S. Federal Aviation Administration 2004.

ignited, thereby increasing the volume and velocity of the gas flow, in turn driving the *turbine*. Since the turbine is coupled to the compressor, the combustion cycle is sustained. A typical application of industrial gas turbines is offshore oil platforms, where power outage can cause an extremely high loss of revenue. In such applications, small improvements in terms of overall availability, which is one expected outcome of implementing condition-based maintenance (CBM), have a substantial positive effect on the total income for the customer.

Condition-based gas turbine maintenance, where component lifetime is dependent on factors such as load profile, quality of fuel, ambient temperature, and particle levels, is therefore becoming more and more common. Although lifetime predictions can sometimes be performed with high precision, the predicted maintenance dates will sooner or later diverge from their estimates depending on on-site conditions.

The approach we present in this article provides the means to quickly optimize maintenance when unplanned events make the previous schedule unsuitable. We use a rolled-out representation of the predicted future maintenance schedule and take positive effects of co-allocation, overall availability due to preventive maintenance, horizon effects, and costs due to both maintenance and loss of production into account. Since proper risk analysis and deterioration model identification is in many practical cases difficult to perform from scratch, maintenance intervals are often based on analytical models and "best practice." In PMOPT, we assume that a safe deadline for all maintenance activities exists, which simplifies the problem and makes it easy to adapt already existing maintenance plans for use in PMOPT.

The contributions of this article include that we (1) precisely define the maintenance-scheduling problem discussed, (2) argue that the planning problem is NP-complete, (3) outline an algorithm that can quickly solve the problem for practical purposes, (4) show results for a real-world scenario, (5) compare the results of our algorithm to the results from using integer programming, and (6) discuss the implementation and deployment of PMOPT.

Related Work

Maintenance optimization has been an active research area for a long time; Dekker (1996) gives a good overview of the different applications. The work most closely related to our article is concerned with maintenance scheduling where economic dependencies between activities exist. Most researchers seem to use a common setup cost s for all maintenance activities performed at a single maintenance occasion. The cost saving for grouping n maintenance activities together then becomes $(n-1)s$. In this article, we propose a model of indirect economic dependence between activities, where the economic effect of joint execution

depends on the resulting down time of the maintenance stop. The down time in turn depends primarily on the length of the activities at the stop and whether activities can be performed in parallel or not.

As an example of the former approach, Wildeman, Dekker, and Smit (1997) discuss the type of economic dependencies with one single shared setup cost and in addition propose a polynomial solution approach to the scheduling problem. The polynomial solution is optimal if the optimal groups are always *consecutive*, that is, the activities occur in the same order as their locally optimal date. In the model proposed in this article, it may very well be optimal to group activities nonconsecutively if the earnings from doing so outweigh the costs. Van Dijkhuizen and van Harten (1997) propose a generalization to a tree-shaped setup structure. Another approach is given by Yamayee, Sidenblad, and Yoshimura (1983), who use dynamic programming to optimize maintenance with respect to equipment reliability, demand of generating units, and maintenance cost. The main difference compared to our work is that Yamayee et al. consider large-scale scheduling of power-generating units, whereas we focus on single-unit multicomponent maintenance, for which a detailed down time model is beneficial.

Tan and Kramer (1997) consider opportunistic maintenance in the chemical processing industry. Monte Carlo simulation is used to estimate costs, which allows a generic cost structure. In this approach, it is possible that an event with a low probability but a very high cost is not sampled and therefore not taken into account. Another difference compared to our work is that, in addition, we consider nonhomogenous loss-of-production costs. The model we employ therefore allows us to optimize maintenance for scenarios that include periods with both lower than normal down time cost and with a higher cost. This is important in many industrial areas, for example, in the oil and gas industry. Marseguerra, Zio, and Podofillini (2002) also use Monte Carlo simulation and genetic algorithms and, in addition, consider other properties such as the number of maintenance technicians available. However, other economic dependencies between components are not considered.

Background

The common practice of gas turbine maintenance planning today is to base the schedules on equivalent operating hours (EOH) and cycles (number of turbine restarts). In calculating EOH, we modify the number of operating hours with factors for load, fuel quality, presence of water injection, and (to a limited extent) significant exhaust temperature differences. However, the model is not detailed enough in how these variables are handled, and factors such as ambient air temperature and pressure, rotational speeds, and more detailed outlet temperatures are not included. Instead, the EOH calculations have built-in safety margins to accommodate for variables not explicitly modeled.

To improve overall maintenance efficiency, new calculations for estimating the remaining lifetime of gas turbine components based on operation profile, environmental conditions, and condition data (obtained through inspections and sensors on the gas turbine) was developed by SIT AB. A lifetime prediction tool, producing deterministic lifetime estimates, was also developed. The lifetime estimates produced by the tool include relevant safety margins. Therefore, changes in lifetime do not affect risk levels negatively as long as the gas turbine is maintained within its predicted service period. In fact, risk levels are in many cases dramatically reduced, since the lifetime prediction tool also detects and decreases maintenance intervals for gas turbines operating under conditions with increased component wear (for example high load, high humidity, or low fuel quality).

Maintenance Activities

In this article, we assume that the optimal maintenance intervals (t_i^{opt}) for the individual components have already been calculated. We use the term *replacement* of a component to represent repair to a state that is as good as new, while inspections (including servicing, lubrication, on-condition, and failure-finding tasks) are considered "as-good-as-old" maintenance and do not affect component lifetime.

Since replacements restore component lifetime fully, it is suboptimal to schedule inspections independently of the performed replacements. The scenario of independent inspections and replacements is shown in figure 2, where a total of five inspections are made. In contrast, the scenario used in this article is shown in figure 3. Here, only three inspections are needed; the replacements that occur in between restore component status, and therefore, fewer inspections are needed. Note that truly independent inspections can still be modeled using a separate artificial component containing only inspection items.

The Gas Turbine Maintenance Process

In this section, we outline a workflow for the use of gas turbine maintenance planning. The process is dependent on both online lifetime predictions using a prognostics tool and on-demand maintenance optimization using the PMOpt scheduling tool. The workflow shown in figure 4 starts with

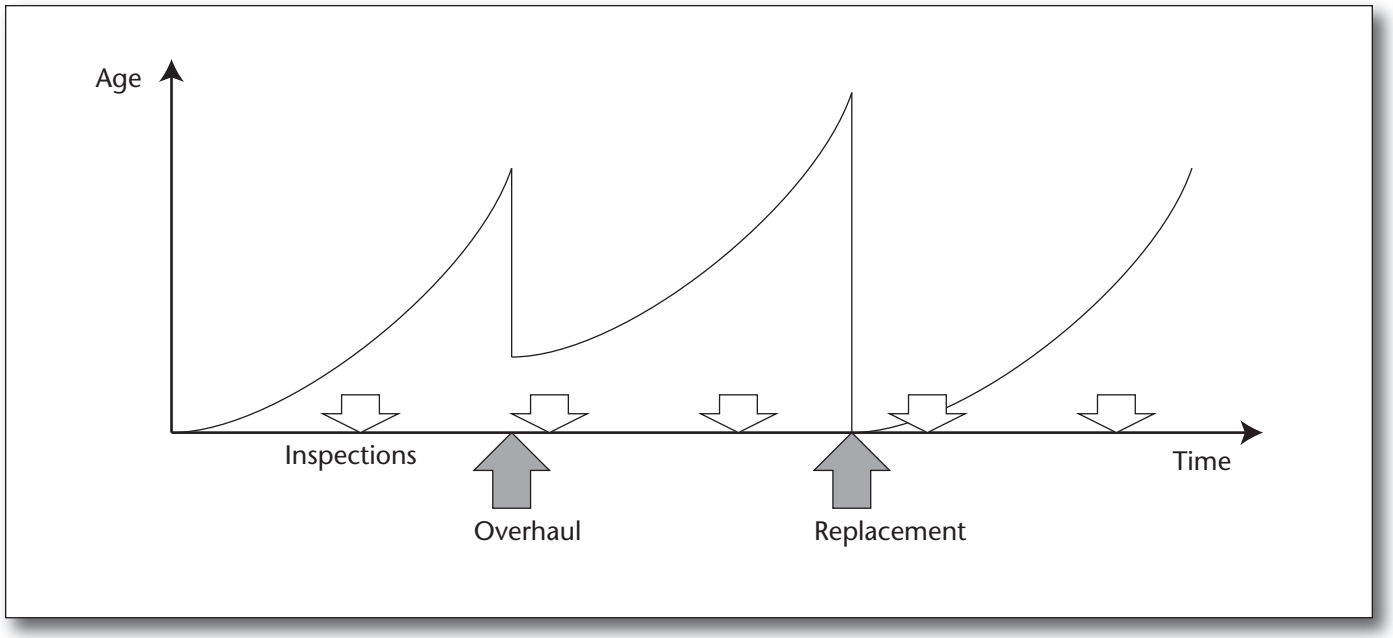

Figure 2. Unrelated Replacements or Overhauls and Inspections of a Single Component.

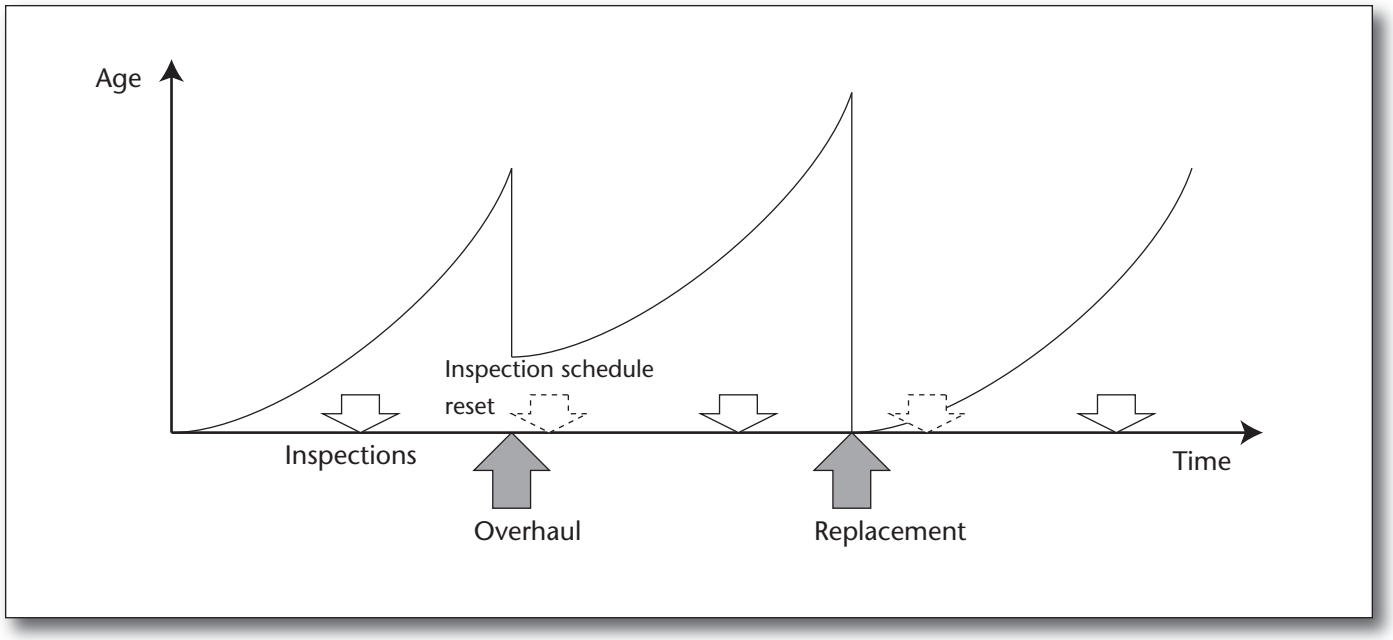

Figure 3. Synchronized Replacements or Overhauls and Inspections, Resulting in the Elimination of Unnecessary Inspections (dotted).

identification and specification of single and composed components and corresponding items. If an item can belong to more than a single component, we either split the item into separate items for the components in question or put the item in a generic component schedule existing solely for that purpose.

Independent of this activity, production planning begins so that requirements from production can be used later in specifying constraints for the maintenance optimization. As an intermediate step, initial lifetime predictions are carried out using EOH and start-stop cycle estimates for components without damage accumulation algorithms

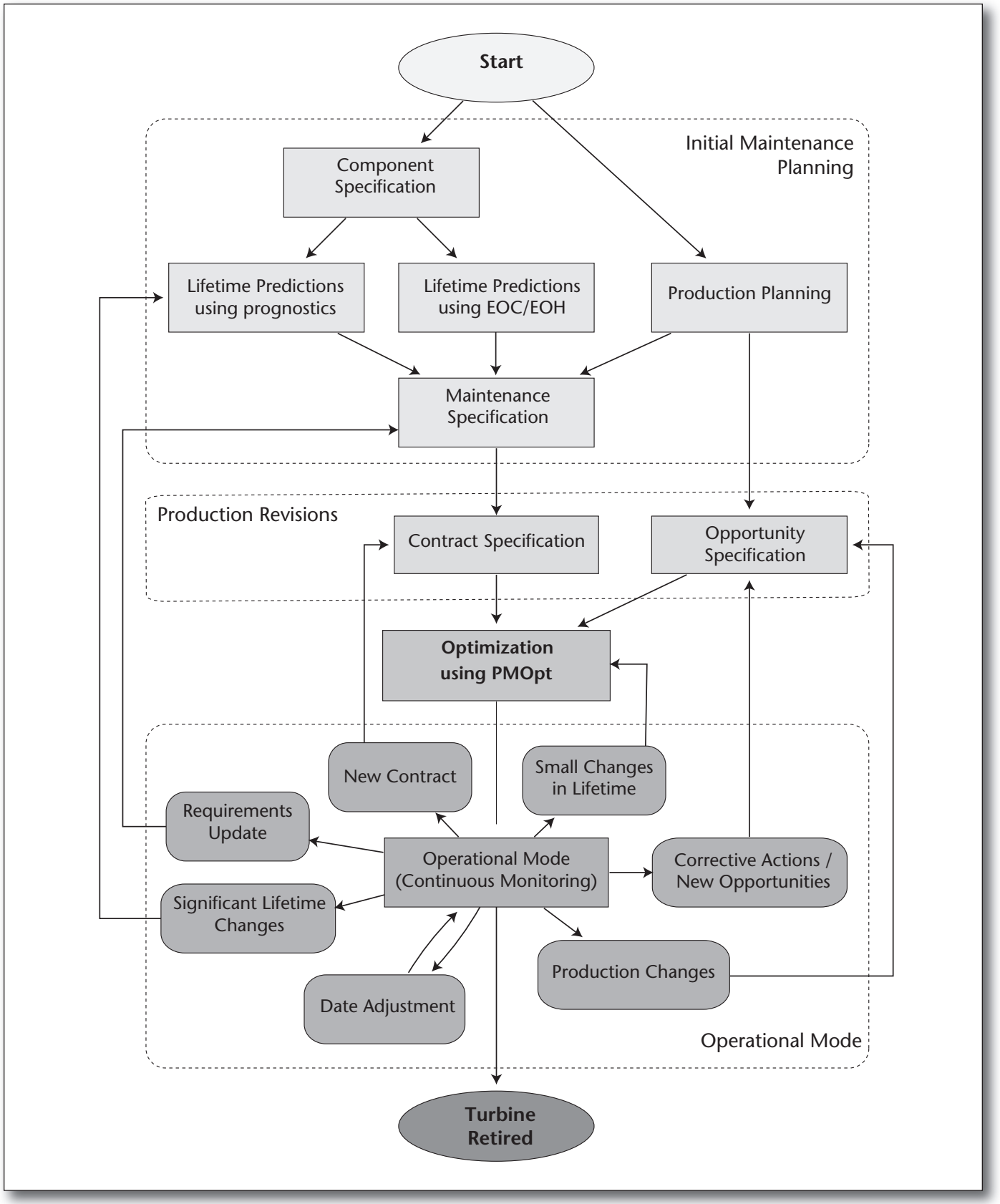

Figure 4. Workflow of Maintenance Planning using PMOpt.

α	Minimum required availability.
b_j	Base cost of opening opportunity j.
v_j	Maximum work time at opportunity j.
δ_j	Date of an opportunity j.
$\mathbf{t}[i]$	Maintenance date of item i.
t_i^{opt}	Locally optimal time of maintenance for item i.
t_i^{dl}	Deadline of item i, relative to p_i.
Δ_{pi}	Duration in phase $q \in 1..p$ for item i.
t_i^{min}	Earliest possible date for item i.
h	Scheduling horizon.
$\perp$	Artificial item representing the schedule start.
c_i	Cost of item i.
$\top$	Artificial item representing the schedule end.
$k^{\max}$	Maximum discrepancy parameter tried in LDS.
$t_i^{\max}$	Latest possible date for item i.
l_i	Loss of production cost at opportunity j.
$\mathcal{E}$	The set of head items.
$\mathcal{L}$	The set of tail items.
obs(i)	Predicate indicating whether item i is obsolete.
p_i	Predecessor of item i.
r_j	Resting time att opportunity j.
t_i^{rt}	Release time of item i, relative to p_i.
s_i	Terminator of item i.
u_j	Total work time at opportunity j.
A	The number of working hours per day.
b	Maximum branching factor in the search tree.
d	Remaining depth of the search tree.
k	LDS discrepancy parameter.
m	Number of opportunities.
n	Number of items.
p	Number of phases.
w	Weight for cost due to unused lifetime.

or using prognostic tools for the other components.

From the lifetime predictions and production requirements, a *maintenance specification* is then constructed using PMOPT. It is important to point out that as many requirements as possible are formalized in the specification. At the same time, a list of known recurring and one-time maintenance opportunities is made together with durations and estimates on the cost of down time during the opportunities. Further, a *contract specification* is needed, formalizing the contract period and availability requirements. Some of the contract requirements, for example calendar constraints, may also have to be put in the maintenance specification; to avoid clutter, these requirements are not shown in figure 4.

Given the output of the process so far, PMOPT is now ready to produce initial maintenance schedules. During production, PMOPT is used continually and updated with the latest information as soon as it is available. For example, PMOPT needs to know which part of the maintenance plan has already been executed and, consequently, which items are still pending. Unexpected corrective actions and maintenance opportunities are inserted manually together with suitable adjustments on the remaining lifetime of the components affected. Production changes mean that the future maintenance opportunities, costs, and priorities change, and such changes must be fed into PMOPT to produce relevant results. When the maintenance plan of the gas turbine is changed for other reasons not part of the CBM process, PMOPT still needs new maintenance schedules, so that maintenance is planned according to the current state of practice.

Scheduling Maintenance with Downtime Costs

We informally describe the maintenance scheduling with opportunities problem (MSOP) as the problem of allocating n maintenance items to m dates (opportunities) for a set of independent components in a single unit and for a time period of h weeks, so that constraints on timeliness, maintenance duration, and total availability (due to preventive maintenance) are satisfied. We assume that maintenance of any component means that the entire unit has to be shut down. The allocation should minimize direct and indirect maintenance costs, including spare parts, labor, and the value of production that is lost due to maintenance. In the rest of this section, we use terminology depicted in the adjacent shaded box.

Each component has a cyclical schedule of arbitrary length, consisting of *replacement* (R_x) and *inspection* (I_{xy}) items. The date of a replacement depends on the previous replacement, while inspections depend on the previous item regardless of type. We assume that the given component schedules are followed; deviations are taken into account by updating the schedule data.

Duration and Downtime Models

In maintaining a gas turbine, the work included in a single maintenance item can often be divided into different *phases*, for example the phases shutdown and cooling, dismantling, repair, reassembly, and startup, which are shown in figure 5.

To estimate work time at a maintenance stop, each maintenance item therefore has a *duration specification* $\Delta_i = \langle \Delta_{1i}, \Delta_{2i}, \ldots, \Delta_{pi} \rangle$ dividing work into p nonnegative work phases. All activities within a single phase at a single stop are assumed to be fully independent and can therefore be executed in parallel. In contrast, the phases themselves are done in an orderly fashion. The total work time u_j of a stop can thus be computed as the sum of the maximum work time in each block:

$$u_j = \sum_{1 \le q \le p} \max_{\substack{1 \le i \le n \\ i \text{ performed at } j}} \Delta_{qi} \qquad (1)$$

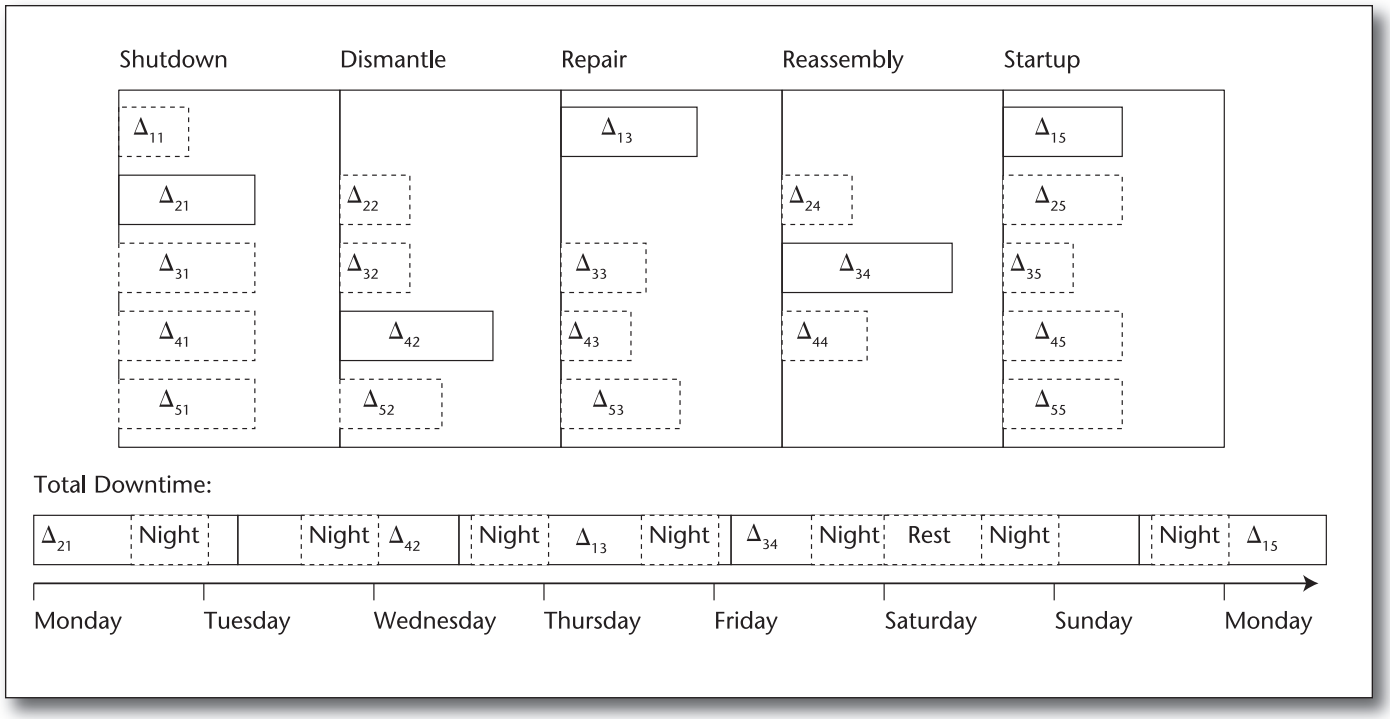

Figure 5.

Duration and down time model, where work is divided into phases in which activities can be executed in parallel. Resting time is then added. We assume that Mondays are spent travelling.

An example is given in figure 5, where the five duration specifications $\Delta_1 = \langle 8, 0, 13, 0, 12\rangle$, $\Delta_2 = \langle 13, 7, 0, 7, 12\rangle$, $\Delta_3 = \langle 13, 7, 8, 17, 8\rangle$, $\Delta_4 = \langle 13, 15, 7, 8, 12\rangle$ and $\Delta_5 = \langle 13, 10, 12, 0, 12\rangle$ are jointly performed at the same stop. The working time for the different phases then becomes $\langle 13, 15, 13, 17, 12\rangle$, and the total work time at the stop is then 70 hours.

Given the total work time at a stop, we can now compute the down time by adding *resting time* for the maintenance crew. We assume that a working day consists of A hours and that all calendar weeks (consisting of 6 working days) are alike. The resting time at an opportunity consists of night and week rest time and is computed as follows:

$$r_j = \begin{cases} (24 - A)\left\lfloor \frac{u_j}{A} - 1 \right\rfloor + 24\left\lfloor \frac{u_j}{6A} - 1 \right\rfloor & \text{if } u_j > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The total down time is the sum of working time and resting time. Continuing our example in figure 5 and assuming that $A = 10$, we obtain a resting time of 108 hours and a total down time of 178 hours.

Scheduling Model

We assume that n maintenance items i have been rolled out to cover weeks 1 to h (the *horizon* of the problem). The decision variable $\mathbf{t}[i]$ represents the date of item i. The schedule end is modeled by the artificial item $\top$ at date $h + 1$, and the schedule start is modeled by another artificial item $\bot$ at date 0. The possible allocation dates within the schedule are modeled by a set of opportunities $j \in 1..m$ with dates δ_j and maximum work time v_j.

The different timeliness constraints in the problem are illustrated in figure 6 and can be expressed as follows. Each item i has a *release time* t_i^{rt} and a *deadline* t_i^{dl} relative to i's *predecessor* p_i. Each item also has an optional *earliest* and *latest date* of execution ($t_i^{\min}$ and $t_i^{\max}$). After a replacement, a sequence of inspections should follow before a new replacement is made. Similarly, the schedule should contain enough replacements to cover the scheduling period, but extra replacements should not be taken into account. We call rolled-out items that do not have to be executed *obsolete* items. For this purpose, each item i has a *terminator* s_i that makes i obsolete if i is done later or at the same date as s_i. For simplicity, we force obsolete items to be allocated to the same date as their terminator. Formally, we define the predicate obs(i), with the meaning that item i is made obsolete by its terminator s_i, as follows.

$$\text{obs}(i) \equiv \mathbf{t}[i] = \mathbf{t}[s_i] \quad (3)$$

Replacements always have $\top$ as their terminator, which implies that they are obsolete when they are

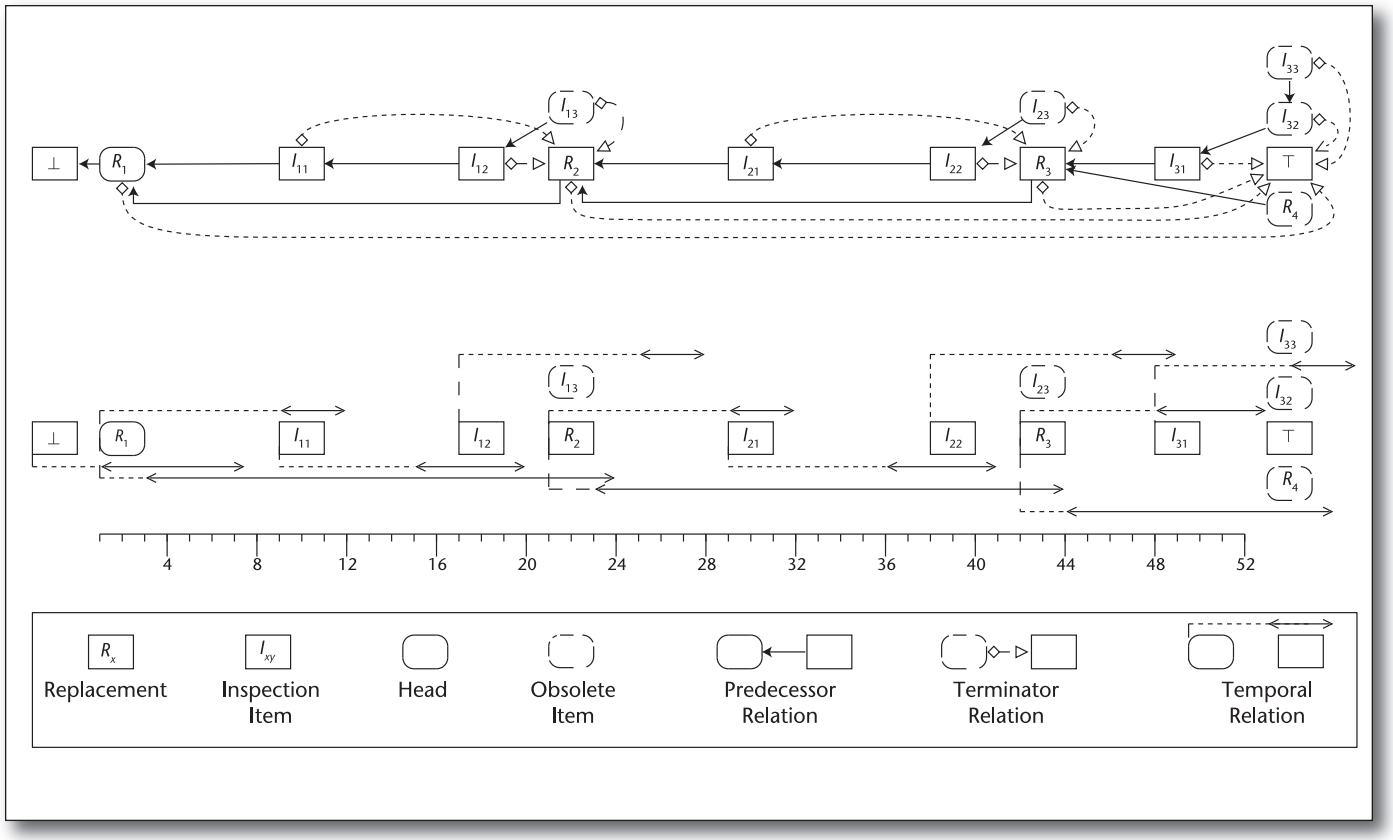

Figure 6.

Dependencies (top) and relative timeliness constraints (bottom) between different maintenance items of a component.

not performed before the problem horizon h. Figure 6 illustrates relative timeliness constraints between pairs of tasks as well as predecessor and terminator relationships in a fictional schedule. The set of items that are first in the schedule for each component is called the set of *head* items, and is denoted $\mathcal{E}$. All head items have $\perp$ as their predecessor.

To ensure that the gaps after sequences of items are not too large, we use special items representing the end of such sequences. We call such items *tail* items. The set of tail items $\mathcal{L}$ consists of (1) the last replacement for each component and (2) the last item in each inspection sequence. To ensure that the generated schedules cover the scheduling period, we force all tail items to be obsolete. The deadline constraints then ensure that the gaps at the end of an inspection or replacement sequence are smaller than required.

Each item also has an *item cost* c_i consisting of work and material cost. In addition, the value of production per hour at an opportunity j is denoted l_j. We also use a *base* cost b_j for opening up opportunity j. The base cost is associated with shared setup costs not related to the down time of the opportunity and includes some of the costs for shutting down and restarting the gas turbine, travel expenses, and other shared costs that cannot be modeled using material, work, or down time costs. Minimum availability is specified using the parameter α (where $0 \le \alpha \le 1$). The total availability is defined as the time not spent on preventive maintenance divided by the total available time.

The constraints can now be formally stated.

Each item i should be allocated to a date less than or equal to its deadline.

$$\forall i \in 1..n : = \mathbf{t}[i] \le \mathbf{t}[p_i] + t_i^{\mathrm{dl}} \tag{4}$$

Each item has to respect its absolute allocation interval.

$$\forall i \in 1..n : = t_i^{\min} \le \mathbf{t}[i] \le t_i^{\max} \tag{5}$$

Each tail item has to be obsolete.

$$\forall i \in \mathcal{L} : \mathrm{obs}(i) \tag{6}$$

Each nontail item i should be either obsolete or allocated to a date larger than its offset.

$$\forall i \in 1..i \setminus \mathcal{L} : \mathrm{obs}(i) \vee \mathbf{t}[i] \le \mathbf{t}[p_i] + t_i^{\mathrm{rt}} \tag{7}$$

For each opportunity j, the work time u_j of j must be lower than the maximum work time v_j.

$$\forall j \in 1..m : u_j \leq v_j \tag{8}$$

The availability of the plan should be greater than the minimum availability α.

$$\frac{1}{7 \cdot 24} \sum_{j \in 1..m} u_j + r_j \leq h(1-\alpha) \tag{9}$$

The objective of the optimization problem is to minimize the cost function f, defined as follows:

$$f(\mathbf{t}) = \sum_{\substack{j \in 1..m \\ \neg \text{obs}(i)}} c_i + \sum_{j \in 1..m} l_j(u_j + r_j) + \sum_{\substack{j \in 1..m \\ \exists i \in \mathbf{1}..m : \mathbf{t}[i] = \delta_j}} b_j \tag{10}$$

The first term is the maintenance cost of all items performed before the schedule horizon, the second term is the indirect cost due to loss of production, and the third term is the total of the shared costs due to initiation of maintenance (base costs).

Complexity

Let FMSOP be the problem of determining whether a feasible maintenance schedule exists. In this section, we argue that FMSOP is NP-complete by showing that (1) FMSOP is in NP, since it has a polynomial-time verification algorithm (Cormen et al. 2001), and (2) that there is a polynomial-time reduction from the bin packing problem (BPP) (Coffman, Garey, and Johnson 1997) to FMSOP. The objective of BPP is to pack items $i \in \{1, \ldots, n\}$ of given sizes a_i into as few bins (with fixed capacity V) as possible. The used capacity of a bin is computed as the sum of the weights of the items in the bin. The decision variant of BPP answers the question whether a packing for any given number of bins m exists.

Given a candidate solution to FMSOP (that is an assignment of dates to the maintenance items), we can verify the constraints on structure and timeliness by simply testing equations 4–7 for the given dates of the item and its predecessor and terminator. This verification can be done in $O(n)$ time. The m capacity constraints in equation 8 can easily be verified by accumulating the items allocated to the opportunities in time $O(np + m)$, where m is the number of opportunities and p is the number of phases. The availability constraint in equation 9 can also, in a similar fashion, be verified in $O(np + m)$ time. The procedure outlined above is polynomial, and therefore FMSOP is in NP.

We can translate a given BPP into a FMSOP by having m opportunities, each opportunity j having date $\delta_j = j$ and capacity $v_j = V$. Let the horizon $h = m + 1$. Each BPP item i is translated into a FMSOP replacement item i with $\perp$ as predecessor, 0 as release time, m as deadline, $t_i^{\min} = 0$ and $t_i^{\max} = h$. The duration $\Delta_{qi} = a_i$ if $q = i$ and 0 otherwise, that is, the duration (weight) of an item is always put in a unique phase in Δ_i. All items i have an artificial item $n + i$ as terminator, which in turn have release time 1, deadline $h + 1$, $t_i^{\min} = 0$, $t_i^{\max} = h + 1$, $\top$ as terminator and arbitrary duration. By definition, the tail items, being replacements, have to occur at $\top$, which is outside the schedule. Let the minimum availability $\alpha = 0$. The transformed problem corresponds directly to BPP, since (1) each BPP item is represented by an FMSOP replacement, (2) each BPP bin is represented by an FMSOP opportunity with unique date and equal capacity, and (3) the total down time of an opportunity is computed as the sum of the item durations at that opportunity, since all durations are in unique working phases, which corresponds directly to the sum of the weights of items in a bin in BPP. All other constructs of FMSOP are disabled and therefore do not constrain the solution, and therefore, BPP is a special case of FMSOP.

If we could find a solution to the transformed FMSOP using a polynomial-time algorithm, we could then use that algorithm to solve BPP (which is NP-complete [Garey and Johnson 1979]) in polynomial time. This property together with the previous conclusion that FMSOP is in NP implies that FMSOP is NP-complete.

Efficient polynomial-time approximations exist for the bin-packing problem (Coffman, Garey, and Johnson 1997). However, MSOP differs in objective from BPP and has complicating side constraints that are missing in BPP. For example, in MSOP, each opportunity (date) can have a different capacity, base cost, and down time cost. In BPP, a bin is defined only by its capacity, which is also uniform. Another difference is that items in MSOP can partially overlap within an opportunity due to the work time model used. These properties make bin-packing approaches inapplicable to MSOP. It is currently an open issue whether polynomial-time approximation schemes exist for MSOP.

A Tool for Maintenance Scheduling

The optimization software consists of two separate programs that communicate using files; PMOpt - GUI and MaintOpt. The schedule and related information are considered to be a *project* and are stored in a *project file*. A typical user would load a previously created project file directly after starting PMOpt. PMOpt-GUI makes it possible to edit the project file, and it immediately displays the effects of edits, such as costs and availability. Edits include changing lifetime estimates, adding or deleting components and items, and moving or copying items within and between components. Whenever the user requests an optimization of the current maintenance plan, PMOpt-GUI produces a rolled-out representation of the specification, which is passed on to the optimizer. As soon as MaintOpt finishes, the solution file is read back into PMOpt-GUI and shown to the user.

```
LDS-PROBE(node, k, d, b)
(1) if LEAF(node) then return node
(2) ⟨suc_0, …, suc_{b–1}⟩ ← SUCCESSORS(node)
(3) bt ← NIL
(4) for i = max(0, k – (b – 1)(d – 1)) to min(b – 1, k)
(5)     bt ← argmin_f(bt, LDS-PROBE(suc_i, k – i, d, b))
(6) return bt

LDS(node, d, b, k^max)
(1) res ← NIL
(2) for k ← 0 to k^max
(3)     res ← argmin_f(res, LDS-PROBE(node, k, d, b))
(4) return res
```

Algorithm 1. Limited Discrepancy Search for Nonbinary Optimization Problems with Objective f.

Optimization Algorithm

The requirement for the optimization algorithm is that it should be able to produce maintenance schedules within a limited time to be used interactively. Many industrial problems can be solved using tree search methods, especially if guiding heuristics are available. For example, best-first search methods such as A* search have been successful on many problems (Nilsson 1971; Hart, Nilsson, and Raphael 1972). However, A* search is reliant on the availability of a good admissible heuristic, and if no such heuristic is available, A* search will on some problems use up too much memory and time to be practically useful. Depth-first-based search methods avoid the memory issues of breadth-first search and A* but can easily get stuck in unproductive areas of the search tree when the heuristic fails. Limited discrepancy search (LDS) addresses this problem (Harvey and Ginsberg 1995; Korf 1996). The basic idea of LDS is to use depth-first search guided by a heuristic but to allow a specified number of "discrepant" choices disagreeing with the heuristic. The number of discrepant choices allowed in each path from root to leaf is the *discrepancy* parameter *k*.

The basic LDS procedure introduced by Harvey and Ginsberg (1995) only works on binary search trees, although the authors also discuss extensions to nonbinary problems. Algorithm 1 shows such an extension, also modified to continue searching for a best possible solution as measured by an objective function *f*, which is infinitely valued for the empty node NIL. In the original paper, it is discussed whether all discrepant choices emanating from a specific node should be treated equally, that is, counted as "using up" a discrepancy of one, or whether each further step away from the heuristic should be counted as using up one more discrepant choice. In algorithm 1, the latter view is taken. Figure 7 shows the unique paths explored for each choice of *k* in a tree with branching factor 3.

Korf (1996) modified LDS so that it only generates paths with exactly *k* discrepancies. This modification is done by keeping track of the remaining depth *d*, pruning branches for which $d \le k$. While modifying LDS to nonbinary trees, a similar improvement can be done if the maximum branching factor *b* is known. At most *d* discrepant choices can be made in a subtree of depth *d*, each choice using up at most $b - 1$ discrepancies. We can therefore prune choices *i* where $(b-1)(d-1) + i < k$, or in other words, start with choice number $\max(0, k - (b-1)(d-1))$. We assume that the function call SUCCESSORS(node) returns a list of feasible successor nodes in increasing order of heuristic value.

In the LDS procedure we use, maintenance items are assigned in order of increasing deadline, and the value-selection heuristic picks opportunities in increasing cost order, with a bias for late opportunities. In essence, the heuristic used is defined as

$$h\left(\mathbf{t}\left[i \mapsto \delta_j\right]\right) = f\left(\mathbf{t}\left[i \mapsto \delta_j\right]\right) + wc_i \frac{\mathbf{t}[p_i] + t_i^{\mathrm{dl}} - \delta_j}{t_i^{\mathrm{opt}}} \qquad (11)$$

where $\mathbf{t}[i \mapsto \delta_j]$ is the result of assigning item *i* to opportunity *j* given assignment **t**, *w* is a balancing weight, and t_i^{opt} is the original (optimal) maintenance interval for component *i*. t_i^{opt} is different from t_i^{dl} in that t_i^{dl} may be changed due to consumed lifetime of component *i*. Note that variable domains are pruned using interval propagation (Lhomme 1993). In our experiments, we found that a maximum discrepancy of $k^{\max} = 2$ gave overall good performance. We used $w = 1$ as the weight value to favor late assignments, and the default optimization time is set to 30 seconds, which is more than enough for the problem instances we have tried.

Development and Deployment

Manual planning is the norm in the gas turbine field, and before PMOPT and the lifetime prediction tool, SIT AB did not have any manual or automatic procedures for improving maintenance schedules. Instead, it used a standard schedule equal to 20,000 operating hours subject to a constant level of degradation at a standard pace for all components in the gas turbine. The initial effort at SIT AB focused on developing more accurate lifetime predictions, and the need for optimization of maintenance schedules first became apparent when the technology was driven to the point that the total

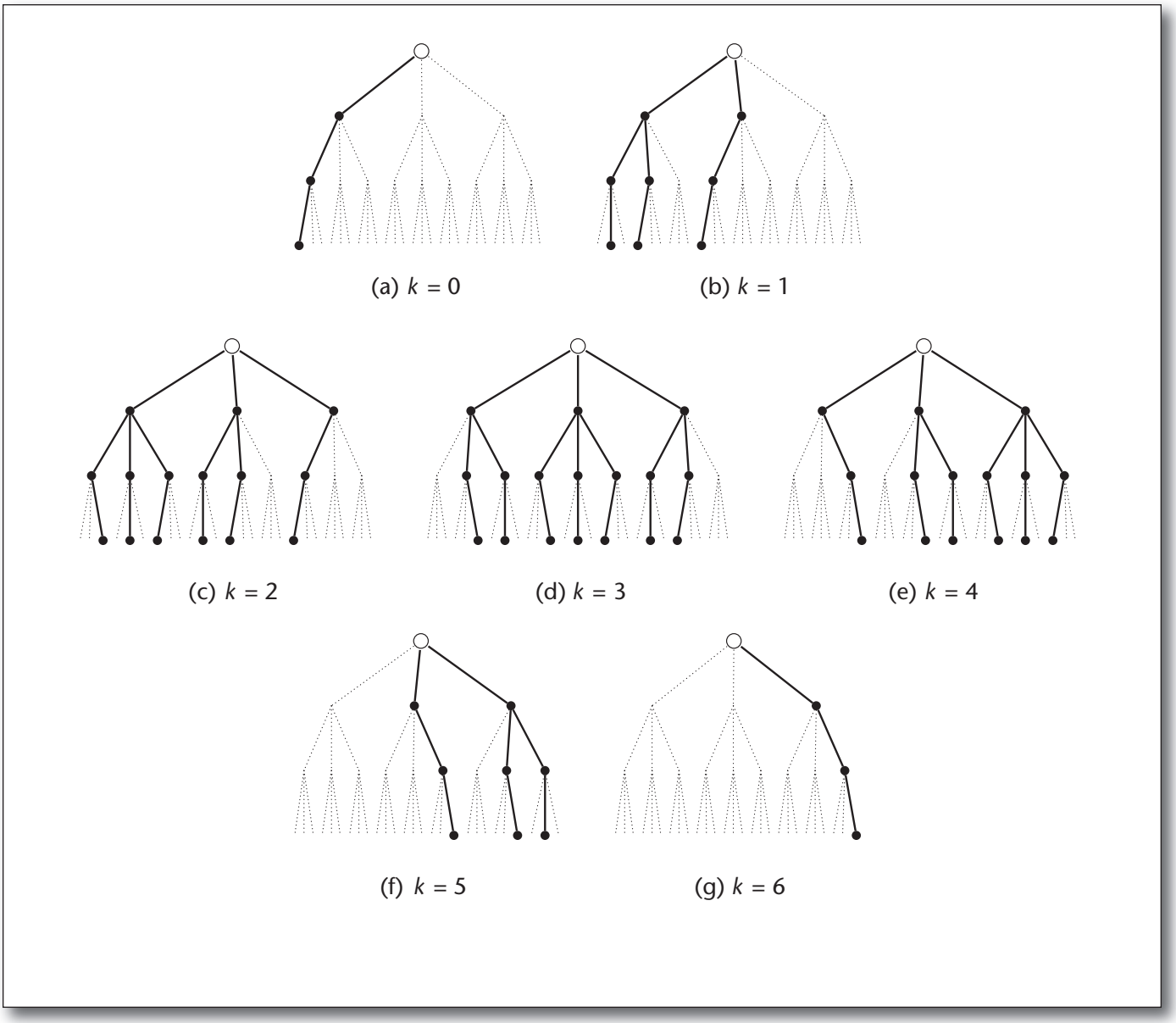

Figure 7. Unique Explored Edges for Different Discrepancy Values in a Tree with Branching Factor 3.

effect of component lifetime extensions could be evaluated.

We were first approached by SIT AB regarding maintenance-scheduling optimization during the summer of 2006 at an international conference related to condition monitoring. This first contact resulted in a sequence of meetings during the autumn at which we discussed and evaluated the feasibility of the project idea. At this time point, we had already developed the core maintenance-optimization engine (MAINTOPT). However, MAINTOPT was in its infancy, and we soon became aware that we had to extend it with side constraints and objective function terms that we had not previously considered. One example is the availability constraints and the focus on down time as a critical parameter, which was not implemented in MAINTOPT at that time. However, being able to demonstrate our ideas helped a lot during the initial meetings.

Before starting the PMOPT development process, a third-party commercial product for maintenance optimization was evaluated at SIT AB. One of the main problems with the third-party product was that it could not model important properties of the gas turbine planning problem, such as seasonal variations and usage profiles for different parameters like load, particle levels, and environmental factors. More importantly, the evaluated tool, and most other generic maintenance-optimization

tools, use costs based on statistics. In reality, it is common that the statistically optimal point of lowest cost is not practically feasible due to lack of data and the need for large safety margins. The consequences of some types of failures, including the possibility of loss of life, are under these circumstances too severe to be modeled and used as a basis for planning. In addition, for a complex machine such as a gas turbine, it is impractical to identify all possible failures, the corresponding statistical distributions, and all consequences and associated costs for each failure. Instead of having too many estimates, we decided that a safe deadline for each maintenance item was a more reasonable alternative.

Throughout the project, a total of five people from the Swedish Institute of Computer Science were directly involved. We had two main contact persons at SIT AB and decided to directly involve several site managers for evaluation. SIT AB was also heavily involved in the specification and development throughout the project, and this form of close collaboration allowed us to design accurate models of the application and was in our opinion a main factor behind the successful project outcome.

First Versions

In November 2006, we received a spreadsheet containing an early draft specification of the problem to be solved. The spreadsheet showed some ideas regarding calculations on down time and maintenance-activity packaging, and we decided to develop a prototype from the draft specification. The prototype was nothing more than a simple graphical front end to MaintOpt without any interaction. Nonetheless, it served the purpose of showing the feasibility of the project proposal well. After this prestudy and basic demonstration, we began discussions regarding the project economy and deliverables in early 2007. Soon after that the contracts were signed and development started. We finished the first release (version 0.9) in mid-April 2007. Due to time pressure, the first version was more of a prototype than a mature software product. With many test releases in between, version 1.0 was finally shipped in June 2007. The global CBM project was however not fully operational at that time, and version 1.0 was therefore mainly used for evaluation and as a platform for further development.

From experience with the first releases, we soon realized that changes in the optimization engine were rather straightforward to implement. However, software maintenance and extensions that primarily affected the management of the problem model was much more time-consuming. One of the biggest problems was to implement functionality to manage the maintenance schedule while verifying model consistency under the entire set of user actions possible. For example, changes in the maintenance schedule made after running the gas turbine for some time needed to be synchronized with the already laid-out maintenance schedule up to the current time point. Other areas that we chose to put more work into than estimated were the models of work time, application security, licensing, management of gas turbine maintenance projects, and user accounts and rights management.

Second Version

We made several changes to the basic design of PMOpt in the second phase of the project to simplify the software maintenance of the application and facilitate future extensions. Rewriting the core of the application from scratch was perhaps the largest change, but we also made some significant changes in the search algorithm, which took some effort. In the beginning, we implemented MaintOpt as a pure branch-and-bound algorithm based on A* search (Russell and Norvig 2003). However, after extensive experimentation with sample maintenance projects we realized that the A* search spent too much time and memory exploring high-level decisions in the search tree and failed to find feasible solutions quickly. Since responsiveness was one of the main criteria of PMOpt, we resorted to experimentation with heuristics and after a while added the LDS procedure as a first stage of the algorithm to quickly find feasible solutions. Lately, we have completely removed the A* search from MaintOpt after observing that the heuristics were not nearly powerful enough. Under these conditions, the A* search did not explore the search tree to its maximum depth within a reasonable time, and therefore, no feasible solutions were found. The ability to produce a reasonably good solution fast was more important than optimality, and we therefore chose LDS as the primary search algorithm.

With major changes to the GUI and improved heuristics, we released version 2.0 in March 2008, two months in advance of its deadline due to the much improved core design, which helped speed up the implementation of new features. Since then, we have made more improvements and shipped a new release in August the same year. The latest release (version 2.4) was shipped in November 2008.

Deployment at SIT AB

During the development of PMOpt, we became increasingly aware that a planning tool of this type is not easily deployed. First of all, key personnel had to be trained in condition-based maintenance and in how to use PMOpt. During the development of the first version, we continuously discussed suggestions and ideas for the usability enhancement of the software. Before deployment could begin,

we also had to develop suitable business models and finish evaluating current technology. In addition, data-acquisition routines, working processes, and suitable information flows needed to be established. Therefore, PMOpt was not deployed in operational use until early 2008 after the release of version 2.0.

The greatest benefit of using PMOpt seems to come from the ability to reoptimize maintenance after unexpected stops have occurred. The software has been used operatively for five months by employees at SIT AB to help plan maintenance for two gas turbine operators (end-user customers). PMOpt is fully operational in planning maintenance for the first operator and is used for validation and testing purposes of the global CBM strategy and the technology involved for the second. We test the planning software to gain feedback from practical experience using both PMOpt and the lifetime prediction algorithms. SIT AB plans to have four or five people working with PMOpt for 10–15 different operational contracts within a few years. Today, the tool is used interactively by experts in extended lifetime maintenance. When more people are starting to use PMOpt on a larger scale, changes and upgrades will likely be necessary as the user requirements on the application emerge.

Application Maintenance and Support

We performed software maintenance of PMOpt on demand when bug reports were filed, which happened mostly from our main contact people at SIT AB after new releases had been shipped. Naturally, we got most bug reports just after the delivery of version 1.0. Overall, larger improvements were mostly related to the GUI and the usability of the system. Current users have direct contact with us and are able to ask questions as well as request changes. During the development, we improved our understanding of the domain substantially, and we gradually implemented several new features in the problem models. SIT AB explicitly requested some changes, while we judged other changes to be necessary to make the code base easy to maintain. As an example, we changed the specification of the optimization model several times. Specifically, after a request from SIT we updated the work time model to more accurately capture the real duration, down time, and cost of a maintenance opportunity. In addition, the team also proposed other changes regarding the model of dependencies between maintenance items and the handling of obsolete items.

Estimated and Measured Benefits

In this section, we evaluate PMOpt using condition data and operational plans for a gas turbine used by a customer in the oil and gas industry. The cost of one engineer man-hour is roughly 120 USD, and the loss-of-production cost is estimated to 12,000 USD per hour. The results of the evaluation are based on the predicted cost savings produced by PMOpt. The turbine under consideration has 17 components with individual schedules. We use a standard maintenance schedule for the site as comparison. We modeled the critical components in the gas generator stage for which lifetime data was available (compressor turbine guide vanes and blades) and analyzed the components in a prognostics tool to determine suitable inspection intervals. The average increase in inspection period was 116 percent, and replacements for the critical components were not necessary, since their predicted lifetime became longer than the standard maintenance-contract length of 15 years. We described the scenario in more detail in a previous paper (Bohlin et al. 2009).

SIT AB has done a partial validation of the obtained lifetimes in that a reference gas turbine, having operated under the same conditions, was dismantled after a standard maintenance interval of 20,000 operating hours and thoroughly inspected. The analysis showed that the accumulated damage was significantly less than predicted using the standard EOH calculations. However, final validation has to wait until one or more reference gas turbines have been dismantled after a longer operational period.

We performed the evaluation on two scenarios. The first scenario had a three-week seasonal stop during the summer, when maintenance could be done without any negative effects on production. Such opportunities for maintenance are common in practice. In the second scenario, no such favorable opportunities existed. In both scenarios, a low base cost was associated with all maintenance stops, and high costs were associated with loss of production. The schedules resulting from running PMOpt were analyzed with regard to cost of production losses and maintenance costs. PMOpt was set to run for at most 30 seconds. In all examples tried, at most 10 hours of work per day were allowed before a resting period.

Results

Table 1 shows simulated results on availability, relative maintenance load, and down time for the eight different cases and a brand new gas turbine. The rows “EOH” and “Progn” correspond to planning maintenance at the last possible date, as specified using standard EOH calculations and the prognostics tool, respectively. This approach minimizes direct maintenance costs while ignoring other costs. The rows marked “EOH opt” and “Progn opt” correspond to optimizing maintenance using PMOpt.

	With Seasonal Stop			Without Seasonal Stop		
	Availability Percentage	Maintenance index	Down-time days	Availability Percentage	Maintenance index	Down-time days
EOH	**97.60**	**100**	**131**	**97.60**	**100**	**131**
EOH opt	99.99	109	0.42	98.15	120	101
Progn	98.20	61	98	98.20	61	98
Progn opt	100.0	62	0	98.81	75	65

Table 1. Predicted Results of Maintenance Optimization for a New Gas Turbine.

	With Seasonal Stop			Without Seasonal Stop		
	Availability Percentage	Maintenance index	Down-time days	Availability Percentage	Maintenance index	Down-time days
EOH	95.26	121	259	95.26	121	259
EOH opt	99.56	133	24.0	97.49	149	137
Progn	96.03	79	217	96.03	79	217
Progn opt	99.79	82	11.6	98.35	85	90

Table 2. Predicted Results of One Maintenance Optimization for a Gas Turbine with Randomly Chosen History for Its 17 Components.

	With Seasonal Stop			Without Seasonal Stop		
	Difference Percentage	Gap Percentage	Time	Difference Percentage	Gap Percentage	Time
New Turbine						
EOH opt	–6.1	0	40 minutes	–	∞	8 hours
Progn opt	–1.1	0	27 minutes	+93	75.6	8 hours
Used Turbine						
Progn	–23.6	1.79	8 hours	–	∞	8 hours
Progn opt	– 0.6	0.95	8 hours	–	∞	8 hours

Table 3. Comparison of Results between CPLEX 9.0 and PMOPT.

Negative numbers mean that CPLEX produced a lower cost than PMOPT.

Results are reported for availability due to preventive maintenance ("Avail"), maintenance costs ("Maint index"), and productive down time ("DT days"). Zero-cost down time corresponds to no production losses and is, therefore, in contrast to the theoretical model used in the optimization, not considered as down time in the results. Maintenance costs are expressed using an index. In it, 100 represents the cost of doing maintenance with the intervals from the standard schedule. In table 1, the results of using the standard schedule are typeset in boldface. Note that these results are suboptimal when low-cost opportunities are present.

As can be seen in table 1, better lifetime estimates had a significant impact on maintenance costs, availability, and down time. Adding the optimization of maintenance using PMOPT yields even better results and increases direct maintenance costs slightly. These results are to be expected, since production losses in this case are very costly and optimization is done with regard to both loss of production costs and direct maintenance costs. When zero-cost maintenance opportunities were available, productive down time was reduced to

zero or almost zero by planning maintenance for the available opportunities. Table 1 also shows that for the schedule with no advantageous opportunities, down time was reduced by more than 50 percent using PMOPT and prognostics.

Table 2 shows predicted results for the eight different cases and gas turbine already in use, simulated by setting the already-used lifetimes of the gas turbine components to a random number drawn from a uniform distribution between zero and the maintenance interval for the component. As expected, table 2 shows higher costs and lower availability than table 1 due to a more spread out maintenance need. Using a prognostics tool and PMOPT in this scenario also yielded significant results. In the experiments, down time was reduced by 65 percent for a schedule with no advantageous opportunities, compared to the current state of practice. In the case where seasonal opportunities were present, down time was reduced by 95 percent.

Comparison with Integer Programming

To investigate how far away from the optimum the results from PMOPT were, we formulated MSOP as an integer linear-programming problem. We used ILOG CPLEX 9.0 on a mainframe computer with a 2.2 GHz dual core AMD Opteron CPU and 8 GB of RAM to solve the problem. The total run time for each case was limited to 8 hours. Although running an algorithm for such a long time is not suitable for our needs, the comparison still gives us valuable insight in where PMOPT can be improved. In contrast, PMOPT was run on a laptop with a 1.6 GHz Intel CPU for 30 seconds in each case.

Results for the eight different cases in the previous sections are compared in table 3. *Diff* gives the relative difference between the best found cost for PMOPT and CPLEX, with negative values indicating that CPLEX found a better solution than PMOPT. The *Gap* column gives the relative optimality gap (distance to the relaxed optimum) as returned by CPLEX, with higher values indicating that the gap is larger. The gap is infinite if no feasible solution was found within 8 hours. The *Time* column reports CPU runtime for proving optimality, with a cutoff at 8 hours.

For the two cases with a new turbine and seasonal stops, CPLEX was able to find the exact optimal solution (indicated by a gap value of 0). For the two cases with a used turbine and seasonal stops, CPLEX had found better solutions than PMOPT when 8 hours had passed, with a quite small optimality gap. While CPLEX produces slightly better results for cases with lifetimes from the prognostics tool (*Progn*), the instances with standard EOH lifetimes appear to benefit more significantly. It is notable that CPLEX reports a result that is more than 23 percent better than PMOPT in the case with EOH lifetimes and seasonal stops. However, when there are no seasonal stops, CPLEX cannot find a solution even close to the result from PMOPT within 8 hours.

Conclusions and Future Work

We described the development and deployment of an opportunity-based maintenance-planning tool, PMOPT, specifically designed to fit the purpose of improving the maintenance schedules for gas turbines from SIT AB. The goal was to reduce both direct maintenance costs and production losses. Thanks to close collaboration with key personnel at SIT AB, we gained important insights into industrial maintenance planning, which allowed us to design and implement the maintenance-planning tool. We believe that the working process we used has contributed greatly to the successful deployment of PMOPTS.

We formally described and characterized the scheduling problem as NP-complete, and discussed a heuristic algorithm for solving it. Our experiments using real-world data for lifetime predictions showed a potential for significantly reduced down time (with up to 65 percent) and costs. Experiments with integer programming gave even greater gains but at the cost of much longer solution times. Expected effects in practical use include large availability improvements and preventive-maintenance reductions of up to 12 percent. Future plans include fleet-level planning and labor-resource optimization and scheduling, inclusion of stochastic deterioration models and expected costs due to corrective maintenance, and application to other domains. We are also considering investigating solution sensitivity with regard to parameter changes.

References

Bäckert, W., and Rippin, D. W. T. 1985. The Determination of Maintenance Strategies for Plants Subject to Breakdown. *Computers and Chemical Engineering* 9(2): 113–126.

Bohlin, M.; Wärja, M.; Holst, A.; Slottner, P.; and Doganay, K. 2009. Optimization of Condition-Based Maintenance for Industrial Gas Turbines: Requirements and Results. Paper no. GT2009-59935 presented at the ASME Gas Turbine Technical Congress and Exposition, June 8–12, Orlando, FL.

Coffman, E. G.; Garey, M. R.; and Johnson, D. S. 1997. Approximation Algorithms for Bin Packing: A Survey. Chapter 2 in *Approximation Algorithms for NP-Hard Problems,* ed. D. Hochbaum, 46–93 Boston: PWS Publishing Company.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. Introduction to Algorithms, second edition. Cambridge, MA: The MIT Press.

Dekker, R. 1996. Applications of Maintenance Optimiza-

tion Models: A Review and Analysis. *Reliability Engineering and System Safety* 51(3): 229–240.

Dekker, R., and Scarf, P. 1998. On the Impact of Optimisation Models in Maintenance Decision Making: The State of the Art. *Reliability Engineering and System Safety* 60(9): 111–119.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1972. Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." *SIGART Bulletin* (37): 28–29.

Harvey, W. D., and Ginsberg, M. L. 1995. Limited Discrepancy Search. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence,* 607–615. San Francisco: Morgan Kaufmann Publishers.

Korf, R. E. 1996. Improved Limited Discrepancy Search. In *Proceedings of the Eighth Innovative Applications of Artificial Intelligence Conference,* 286–291. Menlo Park, CA: AAAI Press.

Lhomme, O. 1993. Consistency Techniques for Numeric CSPs. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence,* 232–238. San Francisco: Morgan Kaufmann Publishers.

Marseguerra, M.; Zio, E.; and Podofillini, L. 2002. Condition-Based Maintenance Optimization by Means of Genetic Algorithms and Monte Carlo Simulation. *Reliability Engineering and System Safety* 77(2): 151–165.

Nilsson, N. J. 1971. *Problem-Solving Methods in Artificial Intelligence.* New York: McGraw-Hill.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach,* second edition. Engelwood Cliffs, NJ: Prentice Hall.

Tan, J. S., and Kramer, M. A. 1997. A General Framework for Preventive Maintenance Optimization in Chemical Process Operations. *Computers and Chemical Engineering* 21(12): 1451–1469.

Van Dijkhuizen, G., and van Harten, A. 1997. Optimal Clustering of Frequency-Constrained Maintenance Jobs with Shared Set-Ups. *European Journal of Operations Research* 99(3): 552–564.

Wildeman, R. E.; Dekker, R.; and Smit, A. C. J. M. 1997. A Dynamic Policy for Grouping Maintenance Activities. *European Journal of Operations Research* 99(3): 530–551.

Yamayee, Z.; Sidenblad, K.; and Yoshimura, M. 1983. A Computationally Efficient Optimal Maintenance Scheduling Method. *IEEE Transactions on Power Apparatus and Systems* AS-102(2): 330–338.

Markus Bohlin is a researcher at the Swedish Institute of Computer Science. He has more than 10 years of experience with industrially-motivated research. In 2009, he successfully defended his Ph.D. thesis on combinatorial optimization in industrial computer systems. His research is focused on real-world combinatorial optimization with applications in maintenance, software engineering and logistics..

Kivanc Doganay is a researcher at the Swedish Institute of Computer Science. He holds a M.Sc. in computer science from Mälardalen University, and was a full-time developer exclusively for the late releases of PMOpt. Currently, he works with maintenance optimization in the railway sector and industrial data analysis.

Per Kreuger is a senior researcher at the Swedish Institute of Computer Science (SICS) specializing in applied combinatorial optimization. He received his Ph.D. at the Computing Science Department of Gothenburg University in 1995 and has since then worked mainly with practical combinatorial problems using constraint programming and mathematical programming techniques within transportation, networking, and related areas. Recent achievements related to AI includes cochairing and editing the proceedings of the Scandinavian AI conference 2008.

Rebecca Steinert was employed in 2006 as a researcher by the Swedish Institute of Computer Science (Industrial Applications and Methods lab) in Kista, Sweden. In 2008, she received a M.Sc. in computer science from KTH in Stockholm, Sweden and is now a Ph.D. student at the School of Computer Science and Communications, KTH. Her research is focused on statistical methods for fault-handling in industrial applications and networked systems.

Mathias Wärja has been a specialist in CBM for rotating equipment at Siemens Industrial Turbomachinery AB since 2004. In 2005, he received his Ph.D. in industrial information and control systems from KTH, Stockholm.