Coordinated Planning of Preventive Maintenance in Hierarchical Production Systems
G. van Dijkhuizen en A. van Harten

PR-18

Te publiceren in

Keywords
Coordinated Planning, Production Systems
Operations Management of Supply/Distribution Processes

BETA-publicatie : PR-18
ISSN : 1386-9213; PR-18
NUGI : 684
Enschede : Oktober 1997
Keywords : Coordinated Planning, Production Systems
BETA-Research Programme : Operations Management of Supply/Distribution Processes
Te publiceren in : Management Science
Coordinated Planning of Preventive Maintenance in Hierarchical Production Systems

Gerhard van Dijkhuizen and Aart van Harten

School of Management Studies, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
e-mail: g.c.vandijkhuizen@sms.utwente.nl

Abstract

We consider a technical system consisting of multiple different components, which are all subject to failure. Creating an occasion for preventive maintenance on one of these components requires a collection of preparatory set-up activities to be carried out in advance, with corresponding set-up costs. Since different components may require one or more shared set-up activities, there is a perspective of significant gains if preventive maintenance activities are carried out simultaneously. In this paper, we consider the case where each component is maintained preventively at an integer multiple of a certain basis interval, which is the same for all components. A general mathematical framework is presented which allows for multiple set-up activities, multiple components, and a large class of preventive maintenance strategies for each component.

1 Introduction

Every few years, new surveys appear on maintenance optimization, showing the use and benefits of mathematical models in the maintenance area, e.g. McCall (1965), Pierskalla and Voelker (1976), Sherif and Smith (1981), Cho and Parlar (1991), and Dekker (1996). During the last two decades, a growing interest can be observed in the modelling and optimization of inspection, maintenance and repair in multi-component systems. Most of these models derive their value from the existence of economies of scale in carrying out maintenance activities simultaneously. The reader is referred to Wildeman (1996) for an extensive literature review on multi-component maintenance models with economic dependence.

By now, there are several methods available that can handle multiple components. Most of them, however, suffer from intractability when the number of components increases (cf. Vanneste 1992), or are restricted to only one set-up activity (cf. Wildeman 1996). The latter implies that creating an occasion for preventive maintenance involves a fixed set-up cost, irrespective of how many and which components are maintained. Although this might be an interesting approach from a theoretical point of view, it is obvious that nowadays advanced production systems (e.g. aircrafts, offshore platforms, nuclear power plants) are much more complicated. In our opinion, preventive maintenance models should at least account for
Preventive Maintenance in Hierarchical Systems

2 General Approach

Consider a technical system consisting of \( m \) set-up activities, denoted \( I = \{1, \ldots, m\} \), and \( n \) components, denoted \( J = \{1, \ldots, n\} \). Creating an opportunity for preventive maintenance on component \( j \in J \) requires a collection \( I_j \subseteq I \) of preparatory set-up activities to be carried out in advance, with corresponding costs \( \sum_{i \in I_j} s_i \). Here, \( s_i > 0 \) denotes the individual cost of set-up activity \( i \in I \). If components are maintained simultaneously, the corresponding set-up activities can be combined. Hence, preventive maintenance on a subset of components \( J \subseteq J \) involves a set-up cost \( S(J) \), which depends completely on the set of required set-up activities:

\[
S(J) = \sum_{i \in \bigcup_{j \in J} I_j} s_i
\]

Within this setting, we consider preventive maintenance activities of the block type. To be specific, each component is maintained preventively at fixed intervals (e.g. daily, weekly,
Preventive Maintenance in Hierarchical Systems

Table 1: Example of a maintenance cycle with \( m = n = 3, k_1 = 1, k_2 = 3, \) and \( k_3 = 8 \): \( \Delta_1(k) = 1, \Delta_2(k) = 5/12, \Delta_3(k) = 1/8. \)

<table>
<thead>
<tr>
<th>components</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 2 3 2 2 2 2 2 2 2 2 2</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>set-up activities</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 2 2 2 2 2 2 2 2 2 2 2</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>frequencies</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 3 8 3 3 3 8 3 3 3 3 3 3 3 3 3 3 3 3</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

monthly, yearly), whereas corrective maintenance is carried out upon failure. With \( \Phi_j(x) \), we denote the expected maintenance costs (exclusive of preventive set-up costs) of component \( j \in J \) per unit of time, if maintained preventively every \( x > 0 \) time units. For notational convenience, and without loss of generality, we restrict ourselves to cost functions of the following type:

\[
\Phi_j(x) = \frac{c_j + M_j(x)}{x}
\]  

(2)

Here, \( c_j > 0 \) reflects the expected cost of preventive maintenance on component \( j \in J \). Moreover, \( M_j(x) \) denotes the expected cumulative deterioration costs of component \( j \in J \) (due to failures, repairs, etc.), \( x \) time units after its last preventive maintenance. By doing this, a variety of maintenance models can be incorporated, allowing different models for each component. The reader is referred to Dekker (1995) for an extensive list of block-type models.

2.1 Notation

Throughout this paper, we will use the following notation in order to identify the relationships between set-up activities and components in the maintenance tree (see Figure 1). As a starting point, we denote with \( I_j \subseteq T \) the subset of set-up activities \( i \in T \) that are required for component \( j \in J \). Similarly, we let \( J_i \subseteq J \) denote the subset of components \( j \in J \) that require set-up activity \( i \in T \). Moreover, \( J^*_i \subseteq J \) denotes the subset of components \( j \in J_i \) that are attached to set-up activity \( i \in T \). Finally, \( S_i \subseteq T \) represents the successors of set-up activity \( i \in T \). As an example, consider the maintenance cycle of Table 1. Then it is easily verified that \( I_1 = \{1\}, I_2 = \{1,2\}, \) and \( I_3 = \{1,2,3\} \). Moreover, \( J_1 = \{1,2,3\}, J_2 = \{2,3\}, \) and \( J_3 = \{3\} \), whereas \( J^*_1 = \{1\}, J^*_2 = \{2\}, \) and \( J^*_3 = \{3\} \). Finally, we have \( S_1 = \{2,3\}, S_2 = \{3\}, \) and \( S_3 = \emptyset. \)
2.2 Problem definition

The coordination of preventive maintenance activities is now modelled as follows: preventive maintenance on component $j \in J$ is carried out every $k_j \cdot t$ time units, where $k_j \in \mathbb{N}^*$ reflects the frequency of component $j$ relative to a basis maintenance interval of $t > 0$ time units, which is the same for all components. As a starting point, we consider the case where both the maintenance interval $t$ and the maintenance frequencies $k_j$ must be chosen from a finite set of possibilities, say $t \in T$ and $k_j \in K$ for all $j \in J$. In general, this yields an optimization problem in $n + 1$ variables $(t, k_1, ..., k_n)$, where our objective is to minimize average maintenance costs per unit of time:

$$
\begin{align*}
\text{(P)} & \quad \min_{t \in T} \min_{k_j \in K} \left\{ \sum_{i \in I} s_i \cdot \Delta_i(k) \frac{1}{t} + \sum_{j \in J} \Phi_j(k_j \cdot t) \right\}
\end{align*}
$$

Here, $\Delta_i(k)$ represents the fraction of times that set-up $i \in I$ is carried out at an occasion for preventive maintenance (see Table 1). For notational convenience, let us now denote with $J_i = \{ j \in J \mid i \in I_j \}$ the set of components that require set-up activity $i \in I$. Then it is easily observed that $\max\{k_j^{-1} \mid j \in J_i\} \leq \Delta_i(k) \leq 1$, implying that $\Delta_i(k) = 1$ if $k_j = 1$ for some $j \in J_i$. In general, we have the following expression for $\Delta_i(k)$ (cf. Dagnupar 1982).

$$
\begin{align*}
\Delta_i(k) &= \sum_{l=1}^{\left|J_i\right|} (-1)^{l+1} \sum_{\{j_1, ..., j_l\} \subseteq J_i} \operatorname{lcm}(k_{j_1}, ..., k_{j_l})^{-1}
\end{align*}
$$

Typically, the basis maintenance interval $t$ is restricted to several days or weeks, whereas the corresponding maintenance cycle $\operatorname{lcm}(k_1, ..., k_n) \cdot t$ varies from several weeks to several months, or even years. In particular, this phenomenon can be observed in calendar-based maintenance planning systems, by which workload and capacity profiles have to be matched on a regular basis. There are situations, however, where there is no explicit need for this kind of regulation. In such cases, our optimization problem becomes even more complex:

$$
\begin{align*}
\text{(Q)} & \quad \min_{t > 0} \min_{k_j \in \mathbb{N}^*} \left\{ \sum_{i \in I} s_i \cdot \Delta_i(k) \frac{1}{t} + \sum_{j \in J} \Phi_j(k_j \cdot t) \right\}
\end{align*}
$$

A typical example of this type can be found in e.g. aircraft maintenance, where the initiation of expensive maintenance activities is usually based on flight hours. In such cases, the need for well-defined maintenance intervals (e.g. multiples of 100 flight hours) is often motivated by intuitive reasoning.
2.3 Literature

The idea of planning maintenance at integer multiples of a certain basis interval originates from multi-item inventory theory (cf. Bomberger 1966), where cost savings can be obtained by coordinating replenishment of several items. It was introduced into maintenance modelling by Gertsbakh (1972), who considered a somewhat similar but less powerful modelling framework for multiple set-up activities. Since then, applications into maintenance modelling have been sparse, e.g. see Sculli and Suraweera (1979), Gits (1984) and van Dijkhuizen and van Harten (1997). At the same time, the single set-up version of our problem ($m = 1$) has gained considerable attention in literature. Pioneering work on this subject was carried out by Goyal and Kusy (1985) and Goyal and Gunasekaran (1992). Just recently, Wildeman (1996) presented a mathematical framework which allows for a large class of deterioration cost functions, and solved this problem to optimality. Unfortunately, this framework is based on the assumption that the correction factor $\Delta_i(k)$ is equal to one in the optimal strategy. Within our setting of multiple set-up activities ($m > 1$), it is obvious that this assumption cannot be justified, since $\Delta_i(k) = 1$ for all $i \in I$ would leave us with a single set-up problem. In other words, the possibility that $\Delta_i(k) < 1$ for some $i \in I$ is essential in our approach.

In general, finding the optimal maintenance strategy with correction factor is a very complex problem, even in case of a single set-up activity (cf. Goyal 1982).

2.4 Outline

The outline of this paper is as follows. In section 3, we will show that problem ($P$) can be formulated in terms of a surprisingly efficient mixed integer linear program. Subsequently, an heuristic approach for problem ($Q$) is presented in section 4. This approach is based on decomposition of problem ($Q$) into three subproblems: one that determines an initial value for $t$, denoted ($Q_{rel}$); one that determines the optimal values of $(k_1, ..., k_n)$ given $t$, denoted ($Q_k$); and one that determines the optimal value of $t$ given $(k_1, ..., k_n)$, denoted ($Q_t$). In section 5, our heuristic approach is illustrated by means of a numerical example. Moreover, computational results in section 6 indicate that near-optimal solutions are obtained within reasonable CPU times. Finally, some concluding remarks are summarized in section 7.

3 Analysis of problem ($P$)

In this section, we will show that problem ($P$) can be formulated in terms of a surprisingly efficient mixed integer linear program. For notational convenience, and without loss of generality, a less efficient but more insightful version is presented first. Subsequently, two reduction techniques are presented which strongly reduce the size of this problem, in particular if the set of possible frequencies $K$ is large.

3.1 Problem decomposition

First of all, let us denote with $I_0 = I_1 \cap ... \cap I_n$ the set of common set-up activities, i.e. set-up activities that are required for each component. Then we can assume without loss of
generality that \(|I_0| = 1\). To see this, \(I_0 = \emptyset\) implies that there is no common set-up activity, in which case the maintenance tree can be decomposed into two or more subtrees which can be treated separately. On the other hand, \(|I_0| > 1\) implies the existence of two or more common set-up activities, which obviously can be merged to a single set-up activity without affecting the problem (see Figure 2). For similar reasons, we can assume that \(J_i^* \neq \emptyset\) for all \(i \in I\).

### 3.2 A mixed integer linear programming formulation

For each \(t \in T\), the assignment of maintenance frequencies \(k \in \mathcal{K}\) to components \(j \in \mathcal{J}\) results in a maintenance cycle of at most \(\text{lcm}(\mathcal{K}) \cdot t\) time units. Here, \(\text{lcm}(k_1, ..., k_p)\) represents the least-common-multiple of the integers \(k_1, ..., k_p\), for example \(\text{lcm}(2, 3, 4) = 12\). With \(\mathcal{L} = \{1, ..., \text{lcm}(\mathcal{K})\}\), we denote the set of so-called maintenance opportunities. For notational convenience, and without loss of generality, we assume that each component is maintained preventively at the end of the maintenance cycle, i.e. at maintenance opportunity \(l = \text{lcm}(\mathcal{K})\).

In line with this, we denote with \(K_I = \{k \in \mathcal{K} \mid l \mod k = 0\}\) the set of maintenance frequencies that correspond with maintenance opportunity \(l \in \mathcal{L}\) (e.g. \(K_8 = K_{16} = \{1, 8\}\) in Table 1). The problem now consists of assigning maintenance frequencies to components, in such a way that the costs of the corresponding maintenance cycle are minimized. In our model, the assignment of set-up activities to maintenance opportunities is comprised into variables \(x_{il}\), with \(i \in I\) and \(l \in \mathcal{L}\), whereas the assignment of maintenance frequencies to components is represented by variables \(y_{jk}\), with \(j \in \mathcal{J}\) and \(k \in \mathcal{K}\):

\[
x_{il} = \begin{cases} 
1 & \text{if set-up } i \in I \text{ is assigned to opportunity } l \in \mathcal{L} \\
0 & \text{otherwise}
\end{cases}
\]

\[
y_{jk} = \begin{cases} 
1 & \text{if component } j \in \mathcal{J} \text{ is assigned to frequency } k \in \mathcal{K} \\
0 & \text{otherwise}
\end{cases}
\]

With \(a_{il} > 0\) we denote the average cost per time unit of assigning set-up activity \(i \in I\) to maintenance opportunity \(l \in \mathcal{L}\). Similarly, \(b_{jk} > 0\) denotes the average cost per time unit of assigning maintenance frequency \(k \in \mathcal{K}\) to component \(j \in \mathcal{J}\). With this in mind, the following expressions for \(a_{il}\) and \(b_{jk}\) are easily verified:
With \( a_{il} \) and \( b_{jk} \) as defined above, our problem can now be formulated in terms of a rather straightforward and inefficient mixed integer linear program, where our objective is to minimize average maintenance costs per unit of time:

\[
\text{Minimize } \sum_{i \in I} \sum_{l \in \mathcal{L}} a_{il} \cdot x_{il} + \sum_{j \in J} \sum_{k \in K} b_{jk} \cdot y_{jk}
\]

Subject to

\[
\begin{align*}
(a) & \quad x_{il} \geq x_{i'l'} \quad \text{for all } i \in \mathcal{I}, l \in \mathcal{L}, i' \in S_i \\
(b) & \quad x_{il} \geq y_{jk} \quad \text{for all } i \in \mathcal{I}, l \in \mathcal{L}, j \in J^*, k \in K_l \\
(c) & \quad \sum_{k \in K} y_{jk} = 1 \quad \text{for all } j \in \mathcal{J} \\
(d) & \quad x_{il} \geq 0 \quad \text{for all } i \in \mathcal{I}, l \in \mathcal{L} \\
(e) & \quad y_{jk} \in \{0, 1\} \quad \text{for all } j \in \mathcal{J}, k \in K
\end{align*}
\]

Here, restrictions (a) and (b) reflect that the assignment of a certain frequency to a certain component requires the corresponding set-up activities to be carried out at the corresponding maintenance opportunities too. Moreover, restrictions (c) and (e) guarantee that exactly one frequency is assigned to each component. As a consequence, restriction (d) is sufficient to ensure that \( x_{il} \in \{0, 1\} \) for all \( i \in \mathcal{I} \) and \( l \in \mathcal{L} \).

### 3.3 Problem reduction

Since the number of maintenance opportunities \( \text{lcm}(k_1, \ldots, k_p) \) grows exponentially with the set of possible maintenance frequencies \( \mathcal{K} = \{k_1, \ldots, k_p\} \), it seems worthwhile to provide a more efficient mixed integer linear programming formulation. This can be done by observing that:

(i) maintenance opportunities \( l \in \mathcal{L} \) with \( K_l = \emptyset \) can be left out of consideration, since evidently \( x_{il} = 0 \) for all \( i \in \mathcal{I} \) in any optimal solution to (8);

(ii) maintenance opportunities \( l_1, l_2 \in \mathcal{L} \), with \( K_{l_1} = K_{l_2} \) and \( l_1 \neq l_2 \), can as well be replaced by a single maintenance opportunity \( l \) with \( K_l = K_{l_1} = K_{l_2} \) and corresponding cost \( a_{il} = a_{il_1} + a_{il_2} \) for all \( i \in \mathcal{I} \), since evidently \( x_{il_1} = x_{il_2} \) for all \( i \in \mathcal{I} \) in any optimal solution to (8).
Obviously, observations (i) and (ii) may lead to significant reductions in the problem size. To be specific, if we denote with $\eta(U)$ the number of times that frequency cluster $U \subseteq \mathcal{K}$ shows up in the maintenance cycle (e.g. $\eta\{1,3\} = 7$ and $\eta\{1,8\} = 2$ in Table 1), then obviously the number of opportunities $|\mathcal{L}^*|$ in the reduced version of the problem equals the number of clusters $U \subseteq \mathcal{K}$ with $\eta(U) > 0$. In general, $\eta(U)$ yields an expression which is similar to (4):

$$\eta(U) = \text{lcm}(\mathcal{K}) \cdot \left\{ \frac{|\mathcal{K}|}{\sum_{i=1}^{U} (-1)^{i-\frac{1}{2}|U|} \sum_{U \subseteq \{k_1,\ldots,k_i\} \subseteq \mathcal{K}} \frac{\text{lcm}(k_1,\ldots,k_i)^{-1}}{\text{lcm}(U)} \right\}$$

(9)

In general, the determination of $\eta(U)$ for all $U \subseteq \mathcal{K}$ with the use of (9) is a complex problem, since the number of clusters $U \subseteq \mathcal{K}$ grows exponentially with the number of possible maintenance frequencies $\mathcal{K}$. For similar reasons, enumeration of all maintenance opportunities is not desirable. Apparently, an efficient method has to be constructed. In this respect, it seems useful to observe that $\eta(U) > 0$ if and only if $\text{lcm}(U) \mod k \neq 0$ for all $k \in \mathcal{K} \setminus U$. By doing this, the majority of clusters can be discarded beforehand. Moreover, the remaining clusters can be evaluated by means of the following implicit relation, which holds for all $U \subseteq \mathcal{K}$. Here, we define $\text{lcm}(\emptyset) = 1$ for notational convenience:

$$\text{lcm}(U) \cdot \sum_{V \supseteq U} \eta(V) = \text{lcm}(\mathcal{K})$$

(10)

Our analysis now proceeds as follows. As a starting point, the set of maintenance frequencies $\mathcal{K} = \{k_1,\ldots,k_p\}$ is ordered such that $k_1 < \ldots < k_p$. Subsequently, $\Omega_i$ is defined as the collection of clusters $U \subseteq \{k_1,\ldots,k_i\}$ with $\eta(U) > 0$, in case the set of possible maintenance frequencies would be reduced to $\mathcal{K} = \{k_1,\ldots,k_i\}$. Obviously, we have $\Omega_1 = \{\{1\}\}$ if $k_1 = 1$, and $\Omega_1 = \{\emptyset,\{k_1\}\}$ otherwise. Since we are mainly interested in $\Omega_p$, it is now sufficient to formulate an (efficient) procedure which constructs $\Omega_{i+1}$ from $\Omega_i$. It is easily verified that this can be done in the following, rather straightforward way. Starting with $\Omega_{i+1} = \emptyset$, it is determined for each $U \in \Omega_i$ whether (i) $\text{lcm}(U) \mod k_{i+1} \neq 0$, and (ii) $\text{lcm}(U \cup \{k_{i+1}\}) \mod k \neq 0$ for all $k \in \{k_1,\ldots,k_i\} \setminus U$. If condition (i) is satisfied, $\Omega_{i+1}$ is extended with cluster $U$. Moreover, if condition (ii) is satisfied, $\Omega_{i+1}$ is extended with cluster $U \cup \{k_{i+1}\}$. Once $\Omega_p$ has been determined, the values of $\eta(U)$ for all $U \in \Omega_p$ are easily obtained by recursion:

$$\eta(U) = \frac{\text{lcm}(\mathcal{K})}{\text{lcm}(U)} - \sum_{V \in \Omega_p : V \supseteq U} \eta(V)$$

(11)

As an example, consider the maintenance cycle of Table 1, with $\mathcal{K} = \{1,3,8\}$. Then it is easily verified that $\Omega_1 = \{\{1\}\}$, $\Omega_2 = \{\{1\},\{1,3\}\}$, and $\Omega_3 = \{\{1\},\{1,3\},\{1,8\},\{1,3,8\}\}$. Starting with $\eta\{1,3,8\} = 1$ by definition, the remaining clusters are evaluated as follows:

$$\eta\{1,3\} = \frac{\text{lcm}(1,3,8)}{\text{lcm}(1,3)} - \eta\{1,3,8\} = 24/3 - 1 = 7,$$
Preventive Maintenance in Hierarchical Systems

Table 2: Reductions in problem size for several frequency sets of the form $\mathcal{K} = \{1, \ldots, n\}$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n = 1$</th>
<th>$n = 2$</th>
<th>$n = 3$</th>
<th>$n = 5$</th>
<th>$n = 10$</th>
<th>$n = 15$</th>
<th>$n = 25$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>\mathcal{L}</td>
<td>$</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>60</td>
<td>2520</td>
</tr>
<tr>
<td>$</td>
<td>\mathcal{L}^*</td>
<td>$</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>12</td>
<td>48</td>
</tr>
</tbody>
</table>

$\eta\{1,8\} = \text{lcm}(1,3,8)/\text{lcm}(1,8) - \eta\{1,3,8\} = 24/8 - 1 = 2,$

$\eta\{1\} = \text{lcm}(1,3,8)/\text{lcm}(1) - \eta\{1,3\} - \eta\{1,8\} - \eta\{1,3,8\} = 24/1 - 7 - 2 - 1 = 14.$

Clearly, this approach yields an alternative problem formulation with significantly less variables and constraints. In general, the reductions in problem size increase exponentially with the number of possible maintenance frequencies (see Table 2). Of course, our MILP formulation was tested on a series of randomly created test problems. For the outcomes of these numerical experiments, we refer to the computational results in section 6. Here, we only mention that the LP-relaxation of problem (8) often generates an integer, and thus feasible solution to problem $(P)$. In our view, this phenomenon is caused by the fact that our clustering problem is closely related to the standard assignment problem, which is known to possess the above-mentioned property.

4 Analysis of problem $(Q)$

In general, problem $(Q)$ is of a very complex nature, and can only be solved by local search techniques (e.g. simulated annealing). Under some weak conditions, however, its complexity can be reduced significantly. In this section, we will present an heuristic approach to problem $(Q)$, which is based on the assumptions mentioned below.

4.1 Model assumptions

To simplify our analysis, we assume that $M_j(x)$ is (i) twice differentiable, (ii) strictly positive, (iii) strictly increasing, and (iv) strictly convex on $(0, \infty)$, for all $j \in \mathcal{J}$. From a practical point of view, these assumptions are related to an increasing (marginal) cost rate for postponing preventive maintenance activities (cf. Berg 1980). From a theoretical point of view, they enable us to formulate some efficient solution procedures from convex analysis. Moreover, they account for a large class of preventive maintenance models of the block type, including the minimal repair model (Barlow and Hunter, 1960), the standard inspection model (Barlow et al., 1963), and the delay time model (Christer and Waller, 1984). On the other hand, if one or more components are modelled according to a standard block replacement model (Barlow and Proschan, 1965), assumption (iv) may not be satisfied. In that case, our methods can still be used to obtain approximate results.

Lemma 1 If $M_j(x)$ is twice differentiable, strictly positive, strictly increasing and strictly convex on $(0, \infty)$, then $\Phi_j(x)$ has exactly one local minimum $x^*_j < \infty$. Moreover, $\Phi_j(1/x)$ is a convex function on $(0, \infty)$. 
Preventive Maintenance in Hierarchical Systems

Proof. As a starting point, observe that \( x_j^* \) must at least satisfy \( \Phi_j'(x_j^*) = 0 \) and \( \Phi_j''(x_j^*) > 0 \), which is equivalent to \( M_j'(x_j^*) = \Phi_j(x_j^*) \) and \( M_j''(x_j^*) > 0 \). Since \( \Phi_j(x_j^*) < \infty \) by definition, and \( M_j'(x) \) is strictly increasing in \( x \), this implies the uniqueness of \( x_j^* \). For similar reasons, \( \lim_{x \to \infty} M_j(x) = \infty \) yields \( x_j^* < \infty \). Moreover, since \( \frac{\partial^2}{\partial x^2} \Phi_j(1/x) = M''(x)/x^3 \), we also have that \( \frac{\partial^2}{\partial x^2} \Phi_j(1/x) \geq 0 \) for all \( x \in (0, \infty) \), and thus \( \Phi_j(1/x) \) is a convex function on \( (0, \infty) \).

4.2 Alternative formulation

As a result of Lemma 1, it seems reasonable to rewrite our problem by transformation of \( t \) into \( t^{-1} \) in (5). By doing this, the objective function becomes a convex function in \( t \), which obviously is a useful property in finding the optimal preventive maintenance cycle. In this alternative formulation, preventive maintenance on component \( j \in \mathcal{J} \) is carried out every \( k_j/t \) time units, where \( k_j \in \mathbb{N}^* \) denotes the frequency of component \( j \in \mathcal{J} \) relative to a basis maintenance interval of \( t^{-1} > 0 \) time units:

\[
(Q) \quad \min_{t > 0} \min_{k_j \in \mathbb{N}^*} \left\{ \sum_{i \in \mathcal{I}} s_i \cdot \Delta_i(k) \cdot t + \sum_{j \in \mathcal{J}} \Phi_j(k_j/t) \right\}
\]

4.3 Problem relaxation

In this section, we will present a relaxation of problem \((Q)\), which enables us to construct a lower bound for the optimal solution with the use of standard search techniques. To a certain extent, this analysis is similar to Wildeman (1996), who considers the case of a single set-up activity \((m=1)\). However, it contains some interesting new elements, which are typical for our setting of multiple set-up activities. As a starting point, define scalars \( 0 \leq \alpha_{ij} \leq 1 \), such that \( \sum_{i \in I_j} \alpha_{ij} = 1 \) for all \( j \in \mathcal{J} \), and observe that the following relation holds:

\[
\sum_{j \in \mathcal{J}} \sum_{i \in I_j} \alpha_{ij} \cdot \Phi_j(k_j/t) = \sum_{j \in \mathcal{J}} \sum_{i \in I_j} \alpha_{ij} \cdot \Phi_j(k_j/t)
\]

Here, \( \alpha_{ij} \) could be interpreted as the contribution of set-up activity \( i \in I_j \) to the costs of component \( j \in \mathcal{J} \). With \( \mathcal{A} \subseteq \mathbb{R}^{mn} \), we denote the set of feasible solutions to \((Q)\). For each \( \alpha \in \mathcal{A} \), this yields an alternative formulation for problem \((Q)\), in which the individual costs of each component are divided among the corresponding set-up activities:

\[
\min_{t > 0} \min_{k_j \in \mathbb{N}^*} \sum_{i \in \mathcal{I}} \left\{ s_i \cdot \Delta_i(k) \cdot t + \sum_{j \in I_i} \alpha_{ij} \cdot \Phi_j(k_j/t) \right\}
\]

To continue our analysis, we substitute \( t_i = \Delta_i(k) \cdot t \) in equation (14), and observe that \( t_i \geq t_j \) if \( j \in S_i \), since obviously \( \Delta_i(k) \geq \Delta_j(k) \) if set-up activity \( i \in \mathcal{I} \) is a parent of set-up activity \( j \in \mathcal{I} \). If we denote with \( \mathcal{T} \subseteq \mathbb{R}^m_+ \) the set of feasible solutions to \((t_1, ..., t_m)\), this
yields the following lower bound for problem (14):

$$\min_{t \in T} \min_{k_i \in \mathbb{N}} \sum_{i \in I} \left\{ s_i \cdot t_i + \sum_{j \in J_i} \alpha_{ij} \cdot \Phi_j(k_j \cdot \Delta_i(k)/t_i) \right\}$$  \hspace{1cm} (15)

Now we substitute $k_{ij} = k_j \cdot \Delta_i(k)$ in equation (15), and observe that $k_{ij} \geq 1$ for all $i \in I$ and $j \in J_i$, since $\Delta_i(k) \geq k_j^{-1}$ for all $j \in J_i$ by definition. Again, this yields an alternative lower bound for problem (15):

$$\min_{t \in T} \min_{k_{ij} \geq 1} \sum_{i \in I} \left\{ s_i \cdot t_i + \sum_{j \in J_i} \alpha_{ij} \cdot \Phi_j(k_{ij}/t_i) \right\}$$  \hspace{1cm} (16)

For notational convenience, and without loss of generality, let us now define functions $\xi_j(t) = \min\{\Phi_j(x/t) | x \geq 1\}$ for all $j \in J$. Since $\Phi_j(\cdot)$ attains its (unique) minimum at $x_j^* < \infty$, and $\Phi_j(1/x)$ is convex on $(0, \infty)$, this yields a decreasing and convex function in $t$:

$$\xi_j(t) = \min_{x \geq 1} \{\Phi_j(x/t)\} = \begin{cases} 
\Phi_j(1/t) & \text{if } t < 1/x_j^* \\
\Phi_j(x_j^*) & \text{if } t \geq 1/x_j^*
\end{cases}$$  \hspace{1cm} (17)

Since we are free to choose $\alpha \in A$ arbitrarily, this leaves us with the following lower bound for problem (Q). Obviously, this lower bound reflects a decomposition into $m$ interrelated subproblems, each corresponding with a single set-up activity, and its adjacent components:

$$\min_{t \in T} \max_{\alpha \in A} \sum_{i \in I} \left\{ s_i \cdot t_i + \sum_{j \in J_i} \alpha_{ij} \cdot \xi_j(t_i) \right\}$$  \hspace{1cm} (18)

Our analysis now proceeds as follows. Since $\xi_j(t)$ is a decreasing function in $t$ for all $j \in J$, and $t_i \geq t_j$ if $j \in S_i$, it can easily be verified that the optimal values of $\alpha_{ij}$ in equation (18) are determined as follows: $\alpha_{ij} = 1$ if $j \in J_i^*$, and $\alpha_{ij} = 0$ otherwise. Summarizing, this leaves us with the following lower bound for problem (Q):

$$(Q_{rel}) \min_{t \in T} \sum_{i \in I} \left\{ s_i \cdot t_i + \sum_{j \in J_i^*} \xi_j(t_i) \right\}$$  \hspace{1cm} (19)

Since $s_i \cdot t_i$ and $\xi_j(t_i)$ are both convex functions in $t_i$, and $T$ is a convex space in $\mathbb{R}^m_+$, this yields a convex programming problem in a convex space, which can easily be solved to optimality using standard search techniques. Here, we used the gradient projection method, e.g. see Luenberger (1984).
4.4 An heuristic approach

Our heuristic approach is based on the assumption that $k_j^* = 1$ for some $j \in J$ in the optimal maintenance cycle, where $i = 1$ denotes the common set-up activity. It seems reasonable therefore to determine the optimal solution $(t_1^*, ..., t_m^*)$ to problem $(Q_{rel})$, and initialize $t^* = t_1^*$. Our heuristic now proceeds by solving subproblems $(Q_k)$ and $(Q_t)$ iteratively, until no improvements are observed in two consecutive iterations. Here, subproblem $(Q_k)$ determines the maintenance frequencies $(k_1^*, ..., k_m^*)$ given the basis maintenance interval $1/t^*$, whereas $(Q_t)$ determines the maintenance interval $1/t^*$ given the maintenance frequencies $(k_1^*, ..., k_m^*)$. In the following section, it is shown that problem $(Q_t)$ can be solved to optimality with the use of standard search techniques. Subsequently, we will show that problem $(Q_k)$ can be decomposed into a number of independent subproblems, which can be solved separately. Moreover, we will present two heuristic to solve these subproblems.

4.5 Analysis of problem $(Q_t)$

If the optimal maintenance frequencies $(k_1^*, ..., k_m^*)$ are known, it is possible to compute the values of $\Delta_i(k^*)$ beforehand for all $i \in I$, by using an analysis similar to section 3.3. To be specific, if we denote with $\eta_h(\emptyset)$ the number of empty maintenance opportunities in a maintenance cycle with frequency set $K_i = \{k_j^* \mid j \in J_i\}$, it is easily verified that the following relations holds:

$$\Delta_i(k^*) = 1 - \frac{\eta_h(\emptyset)}{\text{lcm}(K_i)} \quad (20)$$

Hence, if we denote with $s_i^* = s_i \cdot \Delta_i(k^*)$ the costs associated with set-up activity $i \in I$, problem $(Q)$ reduces to the following optimization problem:

$$(Q_t) \quad \min_{t > 0} \left\{ \sum_{i \in I} s_i^* \cdot t + \sum_{j \in J} \Phi_j(k_j^*/t) \right\} \quad (21)$$

Since $s_i^* \cdot t$ and $\Phi_j(k_j^*/t)$ are both convex in $t$, this leaves us with a convex programming problem, which can easily be solved to optimality using standard search techniques.

4.6 Analysis of problem $(Q_k)$

If the optimal basis maintenance interval $1/t^*$ is known, problem $(Q)$ reduces to the following optimization problem:

$$(Q_k) \quad \min_{k_j \in N^*} \left\{ \sum_{i \in I} s_i \cdot \Delta_i(k) \cdot t^* + \sum_{j \in J} \Phi_j(k_j/t^*) \right\} \quad (22)$$
Preventive Maintenance in Hierarchical Systems

Figure 3: Decomposition of problem \((Q_k)\) into \(|S_1| + 1\) independent subproblems.

Under the assumption that \(\Delta_1(k) = 1\) in the optimal solution to problem \((Q)\), it is easily verified that \((Q_k)\) can be decomposed into \(|S_1| + 1\) independent subproblems, which can be solved separately. To be specific, one of these subproblems corresponds to the common set-up activity \(i = 1\), and its adjacent components \(j \in J^*_1\), whereas each of the remaining \(|S_1|\) subproblems corresponds to the subtree associated with a second-level set-up activity \(i \in S_1\) (see Figure 3). In general, each subproblem is similar to problem \((Q_k)\), but refers to a reduced set of set-up activities and components. Obviously, this decomposition may lead to significant reductions in problem sizes and computation times, whatever algorithm we decide to use.

Let us now present two heuristics for problem \((Q_k)\). In the first heuristic, we restrict ourselves to a limited set of possible maintenance frequencies, and solve this problem by means of a mixed integer linear programming formulation. In the second heuristic, we consider the case where each set-up activity must be carried out at an integer multiple of its parental set-up activity, and show that this problem can be solved by means of an efficient dynamic programming formulation. For notational convenience, we will refer to these heuristic as the linear programming heuristic, and the dynamic programming heuristic respectively.

4.6.1 A linear programming heuristic

In the linear programming heuristic (LPH), each subproblem \((Q_k)\) is solved by means of an efficient mixed integer linear programming formulation, as was designed for problem \((P)\). To this end, the set of possible maintenance frequencies is reduced to \(K = \{k_{\text{min}}, ..., k_{\text{max}}\}\), where \(k_{\text{min}}\) and \(k_{\text{max}}\) are determined as follows. For each \(j \in J\), we determine the optimal maintenance interval \(x_j^*\) if no preventive set-up costs would be charged. Similarly, we let \(y_j^*\) denote the optimal maintenance interval for component \(j \in J^*_1\) if preventive set-up costs \(s_i\) would be charged:

\[
x_j^* = \arg \min_{x > 0} \left\{ \frac{c_j + M_j(x)}{x} \right\}
\]

(23)
Preventive Maintenance in Hierarchical Systems

\[ y_j^* = \arg \min_{y > 0} \left\{ \frac{s_i + c_j + M_j(y)}{y} \right\} \quad (24) \]

To a certain extent, \( x_j^* \) and \( y_j^* \) can be interpreted as a lower and upper bound on the optimal maintenance interval for component \( j \in J \). In line with this, \( k_{\min} \) and \( k_{\max} \) are based on the intuitive reasoning that the maintenance frequencies \( (k_1^*, \ldots, k_n^*) \) in any optimal solution to problem \( (Q_k) \) will satisfy:

\[ k_{\min} = \min_{j \in J} \left[ x_j^* \cdot t^* \right] \leq \min_{j \in J} \left\{ k_j^* \right\} \leq \max_{j \in J} \left\{ k_j^* \right\} \leq \max_{j \in J} \left[ y_j^* \cdot t^* \right] = k_{\max} \quad (25) \]

Since \( M_j(\cdot) \) is assumed to be strictly increasing and convex on \((0, \infty)\), it follows from Lemma 1 that \( 0 < x_j^* < y_j^* < \infty \) for all \( j \in J \), and thus \( 0 \leq k_{\min} \leq k_{\max} < \infty \). Obviously, we do not allow \( k_{\min} = 0 \), and in that case set \( k_{\min} = 1 \).

4.6.2 A dynamic programming heuristic

In the dynamic programming heuristic (DPH), each subproblem \( (Q_k) \) is solved by means of an efficient dynamic programming formulation. To this end, we restrict ourselves to the case where each set-up activity \( i \in I \) is carried out at an integer multiple of \( \hat{k}_i \) of the basis interval \( 1/t^* \). In line with this, components \( j \in J_i^* \) must be carried out at an integer multiple of \( \hat{k}_i/t^* \) time units, i.e. \( k_j \mod \hat{k}_i = 0 \) for all \( j \in J_i^* \). Moreover, each set-up activity must be carried out at an integer multiple of its parental set-up activity, i.e. \( \hat{k}_j \mod \hat{k}_i = 0 \) for all \( j \in S_i \). As a starting point, let us denote with \( f_i(k) \) the costs associated with set-up activity \( i \in I \) and components \( j \in J_i^* \), provided that set-up activity \( i \in I \) is carried out each \( k/t^* \) time units:

\[ f_i(k) = \frac{s_i \cdot t^*}{k} + \sum_{j \in J_i^*} \min_{l_j \geq 1} \{ \Phi_j(k \cdot l_j/t^*) \} \quad (26) \]

To continue our analysis, we denote with \( g_i(k) \) the minimal costs of the subtree associated with set-up activity \( i \in I \), provided that it must be carried out at an integer multiple of \( k/t^* \) time units. Then it is easily verified that \( g_i(k) \) can be determined recursively by means of the following dynamic programming equations:

\[ g_i(k) = \min_{l \geq 1} \left\{ f_i(k \cdot l) + \sum_{j \in S_i} g_j(k \cdot l) \right\} \quad (27) \]

The problem in the above formulation is that, in order to determine \( g_i(k) \), an infinite number of alternatives \( l \geq 1 \) must be evaluated. To avoid this, let us denote with \( l_i^*(k) \geq 1 \) the value of \( l \) for which \( f_i(k \cdot l) \) is minimized, and observe that \( f_i(k \cdot l) \geq f_i(k \cdot l_i^*(k)) \) and \( g_j(k \cdot l) \geq g_j(k \cdot l_i^*(k)) \) for all \( l > l_i^*(k) \). In other words, these alternatives can as well be
discarded beforehand without affecting the problem. Obviously, this yields an equivalent dynamic programming formulation, in which the number of alternatives $1 \leq l \leq l_i^*(k)$ is finite. The question remains how to determine $l_i^*(k)$ for all $i, k$. To this end, let us assume for a moment that $k \cdot l_i^*(k)/t^* > x_j^*$ for all $j \in J_i^*$. Under this assumption, it follows from Lemma 1 that $\Phi_j(k \cdot l/t^*)$ is increasing in $l$ for all $l \geq l_i^*(k)$, and thus $f_i(k \cdot l)$ can be rewritten as follows:

$$f_i(k \cdot l) = \frac{s_i \cdot t^*}{k \cdot l} + \sum_{j \in J_i^*} \Phi_j(k \cdot l/t^*) \quad l \geq l_i^*(k)$$  \hspace{1cm} (28)$$

Let us now substitute $\tau_i = t^*/(k \cdot l)$ in equation (28), and denote with $\tau_i^* > 0$ the unique minimum of the following convex programming problem:

$$\min_{\tau_i > 0} \left\{ s_i \cdot \tau_i + \sum_{j \in J_i^*} \Phi_j(1/\tau_i) \right\}$$  \hspace{1cm} (29)$$

Since we assumed that $k \cdot l_i^*(k)/t^* > x_j^*$ for all $j \in J_i^*$, it is now obvious that either $l_i^*(k) = \lfloor t^*/(k \cdot \tau_i^*) \rfloor$, or $l_i^*(k) = \lceil t^*/(k \cdot \tau_i^*) \rceil$. In all other cases, we have that $l_i^*(k) = \lfloor x_j^* \cdot t^*/k \rfloor$ for some $j \in J_i^*$. Summarizing, this leaves us with the following upper bound for $l_i^*(k)$. Here, we denote $\theta_i^* = \min\{1/x_j^* \mid j \in J_i^*\}$:

$$l_i^*(k) \leq \max \left\{ \left\lfloor \frac{t^*}{k \cdot \tau_i^*} \right\rfloor, \left\lceil \frac{t^*}{k \cdot \theta_i^*} \right\rceil \right\}$$  \hspace{1cm} (30)$$

Hence, if we determine $\tau_i^*$ and $\theta_i^*$ for each set-up activity $i \in \mathcal{I}$, the dynamic programming recursion can be used to determine the optimal maintenance cycle by evaluation of $g_1(1)$.

5 Numerical Example

Consider a technical system consisting of $m = 4$ set-up activities and $n = 4$ components, with $S_1 = \{2\}$, $S_2 = \emptyset$, $S_3 = \{4\}$ and $S_4 = \emptyset$, whereas $J_1^* = \{1\}$, $J_2^* = \{2\}$, $J_3^* = \{3\}$ and $J_4^* = \{4\}$. For all components $j \in \mathcal{J}$, the deterioration cost function $\Phi_j(\cdot)$ are of the following form. Here, $a_j > 0$, $b_j > 0$ and $c_j > 0$ are strictly positive constants:

$$\Phi_j(x) = \frac{a_j}{x} + b_j \cdot x^{c_j}$$  \hspace{1cm} (31)$$

The costs $s_i$ of set-up activities $i \in \mathcal{I}$, as well as the parameters $(a_j, b_j, c_j)$ for components $j \in \mathcal{J}$, are depicted in Table 3. From these parameters, the corresponding values of $x_j^*$ and $y_j^*$ for components $j \in \mathcal{J}$, as well as $t_i^*$, $\tau_i^*$ and $\theta_i^*$ for set-up activities $i \in \mathcal{I}$ are determined. Our heuristic now proceeds as follows. As a starting point, we determine the optimal solution to
Table 3: Parameter settings for numerical example.

<table>
<thead>
<tr>
<th>i, j</th>
<th>s_i</th>
<th>a_j</th>
<th>b_j</th>
<th>c_j</th>
<th>x_j^*</th>
<th>y_j^*</th>
<th>t_j^*</th>
<th>\theta_j^*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>92</td>
<td>453</td>
<td>19</td>
<td>3</td>
<td>1.679</td>
<td>1.758</td>
<td>0.569</td>
<td>0.527</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>125</td>
<td>14</td>
<td>2</td>
<td>1.647</td>
<td>1.890</td>
<td>0.529</td>
<td>0.529</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>252</td>
<td>11</td>
<td>1</td>
<td>4.786</td>
<td>5.027</td>
<td>0.365</td>
<td>0.288</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>132</td>
<td>23</td>
<td>1</td>
<td>2.396</td>
<td>2.528</td>
<td>0.365</td>
<td>0.396</td>
</tr>
</tbody>
</table>

Table 4: Consecutive iterations of the linear programming heuristic (LPH), and the dynamic programming heuristic (DPH), for the numerical example of Table 3.

<table>
<thead>
<tr>
<th></th>
<th>t^*</th>
<th>{k_1^<em>, ..., k_4^</em>}</th>
<th>{\Delta_1(k^<em>), ..., \Delta_4(k^</em>)}</th>
<th>costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPH</td>
<td>(Q_k)</td>
<td>0.569</td>
<td>{1, 1, 3, 2}</td>
<td>802.36</td>
</tr>
<tr>
<td></td>
<td>(Q_t)</td>
<td>0.575</td>
<td>{1, 1, 3, 2}</td>
<td>802.26</td>
</tr>
<tr>
<td></td>
<td>(Q_k)</td>
<td>0.575</td>
<td>{1, 1, 3, 2}</td>
<td>802.26</td>
</tr>
<tr>
<td>DPH</td>
<td>(Q_k)</td>
<td>0.569</td>
<td>{1, 1, 2, 2}</td>
<td>804.43</td>
</tr>
<tr>
<td></td>
<td>(Q_t)</td>
<td>0.562</td>
<td>{1, 1, 2, 2}</td>
<td>804.29</td>
</tr>
<tr>
<td></td>
<td>(Q_k)</td>
<td>0.562</td>
<td>{1, 1, 2, 2}</td>
<td>804.29</td>
</tr>
</tbody>
</table>

problem (Q_{rel}). This yields t_1^* = 0.569, t_2^* = 0.529, and t_3^* = t_4^* = 0.365, with corresponding lower bound 794.71. Subsequently, subproblems (Q_k) and (Q_t) are solved iteratively, until no improvements are observed in two consecutive iterations.

As can be observed from Table 4, the linear programming heuristic (LPH) generates a maintenance cycle of lcm{1, 2, 3}/t^* = 6/0.575 \approx 10.44 time units. Within this maintenance cycle, total maintenance costs amount to 10.44 \cdot 802.26 \approx 8376 on average, whereas preventive set-up costs equal 6 \cdot {92 + 64 + \frac{2}{3} \cdot 26 + \frac{1}{2} \cdot 15} = 1085, i.e. approximately 13.0\% of total maintenance costs. Similarly, the dynamic programming (DPH) generate a maintenance cycle of lcm{1, 2}/t^* = 2/0.562 \approx 3.56 time units. Within this maintenance cycle, total maintenance costs amount to 3.56 \cdot 804.29 \approx 2863 on average, whereas preventive set-up costs equal 2 \cdot \{92 + 64 + \frac{1}{2} \cdot (26 + 15)\} = 353, i.e. approximately 12.3\% of total maintenance costs. Moreover, the performance of the (LPH) and (DPH) heuristic, i.e. the maximal deviation with respect to their lower bound, equals 0.95\% and 1.21\% respectively.

6 Computational results

In this section, we will discuss the results of a series of numerical experiments, which were carried out to investigate the performance of both heuristics, expressed in quantitative as well as qualitative measures. To be specific, we tested our heuristics on several test problems, in which the number of set-up activities, the number of components, and the corresponding costs were varied randomly. In order to prevent \Delta_i(k) = 1 for all i \in \mathcal{I} as much as possible, we decided to attach exactly one component to each set-up activity. As a consequence, the number of set-up activities equals the number of components in each test problem, which in fact were drawn at random between 1 and 50. Moreover, set-up activities were attached to each other by chosing the parent of set-up activity i randomly among set-up activities j < i.
Preventive Maintenance in Hierarchical Systems

Table 5: Computational results for the linear programming heuristic (LPH), and the dynamic programming heuristic (DPH), based on 1000 randomly generated 'small' test problems with $s_i \in (1,100)$, $a_j \in (100,500)$, $b_j \in (10,50)$, and $c_j \in (1,5)$.

<table>
<thead>
<tr>
<th></th>
<th>(LPH)</th>
<th></th>
<th>(DPH)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>minimum</td>
<td>average</td>
<td>maximal</td>
</tr>
<tr>
<td># iterations</td>
<td>2</td>
<td>4.39</td>
<td>11</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>0.00</td>
<td>5.78</td>
<td>72.86</td>
</tr>
<tr>
<td>performance (%)</td>
<td>0.00</td>
<td>2.42</td>
<td>12.69</td>
</tr>
<tr>
<td>interval length</td>
<td>0.84</td>
<td>1.25</td>
<td>4.36</td>
</tr>
<tr>
<td>cycle length</td>
<td>1.03</td>
<td>10.89</td>
<td>117.69</td>
</tr>
</tbody>
</table>

6.1 Small test problems

First of all, we investigated the performance of both heuristics for a series of small test problems, i.e. problems which can be solved by both heuristics within reasonable computation times. Obviously, this requires that the set of possible maintenance frequencies does not become too large. This was achieved by choosing $s_i$, $a_j$, $b_j$ and $c_j$ in such a way that the corresponding values of $x_j^*$ and $y_j^*$ were not subject to extreme fluctuations. To be specific, the costs of set-up activities $i \in \mathcal{I}$ were drawn at random from $s_i \in (1,100)$, whereas the costs of component $j \in \mathcal{J}$ were drawn at random from $a_j \in (100,500)$, $b_j \in (10,50)$, and $c_j \in (1,5)$. As a consequence, $x_j^* \in (0.86,7.07)$ and $y_j^* \in (0.86,7.75)$ for all $j \in \mathcal{J}$.

From the results in Table 5, it can be concluded that the difference between the (LPH) and (DPH) heuristic is very small. In fact, a closer look at the results showed that both heuristics generated identical maintenance cycles in 971 out of 1000 test problems. In the remaining 29 test problems, the linear programming heuristic (LPH) outperformed the dynamic programming heuristic (DPH) in 28 cases, with a maximum of 0.26%. Apparently, the underlying structure between the dynamic programming heuristic does not affect the problem significantly. With respect to the (guaranteed) performance of both heuristics, we claim that an average deviation of 2.42% from the lower bound is quite satisfactorily. Moreover, a closer look at the results showed that in only 29 out of 1000 test problems, the performance was worse than 5%.

6.2 Large test problems

As mentioned before, the use of the linear programming heuristic (LPH) becomes undesirable if the set of possible frequencies becomes too large, e.g. $\mathcal{K} = \{1,...,100\}$. In that case, we have to use the dynamic programming heuristic (DPH), in order to obtain reasonable solutions within acceptable computation times. To this end, we carried out another series of 1000 test problems, in which the costs of set-up activities $i \in \mathcal{I}$ were drawn at random from $s_i \in (1,1000)$. Moreover, the costs of component $j \in \mathcal{J}$ were drawn at random from $a_j \in (1,1000)$, $b_j \in (1,100)$, and $c_j \in (1,10)$. As a consequence, $x_j^* \in (0.10,31.62)$ and $y_j^* \in (0.14,44.72)$ for all $j \in \mathcal{J}$, which clearly complicates the problem.

From the results in Table 6, it can be observed that the number of iterations as well as the
Table 6: Computational results for the dynamic programming heuristic (DPH) based on 1000 randomly generated 'large' test problems with $s_i \in (1,1000)$, $a_j \in (1,1000)$, $b_j \in (1,100)$, and $c_j \in (1,10)$.

<table>
<thead>
<tr>
<th></th>
<th>minimum</th>
<th>average</th>
<th>maximal</th>
</tr>
</thead>
<tbody>
<tr>
<td># iterations</td>
<td>2</td>
<td>3.89</td>
<td>9</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>0.00</td>
<td>0.05</td>
<td>0.61</td>
</tr>
<tr>
<td>performance (%)</td>
<td>0.00</td>
<td>2.85</td>
<td>17.39</td>
</tr>
<tr>
<td>interval length</td>
<td>0.78</td>
<td>1.15</td>
<td>5.20</td>
</tr>
<tr>
<td>cycle length</td>
<td>0.94</td>
<td>22.80</td>
<td>3147.09</td>
</tr>
</tbody>
</table>

computation times needed for the dynamic programming heuristic (DPH) hardly relate to the complexity of the test problems. On the other hand, the length of the corresponding maintenance cycle increases significantly, whereas the performance decreases slightly. Although these observations were to be expected, we claim that an average deviation of 2.85% from the lower bound is quite satisfactorily. In fact, a closer look at the results showed that in only 20 out of 1000 test problems, the (guaranteed) performance was worse than 7.5%.

7 Concluding remarks

In this study, we presented a variety of optimization techniques in order to create a cost-optimal preventive maintenance cycle for a production system with hierarchically ordered set-up activities and components. To be specific, we developed a linear programming heuristic (LPH) as well as a dynamic programming heuristic (DPH), which both enabled us to determine an (optimal) maintenance frequency for each component. Based on a series of 1000 randomly generated test problems, we concluded that the (DPH) heuristic should be preferred for its efficiency. It generates reasonable solutions within negligible computation times, even for production systems with a huge number of set-up activities and/or components.

In general, we claim that the problem of finding a cost-optimal preventive maintenance cycle has been solved quite satisfactorily by our methods. It is not difficult, however, to come up with a number of promising extensions to our modelling framework. As a starting point, one might be interested in the possibilities for adjusting the maintenance cycle, such that the peak workload is minimized, but overall maintenance costs are not affected. Similarly, it might be interesting to combine the maintenance cycles of different production systems to an overall maintenance cycle, in which the peak workload for the maintenance department is minimized. For the moment, these suggestions are left for future research.

References

Preventive Maintenance in Hierarchical Systems


