

A Tool for Gas Turbine Maintenance Scheduling*

Markus Bohlin and Kivanc Doganay and Per Kreuger and Rebecca Steinert

Swedish Institute of Computer Science
SE-164 29 Kista, Sweden

Mathias Wärja

Siemens Industrial Turbomachinery AB
SE-612 83 Finspong, Sweden

Abstract

We describe the implementation and deployment of a software decision support tool for the maintenance planning of gas turbines. The tool is used to plan the maintenance for turbines manufactured and maintained by Siemens Industrial Turbomachinery AB (SIT AB) with the goal to reduce the direct maintenance costs and the often very costly production losses during maintenance downtime. The optimization problem is formally defined, and we argue that feasibility in it 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 mixed integer linear programming, and discuss the deployment of the application. The experimental results indicate that downtime reductions up to 65% can be achieved, compared to traditional preventive maintenance. In addition, using our tool is expected to improve availability with up to 1% and reduce the number of planned maintenance days with 12%. Compared to a mixed integer programming approach, our algorithm not optimal, but is orders of magnitude faster and produces results which 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.

Introduction

Preventive maintenance can reduce breakdowns and costs associated with them, but is also costly when done frequently. That is why considerable effort (e.g. [Dekker, 1996; Tan and Kramer, 1997]) has previously been spent on optimizing maintenance so that the expected total cost due to failures and preventive maintenance is minimized. Most preventive maintenance approaches use fixed schedules, which are optimized for minimum cost in advance. However, there are many situations in which maintenance re-planning is in practice necessary to being able to continue operation and to lower costs. For example, unexpected breakdowns force the production unit to stop for emergency repair, and it would be

unwise not to consider performing other maintenance tasks at the same time. Other examples include production stops for other reasons than maintenance, which provides valuable opportunities for maintenance. The introduction of condition monitoring has also lead to the replacement of preventive maintenance with condition-based corrective maintenance, which is by nature less predictable than a fixed preventive maintenance plan.

In this paper, we present the ideas behind a tool, PMOPT (Preventive Maintenance Optimizer), for gas turbine maintenance planning. PMOPT was developed 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 downtime costs. A typical gas turbine application is offshore oil platforms, where time spent without power 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 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 becoming more and more common. Although lifetime predictions can sometimes be performed with high precision, maintenance time points will still vary depending on the conditions on site, and the actual time points will therefore also diverge from their original estimates over time.

The approach presented in this paper aims at providing a tool which can quickly optimize maintenance when unplanned events make the current maintenance schedule unsuitable. We use a rolled-out representation of a deterministic future maintenance schedule, which makes it possible to take into account positive effects of co-allocation, maintenance opportunities, overall availability, horizon effects and costs due to both maintenance and loss-of-production.

Proper risk analysis and deterioration model identification can in many practical cases be difficult to perform from scratch. As a consequence, maintenance intervals are often based on analytical models and "best practice". Instead of using failure rate distributions to make tradeoffs between costs for breakdown and preventive maintenance, we therefore assume a safe deadline for maintenance activi-

*This work was supported by VINNOVA, SIT AB and SICS.
Copyright © 2009, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

ties, which simplifies the problem and makes it easy to adapt already existing maintenance plans for use in PMOPT.

The contributions of this paper 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 mixed integer linear programming, and 6) discuss the implementation and deployment of PMOPT.

Related Work

Maintenance optimization is certainly not a new topic. An excellent overview of the many applications considered can be found in [Dekker, 1996]. In [Yamayee, Sidenblad, and Yoshimura, 1983] dynamic programming is used to optimize maintenance planning with respect to acceptable equipment reliability, demand of generating units and maintenance cost. However, overall availability as a fraction of the total time is not considered, and the crew and resource model used does not consider downtime due to day and week rest.

Other approaches to maintenance optimization are based on Monte Carlo simulations combined with genetic algorithms [Marseguerra, Zio, and Podofillini, 2002]. In a related approach described in [Tan and Kramer, 1997], pre-planned maintenance opportunities are taken into account similarly to our own method. However, their approach is non-deterministic in contrast to our optimization method.

In [Wildeman, Dekker, and Smit, 1997], maintenance planning is done for a more restrictive system where certain properties of the cost function must hold, and where potential gain of co-allocating maintenance is constant for all activities. In our model, cost is a function of component costs and indirect costs, resulting from unavailability of the gas turbine due to maintenance. This makes our model more expressive, and thus unsolvable using the polynomial solution approach in [Wildeman, Dekker, and Smit, 1997].

Background

The common practice of gas turbine maintenance planning today is to base the schedules on Equivalent Operating Hours (EOH) and cycles (i.e., the number of turbine restarts). The number of operating hours is modified 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 substantial built-in safety margins to accommodate for variables not explicitly modeled.

In order 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 has been developed by SIT AB. A lifetime prediction tool, producing deterministic lifetime estimates, has also been developed. The lifetime estimates produced by the tool include relevant safety margins.

Therefore, changes in lifetime should not affect risk levels negatively as long as the gas turbine is serviced within its predicted lifetime. In fact, risk levels can in many cases be 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).

Improved Analytical Lifetime Predictions

Gas turbine component lifetime is to a great extent determined by operation temperatures. However, it is also determined by the extreme rotational load and pressures that some parts are exposed to. The gas turbine cycle is also highly sensitive to ambient conditions (mainly inlet air pressure and temperature). The following procedure is employed to calculate the component lifetime for a specific situation.

1. First, the overall energy balance of the gas turbine is calculated using heat balance evaluations based on measurements of pressures, temperatures and rotational speeds at various locations in the gas turbine.
2. Based upon the energy balance input, we then calculate the expected mass flow, temperature and pressure at locations where sensors cannot easily be placed or hamper performance (such as within the hot gas pass, inside the combustion chamber, and inside the rotors). The calculations are performed using standard methods from combustion kinetics, aerodynamics, flow distribution and cooling codes.
3. Finally, we compute the mechanical response to the thermal, aerodynamical and mechanical loads for component sets that interact mechanically with each other.

The results of step 2 and 3 are then used to compute an expected lifetime. Some of the involved calculations are carried out using the finite element method (see for example [Zienkiewicz, Taylor, and Zhu, 2005]). However, the applied fluid and solid models are specifically adapted to gas turbine conditions and materials.

Calculation time for the process outlined above can range from weeks to months per iteration. Therefore, a pre-computed approximation is used for real-life prediction. The approximation is refined by manual correction using experience from service and risk assessments to be accurate enough and to provide sufficient safety margins.

Problem Description

In this section, we give first an informal description of the scheduling problem that PMOPT is aimed at solving. We then define the duration model adopted in this paper, which includes calculations of total work and stop time for an maintenance stop. This is followed by a more rigorous definition of the scheduling problem we want to solve. The section ends with an argument for why feasibility in the maintenance scheduling problem is NP-complete, and why new solution methods are needed to solve it.

We can informally describe the Maintenance Scheduling with Opportunities Problem (MSOP) as the problem of allocating maintenance *items* to dates for k independent components in a single unit and for a time period of h weeks,

so that constraints on timeliness, work time capacity and total availability are satisfied. The allocation should minimize direct and indirect maintenance costs, including spare parts, labor, and value of production lost due to maintenance.

Each component has a cyclical schedule of arbitrary length, consisting of *inspections* and *replacements*. The date of a replacement depends only on the previous replacement, while inspections depend on the previous item regardless of type. We assume that the obtained lifetime estimates used as input to the optimizer are safe in the sense that if maintenance deadlines are met, risk levels are negligible. Also, we assume that the given component schedules are followed and that deviations are taken into account by updating the schedule data. The problem is therefore deterministic in nature.

Duration Models

To estimate work time at a maintenance stop, each maintenance item has a *duration specification* $\Delta_i = \langle \Delta_{1i}, \Delta_{2i}, \dots, \Delta_{Bi} \rangle$ divided into non-negative *work phases* Δ_{bi} , where at least one phase has to be non-zero. The set of work phases are denoted by $\mathcal{B}$. All items with 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 have to be done in an orderly fashion, and therefore have to be executed serially. The total work time u_j of a stop can thus be computed as the sum of the maximum work time in each block.

As an example, consider the two duration specifications $\langle 3, 1, 0, 5 \rangle$ and $\langle 4, 0, 2, 3 \rangle$ allocated to the same stop. The working time for the different phases then becomes $\langle 4, 1, 2, 5 \rangle$, and the total work time at the stop is 12.

Given the total work time at a stop, we can now compute the *downtime*. We assume that a working day consists of A hours, and that all calendar weeks (consisting of 6 working days) are alike. The downtime of non-empty stops is computed by adding night-rest time for each day when all work was not finished, and week-rest time for each week when all work was not finished, using the following function.

$$\mathcal{D}(W) = W + (24 - A) \left\lceil \frac{W}{A} - 1 \right\rceil + 24 \left\lceil \frac{W}{6A} - 1 \right\rceil \quad (1)$$

For empty stops, $\mathcal{D}(W) = 0$. In the rest of this paper, we assume that $A = 10$.

Optimization Model

We assume that n maintenance items denoted by $i \in \mathcal{I}$ have been rolled out to cover weeks 1 to h (the *horizon* of the problem). The decision variable $\mathbf{t}[i]$ represent 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 $\perp$ at date 0. The possible allocation dates within the schedule are modeled by a finite set $\mathcal{O}$ of *opportunities* j with dates δ_j and work time capacity v_j .

Timeliness constraints are expressed as follows. Each item i has a *release time* o_i and a *deadline* d_i relative to i 's *predecessor* p_i . Each item also has an optional *earliest* and *latest date* of execution ($t_i^{\min}$ and $t_i^{\max}$). We assume

that each replacement for a component starts a new sequence of inspections, which makes items from previous sequences redundant. We call rolled-out items that do not have to be executed *obsolete* items.

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 $\text{obs}(i)$, with the meaning that activity i is made obsolete by its terminator s_i , as follows.

$$\text{obs}(i) \equiv \mathbf{t}[i] = \mathbf{t}[s_i] \quad (2)$$

Replacements always have $\top$ as their terminator, which implies that they are only made obsolete by being moved over the problem horizon h . Figure 1 illustrates relative timeliness constraints (release times and deadlines) between pairs of tasks as well as predecessor and terminator relationships in a fictional schedule.

The first items in the schedule for each component is called the set of *head* items, and is denoted $\mathcal{E}$. All head items are assumed to 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. By forcing all tail items to be obsolete, the normal deadline constraints ensure that end gaps are smaller than required for all feasible solutions. The concepts are illustrated in Figure 1.

Each item also has an *item cost* c_i consisting of work and material cost. The value of production per hour at an opportunity j is denoted l_j . In addition, we use a fixed *base cost* (b_j) for opening up opportunity j . The base cost is associated with setup costs for shutting down and restarting the gas turbine, travel expenses, and other costs that cannot be modeled using material, work or downtime costs.

Minimum availability is specified by the user via the parameter α (where $0 \leq \alpha \leq 1$). The total availability is defined as the productive time not spent on preventive maintenance divided by the total available productive time. The constraints in the problem can now be stated.

- Each item i should be allocated to a date $\mathbf{t}[i]$ that is less than or equal to its deadline.

$$\forall i \in \mathcal{I} : \mathbf{t}[i] \leq \mathbf{t}[p_i] + d_i \quad (3)$$

- Each item has to respect its absolute allocation interval.

$$\forall i \in \mathcal{I} : t_i^{\min} \leq \mathbf{t}[i] \leq t_i^{\max} \quad (4)$$

- Each tail item has to be obsolete.

$$\forall i \in \mathcal{L} : \text{obs}(i) \quad (5)$$

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

$$\forall i \in \mathcal{I} \setminus \mathcal{L} : \text{obs}(i) \vee \mathbf{t}[i] \geq \mathbf{t}[p_i] + o_i \quad (6)$$

- For each opportunity j , the total work time u_j allocated to j must be lower than the capacity of j .

$$\forall j \in \mathcal{O} : u_j = \sum_{b \in \mathcal{B}} \max_{i \in \mathcal{I}} \Delta_{bi} \quad (7)$$

$$\forall j \in \mathcal{O} : u_j \leq v_j$$

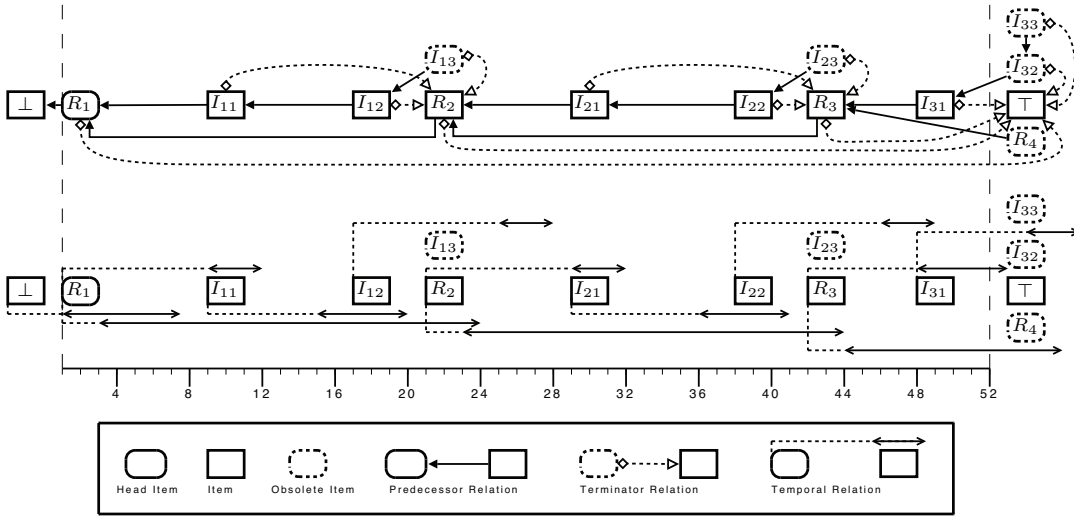


Figure 1: Dependencies (top) and relative timeliness constraints (bottom) between different maintenance activities of a component.

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

$$\frac{1}{7 \cdot 24} \sum_{j \in \mathcal{O}} \mathcal{D}(u_j) \leq h(1 - a) \quad (8)$$

$\exists i \in \mathcal{I} : \mathbf{t}[i] = \delta_j$

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

$$f(\mathbf{t}) = \sum_{i \in \mathcal{I}} c_i + \sum_{j \in \mathcal{O}} l_j \mathcal{D}(u_j) + \sum_{j \in \mathcal{O}} b_j \quad (9)$$

$\neg \text{obs}(i)$ $\exists i \in \mathcal{I} : \mathbf{t}[i] = \delta_j$

The first term is the maintenance cost of all items within the horizon, the second term is the indirect costs for the opportunities, and the third term is the base costs.

Complexity

Feasibility in MSOP (or FMSOP for short), that is, the question whether any feasible solution to MSOP exists, is NP-complete. We argue in this section that 1) FMSOP is in NP by outlining a polynomial-time verification algorithm ([Cormen et al., 2001]), and 2) that there is a polynomial-time reduction from the bin packing problem (BPP; see for example [E. G. Coffman, Garey, and Johnson, 1997]), to FMSOP.

The objective of BPP is to pack items $i \in \{1, \dots, n\}$ 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 B exists.

1. Given a candidate solution C to FMSOP (i.e. an assignment of dates to the maintenance items in C), we can verify the constraints on structure and timeliness by simply testing Equations (3), (4), (5) and (6) for the given dates of the item and its predecessor and terminator. This can be done in linear time to the number of

items. The capacity constraints in Eqn. (7) can easily be verified by investigating the items allocated to that opportunity in time $O(nm)$. The availability constraint in Eqn. (8) can be verified in a similar way as for the capacity constraints. This together with computations of the downtime function in Eqn. (1) can be done in time $O(nm)$. The procedure outlined above is clearly polynomial, and therefore FMSOP is in NP.

2. We can translate a given BPP into a FMSOP by having B opportunities, each opportunity j (where $1 \leq j \leq B$) having date $\delta_j = j$ and capacity $v_j = V$. Let the horizon $h = B + 1$. Each BPP item i is translated into a FMSOP replacement item i with $\perp$ as predecessor, 0 as release time, B as deadline, $t_i^{\min} = 0$ and $t_i^{\max} = h$. The duration $\Delta_{bi} = a_i$ if $b = i$ and 0 otherwise, i.e., 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 $h + 1$, $t_{n+i}^{\min} = 0$, $t_{n+i}^{\max} = h + 1$, $\top$ 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 requirement $\alpha = 0.0$.

The transformed problem corresponds directly to BPP, since 1) each BPP item is represented by a FMSOP replacement, 2) each BPP bin is represented by a FMSOP opportunity with unique date and equal capacity, and 3) the total duration 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, see [Garey

and Johnson, 1979]) in polynomial time. This, together with the previous conclusion that FMSOP is in NP, shows that FMSOP is NP-complete.

Efficient polynomial-time approximations exist for the bin-packing problem; see for example [E. G. 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 downtime 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. This makes 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 architecture is shown in Figure 2. MAINTOPT is written in C++, and PMOPT-GUI is written in the C++/CLI programming language using the .NET platform. PMOPT does not require any special installation procedure; it simply runs as a stand-alone application on any computer where the .NET framework is installed.

The schedule and related information are considered to be a *project*, and is 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 immediately displays the effects of edits, such as costs and availability. Edits include changing lifetime estimates, adding/deleting components and activities and moving/copying activities within and between components.

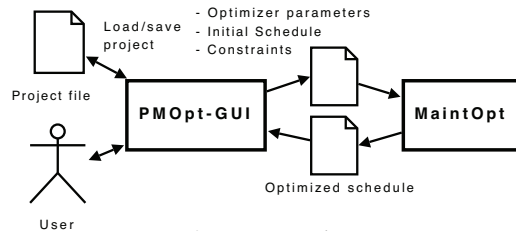


Figure 2: System architecture.

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. Time is translated into integer values, so that MAINTOPT does not need to be aware of the time scale. As soon as MAINTOPT finishes, the solution file is read back into PMOPT-GUI and shown to the user.

MAINTOPT and the Optimization Algorithm

The optimization algorithm should be able to produce maintenance schedules within a limited time in order to be used interactively. The optimization algorithm in MAINTOPT is based on Limited Discrepancy Search (LDS) [William D. Harvey, 1995]. 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. Only consistent assignments are considered; variable domains are pruned using interval propagation [Lhomme, 1993]. In our experiments, we have found that iteratively increasing the LDS width k from 0 to 2, resolving the problem for each k , gives overall good performance. The default optimization time is set to 30 seconds, which is more than enough for normal instances.

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. A standard schedule was used, which is equal to 20 000 operating hours and assumes a constant level of degradation at a standard pace for all components of the gas turbine. When the lifetime prediction techniques outlined in this paper had been developed, the need for maintenance planning in order to take advantage of possible lifetime extensions soon became apparent.

The Swedish Institute of Computer Science (SICS) was 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 with the purpose of evaluating the feasibility of the project idea. At this time point, the core maintenance optimization engine (MAINTOPT) had already been developed for use in a different project in the railway domain. However, MAINTOPT was in its infancy, and it became obvious during our collaboration with SIT AB that we had to extend it with side constraints and objective function terms previously not considered. One example is the availability constraints and the focus on downtime as a critical parameter, which was not present in MAINTOPT at that time. However, being able to demonstrate the early version of the planning software together with demonstrator applications from previous projects helped a lot during these initial meetings.

Before starting the PMOPT development process, a commercial product for maintenance optimization had been evaluated at SIT AB. One of the main problems with the product was that it was not able to 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, generic tools often use costs based on statistics. In reality, it is not uncommon that one prefers not to use the statistically optimal point of lowest cost due to the need for safety margins. The consequences of some types of failures are also too severe to be estimated using statistical approaches. In addition, for a complex machine such as a gas turbine, it can be 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, it was decided that a safe deadline for each maintenance activity was a better alternative.

SIT AB were heavily involved in the specification and development throughout the project, and this was a main factor behind the successful outcome. Without close collaboration

with SIT AB, many details regarding the application area would have been missed due to lack of knowledge in that area. It would also have been difficult to motivate necessary design changes without support from our main contacts.

Throughout the project, five people from SICS were directly involved. We had two main contact persons at SIT AB, and several site managers were directly involved.

First Versions

In November 2006, SICS received a spreadsheet containing an early draft specification of the problem to be solved. The spreadsheet showed some ideas regarding calculations on downtime and maintenance activity packaging, and it was decided that a prototype should be developed 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 pre-study 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 ended up being more of a prototype than mature software. With many test releases in between, version 1.0 was finally shipped in June 2007.

From experience with the first releases, we soon realized that changes in the optimization engine were rather straightforward to implement. However, maintenance and extensions that primarily affected the management of the problem model proved to be much more time consuming. One of the biggest problems was to keep the model consistent and to handle the entire set of user actions and model parameters. 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. We soon realized that we had severely underestimated the work involved in managing the maintenance schedule. Other areas that needed more attention than expected were 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 maintenance of the application and facilitate future extensions. Rewriting the core of the application from scratch was perhaps the largest one, but significant changes were also made in the search algorithm. In the beginning, MAINTOPT was a pure branch-and-bound algorithm based on A* search [Russell and Norvig, 2003]. However, after extensive experimentation with sample maintenance projects it became apparent that A* search, being based on breadth-first search, was spending too much time exploring high-level decisions in the search tree, and failed in finding feasible solutions quickly. Since responsiveness of the application was one of the main criteria of PMOPT, we resorted to experimentation with heuristics, and after a while, the LDS procedure

was added as a first stage of the algorithm. Lately, the second stage A* search has been completely removed from MAINTOPT, since it does not really help in solving typical problem instances. In our experience, system responsiveness and producing a reasonably good solution fast was more important than producing the absolute optimum. Tuning heuristics turned out to be an important task, as the standard A* and LDS algorithms were of limited value without guidance using the specific problem characteristics.

With major changes to the GUI and improved heuristics, a second major version (version 2.0) was released in March 2008. This version was delivered 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, more improvements have been made, with 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, it became increasingly clear that a planning tool of this type is not easily deployed. First of all, key personnel need to be educated in the theories behind condition-based maintenance planning, and in how an automated tool can help in adjusting a schedule to customer-specific conditions. In addition, it was necessary to gain adequate insight into maintenance planning practices in order to increase the usability of the PMOPT tool. During the development of the first version, suggestions and ideas for the usability enhancement of the software were continuously discussed. Before deployment could begin, suitable business models also had to be developed, evaluations of current technology needed to be completed, personnel had to be trained in using the tool, and 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.

Currently, PMOPT is used by two people, mainly for planning of maintenance after deviations have occurred. PMOPT has been running operatively for verification/validation of the global CBM strategy for five months. It is used within two maintenance contracts; in the first, PMOPT is fully operational, while it is used for validation and testing purposes in the second one. Testing is done mainly for gaining feedback from practical experience, monitoring of environmental variables (e.g. temperatures), and time increments. In a couple of years, four or five people working within maintenance planning are expected to use the tools for 10–15 different operational contracts.

Application Maintenance and support

Maintenance of PMOPT were performed by SICS 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, most bugs were reported 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, our understanding of the domain improved and matured, and several improvements of the problem models were gradually implemented. Some changes were explicitly requested by SIT AB, while others were necessary to make the code base easy to maintain. As an example, the specification of the optimization model was changed several times, and the work time model, by request from SIT AB, was updated to more accurately capture the real duration, downtime and cost of a maintenance opportunity. Another change was proposed by the development team regarding the model of dependencies between maintenance items and the handling of obsolete items. The first model proposed was too simplistic in that there was no difference between inspections and replacements.

Estimated and Measured Benefits

In this section, PMOPT is evaluated on a real world scenario in the oil and gas business. The turbine under consideration has 17 components with individual schedules. A standard maintenance schedule for the site was used as a comparison. The critical components in the gas generator stage for which lifetime data was available (compressor turbine guide vanes and blades) were modeled and evaluated in a prognostics tool to determine suitable inspection intervals. The average increase in inspection time was 116 %, and replacements for the critical components were not necessary, since their predicted lifetime were much longer than the standard maintenance contract length of 15 years. The scenario is described in more detail in [Bohlin et al., 2009].

A partial validation of the obtained lifetimes has been done 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.

The evaluation was done on two scenarios. The first scenario had a three week seasonal stop during the summer, where 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 1) cost of production losses and 2) maintenance costs. PMOPT was set to run for at most 30 seconds.

Results

Table 1 shows results for a simulated 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		
	Avail %	Maint index	DT days	Avail %	Maint index	DT 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: Results of maintenance optimization for a new gas turbine.

	With seasonal stop			Without seasonal stop		
	Avail %	Maint index	DT days	Avail %	Maint index	DT 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: Results of maintenance optimization for a gas turbine with randomly chosen history.

Results are reported in the form of availability (“Avail”), maintenance costs (“Maint index”) and productive days spent doing maintenance (“DT days”). Maintenance costs are expressed using an index. In it, 100 represents the cost of doing maintenance according to the maintenance intervals computed using the standard schedule. In Tab. 1, this corresponds to the row typeset in boldface.

As can be seen in Tab. 1, better lifetime estimates had a significant result on maintenance costs, availability and downtime. Adding the optimization of maintenance using PMOPT yields even better results, and increases direct maintenance costs slightly. This is natural, 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. Table 1 also shows that for a schedule with no advantageous opportunities, downtime can be reduced by more than 50 % using PMOPT and a prognostics tool.

Used Gas Turbine Table 2 shows the same scenario but for a simulated gas turbine already in use. The scenario is simulated by setting the already-used lifetimes of the gas turbine components to a random number drawn from a uniform distribution between 0 and the maintenance interval for the component. As expected, Tab. 2 shows higher costs and lower availability than Tab. 1 due to a more spread out maintenance need. Using a prognostics tool and PMOPT in this scenario also yields significant results. Downtime can be reduced by 65 % for a schedule with no advantageous opportunities, compared to the current state of practice. In the case where seasonal opportunities are present, downtime can be reduced from 259 to 11.6 days.

Comparison with CPLEX

In order to investigate how far away from the optimum the results from PMOPT are, we formulated MSOP as a mixed integer linear programming problem. We used ILOG CPLEX 9.0 on a mainframe computer with a 2.2 GHz Dual

	With seasonal stop			Without seasonal stop		
	Diff. %	Gap %	Time	Diff. %	Gap %	Time
<i>New turbine</i>						
EOH opt	-6.1	0	40m	-	∞	8h
Progn opt	-1.1	0	27m	+93	75.6	8h
<i>Used turbine</i>						
EOH opt	-23.6	1.79	8h	-	∞	8h
Progn opt	-0.6	0.95	8h	-	∞	8h

Table 3: Comparison of results between CPLEX 9.0 and PMOPT.

Core AMD Opteron CPU and 8 GB of RAM to solve the problem. The total runtime 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 (described previously) are compared in Tab. 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 report 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 appears to benefit more significantly. It is notable that CPLEX reports a result which is more than 23% 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 this has contributed greatly to the success of PMOPT.

We formally described and characterized the scheduling problem as NP-complete, and discussed a heuristic algorithm for solving it. Our experiments on a real-world example showed significantly reduced downtime (with up to 65%) and costs. Experiments with CPLEX 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 with up to 12 %. Future plans include fleet level planning and labor resource optimization and scheduling, and application to other domains. We are also considering investigating solution sensitivity with regard to parameter changes.

Acknowledgment

We are thankful to Pontus Slottnér and Per Almroth at Siemens Industrial Turbomachinery AB for providing expert knowledge regarding gas turbine lifetime predictions.

References

- Bohlin, M.; Wärja, M.; Holst, A.; Slottnér, P.; and Doganay, K. 2009. Optimization of Condition-Based Maintenance for Industrial Gas Turbines: Requirements and Results. In *Proc. ASME Turbo Expo 2009*. To Appear.
- Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, second edition.
- Dekker, R. 1996. Applications of Maintenance Optimization Models: A Review and Analysis. *Reliability Engineering and System Safety* 51:229–240.
- E. G. Coffman, J.; Garey, M. R.; and Johnson, D. S. 1997. *Approximation Algorithms for Bin Packing: A Survey*. Boston, USA: PWS Publishing Co. chapter 2, 46–93.
- Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.
- Lhomme, O. 1993. Consistency Techniques for Numeric CSPs. In *Proc. IJCAI-93*, 232–238.
- Marseguerra, M.; Zio, E.; and Podofillini, L. 2002. Condition-Based Maintenance Optimization by Means of Genetic Algorithms and Monte Carlo Simulation. *Reliability Engineering & System Safety* 77(2):151–165.
- Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition.
- Tan, J. S., and Kramer, M. A. 1997. A General Framework for Preventive Maintenance Optimization in Chemical Process Operations. *Computers & Chemical Engineering* 21(12):1451–1469.
- Wildeman, R. E.; Dekker, R.; and Smit, A. C. J. M. 1997. A Dynamic Policy for Grouping Maintenance Activities. *European Journal of Operational Research* 99(3):530–551.
- William D. Harvey, M. L. G. 1995. Limited Discrepancy Search. In *Proc. IJCAI-95*, volume 1, 607–615. Montréal, Québec, Canada: Morgan Kaufmann, 1995.
- Yamayee, Z.; Sidenblad, K.; and Yoshimura, M. 1983. A Computationally Efficient Optimal Maintenance Scheduling Method. *IEEE Transactions on Power Apparatus and Systems* PAS-102(2):330–338.
- Zienkiewicz, O. C.; Taylor, R.; and Zhu, J. Z. 2005. *The Finite Element Method*. Butterworth-Heinemann, sixth edition.