A Dynamic Policy for Grouping Maintenance Activities

R.E. Wildeman R. Dekker
Econometric Institute

A.C.J.M. Smit
Koninklijke/Shell-Laboratorium, Amsterdam[*]

Abstract

A maintenance activity carried out on a technical system often involves a system-dependent set-up cost that is the same for all maintenance activities carried out on that system. Grouping activities thus saves costs since execution of a group of activities requires only one set-up. Many maintenance models consider the grouping of maintenance activities on a long-term basis with an infinite horizon. This makes it very difficult to incorporate short-term circumstances such as opportunities or a varying use of components because these are either not known beforehand or make the problem intractable. In this paper we propose a rolling-horizon approach that takes a long-term tentative plan as a basis for a subsequent adaptation according to information that becomes available on the short term. This yields a dynamic grouping policy that assists the maintenance manager in his planning job. We present a fast approach that allows interactive planning by showing how shifts from the tentative planning work out. We illustrate our approach with examples.

Keywords: Maintenance, multi-component, planning, dynamic programming.

1 Introduction

The cost of maintaining a component of a technical system (such as a transportation fleet, a machine, a road, or a building) often consists of a cost that depends on the component involved and of a fixed cost that only depends on the system. In that case, the system-dependent cost, the so-called *set-up cost*, is shared by all maintenance activities carried out simultaneously on the system. For example, the set-up cost can consist of the downtime cost due to production loss if the system cannot be used during maintenance, or of the preparation cost associated with erecting a scaffolding or opening a machine. Set-up costs can be saved when maintenance activities are executed simultaneously, since execution of a group of maintenance activities requires only one set-up.

Grouping of maintenance activities can be modelled on the long term (with infinite-horizon models) and on the short term (with finite-horizon models). An infinite horizon is applied in practice as an approximation of a long-term stable situation. This allows one to determine long-term maintenance frequencies for groups of related activities.

In practice however, planning horizons are usually finite for a number of reasons: information is only available on the short term, a modification of the system changes the problem completely, and some events are unpredictable. However, as components mostly have a lifetime that is longer than the length of the horizon, a finite horizon is in practice often applied in a rolling-horizon approach, where the (first) decisions in the finite horizon are implemented and subsequently a new horizon starts, and so on.

For a literature overview of the field of maintenance of multi-component systems, we refer to the review article by Cho and Parlar [3]. By now there are several methods that can handle multiple components. However, most of them suffer from intractability when the number of components grows, unless

[*]Current address: NAM B.V. Assen, the Netherlands

a special structure is assumed. For instance, the maintenance of a deteriorating system is frequently described using Markov decision theory (see, for example, Howard [11], who was the first to use such a problem formulation). Since the state space in such problems grows exponentially with the number of components, the Markov decision modelling of multi-component systems is not tractable for more than three non-identical components (see, for example, Bäckert and Rippin [1]). For problems with many components heuristic methods can be applied. For instance, Dekker and Roelvink [5] present a heuristic replacement criterion in case always a fixed group of components is replaced. Van der Duyn Schouten and Vanneste [13] study structured strategies, viz. (n, N)-strategies, but provide an algorithm for only two identical components. Also worth mentioning are Goyal and Kusy [10] and Goyal and Gunasekaran [9]; they allow many components, but with a very specific deterioration structure only. All these approaches are for an infinite horizon and they provide simple strategies that are difficult to adapt to short-term information.

Stinson and Khumawala [12] consider a finite planning horizon and formulate their problem as a mixed-integer nonlinear programming problem. However, such an approach does not give any structure results and hence little insight. The optimal policy is often quite complex and time dependent. Moreover, it is usually not robust against changes in the horizon length. Another example of short-term planning can be found in the model-based approach for road maintenance by Worm and Van Harten [17]. Roads can be split up into lanes, and lanes into segments of a certain length. Doing the same action on adjacent lanes or segments yields set-up savings. The authors first apply an infinite-horizon single-component model to determine the best optimal action for each lane segment, and subsequently they apply a very simple rule for the grouping of maintenance. For given actions in a certain year, the authors consider whether it is profitable to execute the same action on other lane segments.

In this paper we present a general approach that can assist a maintenance manager in making a short-term maintenance plan based on long-term maintenance strategies. To this end we apply a decomposition and we determine for each activity separately how much it costs to deviate from its tentatively planned maintenance time. Subsequently, we determine the best way of grouping the activities in a given planning horizon. We present an algorithm that leaves enough flexibility to adapt the plan according to requirements set by the maintenance manager. The need of such a short-term planning approach appeared useful after the development of a decision-support system for maintenance at the Shell research laboratory in Amsterdam.

The application of decomposition and a subsequent dynamic grouping allows us to use several models for the individual components, such as minimal repair and block replacement. Besides, our approach yields a dynamic policy that cannot only be applied on the short term but also on the long term. In Dekker, Wildeman and Van Egmond [7] the approach of this paper is numerically validated and extended to age-replacement policies for a discrete-time Markov decision chain, both for an infinite and a finite horizon. The performance of the approach turned out to be very good: the deviation from the optimal costs is less than one per cent provided that certain harmonisation effects are incorporated. This is an important result since -as we already stated- the current Markov decision models allow only few components. A great advantage of our approach is that many components can be handled.

This paper is structured as follows. In the next section we give the problem formulation and we outline a general five-phase approach to solve it. In Section 3 we deal with each of these five phases successively and throughout the paper we use the same example to illustrate each phase when it is discussed. Section 4 briefly discusses some advantages of our approach and in Section 5 we draw conclusions.

2 Problem Definition and Outline of Approach

Consider a multi-component system with n components i, $i \in \{1, \ldots, n\}$. On each component i a preventive-maintenance activity i can be carried out. (We assume in this paper that there is one preventive-maintenance activity for each component, but the modelling is easily extended to deal with more activities per component.) Preventive-maintenance activity i has a component-dependent cost c_i^p and a system-dependent cost S. This system-dependent cost S is called the set-up cost and is the same for all activities. When activity i is executed on its own $c_i^p + S$ has to be paid, whereas for a group $G \subseteq \{1, \ldots, n\}$ of activities that are executed together the set-up cost has to be paid only once, that is, we then have to pay $S + \sum_{i \in G} c_i^p$. This implies that combining m activities yields a cost reduction of $(m-1)S$ compared to executing these m activities separately. A fixed set-up cost is not an uncommon assumption as it is due to, for example, crew travelling, scaffolding, shutdown, etc., which is assumed to be the same for all activities. Another practical motivation is that it is very hard to obtain more specific data; no present-day management information system supports a data structure for each possible combination of activities. Van Dijkhuizen and Van Harten [14] present an approach in which multiple levels of set-up costs are considered. However, these authors consider an infinite horizon and hence do not incorporate short-term circumstances.

Components $1, \ldots, n$ are not used at a constant rate; on the short term the use of a component can vary because of, for example, a varying demand (in case of a production system). Furthermore, an opportunity may occur because of a shutdown of the system for whatever reason; in that case no set-up cost has to be paid for activities that are carried out at that time.

Opportunities and a varying use of components are typical short-term circumstances that cannot be incorporated in long-term (infinite-horizon) maintenance optimisation models. Short-term circumstances can be incorporated in finite-horizon models, but this has other disadvantages. Finite horizons are often much shorter than the lifetime of a component, which implies that we have to introduce a residual value for each component at the end of the horizon. This can cause very capricious finite-horizon effects that depend on the length of the horizon and the definition of the residual values (see, for example, Dekker, Wildeman and Van Egmond [7]). In practice one would like to have a situation in which a small change in the horizon causes small changes in the generated solution.

In this paper we present a five-phase rolling-horizon approach that has a more stable character because short-term plans are made on the basis of a long-term tentative plan. The approach has many other advantages, some of which will be discussed in Section 4. Here we first give an outline of the approach.

Phase 1: Decomposition

Formulate an infinite-horizon maintenance model for each activity separately and optimise this model to obtain an optimal frequency with respect to the long-term average costs. Here we assume an average use of components and we neglect or approximate global interactions between components. The result is an individual maintenance rule for each activity. Usually, this phase has to be done only once.

Phase 2: Penalty Functions

We then derive a *penalty function* $h_i(\Delta t)$ for each activity i, expressing the additional expected costs of shifting the execution time of activity i Δt time units from a tentatively planned time. This shift Δt may be positive and negative (forward and backward in time). The penalty functions are derived from the individual maintenance models in Phase 1 and usually this needs to be done only once.

Phase 3: Tentative Planning

Suppose the system is considered at a certain time t. Based on the individual maintenance rules of Phase 1, the current state of component i and short-term information, the time t_i is determined at which activity i is carried out when it were on its own. Now take a finite planning horizon and consider the activities to be executed in this horizon. Here we choose the horizon $[t, \max_i t_i]$ induced by the tentative execution times t_i, $i = 1, \ldots, n$, but other choices are also possible. In this phase opportunities can be incorporated.

Phase 4: Grouping Maintenance Activities

In this phase it is allowed to shift the tentatively planned times within the planning horizon $[t, \max_i t_i]$ to make joint execution of activities possible. The optimal grouping structure of the n activities maximises the set-up cost reduction (because of joint maintenance) minus the costs of shifting from the tentatively planned times. The latter costs are expressed by the penalty functions derived in Phase 2. In Section 3.4 we show that under general conditions an optimal grouping can be found in $\mathcal{O}(n^2)$ time.

Phase 5: Rolling-Horizon Step

Having applied Phase 4 we have a grouping structure for the n activities in $[t, \max_i t_i]$. The maintenance manager can change the planning in case he/she is not satisfied with it and then go back to Phase 3; this can be done interactively and as often as desired. Finally, the maintenance manager can carry out one or more groups of activities according to the generated grouping structure and start with Phase 3 when a planning for a new period is required.

3 Rolling-Horizon Approach

In this section we will discuss each phase of our rolling-horizon approach in detail and of each phase we will give an example. We want to stress here that the examples are indeed only for illustrative reasons; they do not show all possible aspects and extensions of our approach. Some of these aspects and extensions are discussed in Section 4, but showing all would go far beyond the scope of this paper.

3.1 Phase 1: Decomposition

For each individual activity i we formulate an infinite-horizon maintenance model in which we assume an average use of component i and in which we neglect or approximate global interactions between components.

In this paper we consider the following kind of model. Activity i is a preventive replacement of component i and is executed each x time units at a cost of $c_i^p + S$. Let $M_i(x)$ denote the expected deterioration costs for component i, i.e., the expected costs (because of repairs, or increasing energy use) incurred in x time units since the latest execution of activity i. It is assumed that $M_i(\cdot)$ is strictly convex.

Long-term optimisation of the interval between executions of activity i is easily achieved by applying renewal theory. We find that the long-term average costs $\Phi_i(x)$ as a function of the interval length x equals (see, for example, Dekker [4])

$$\Phi_i(x) = \frac{c_i^p + S + M_i(x)}{x}.$$

Let x_i^* denote the optimal interval length (if it exists) and $\Phi_i^* := \Phi_i(x_i^*)$ the associated long-term average costs. Using $\Phi_i'(x) = 0 \Leftrightarrow M_i'(x) - \Phi_i(x) = 0$ and the fact that $M_i'(\cdot)$ is strictly increasing, it is not difficult to show that this x_i^* exists if, e.g., $\lim_{x\to\infty} M_i'(x) = \infty$. Furthermore, x_i^* is then unique and the following holds: $M_i'(x) - \Phi_i^* < 0$ for $x < x_i^*$ and $M_i'(x) - \Phi_i^* > 0$ for $x > x_i^*$. Strategies like block replacement and the version of minimal repair that we use in this paper are captured by this type of maintenance optimisation model (see Dekker [4] for a list of models that can be captured). Age replacement is not included in this model, but can be incorporated approximately in an extension (see Dekker, Wildeman and Van Egmond [7]).

In this model we neglect the economic dependence between components since in $\Phi_i(\cdot)$ the set-up cost S is attributed wholly to component i, assuming that activity i is always executed on its own. The modelling is easily extended to deal with harmonisation of frequencies, for example by applying a correction factor (see Dekker, Wildeman and Van Egmond [7]). However, that does not change the concept of decomposition and will thus not be considered here.

Example

We consider $n = 16$ maintenance activities that are modelled according to a minimal-repair model with block replacements. In this minimal-repair model, preventive replacements are carried out after fixed intervals, with failure repair occurring whenever necessary. A failure repair restores the component involved into a state as good as before. No combination of repair and replacement is possible in this variant. Activity i, $i = 1, \ldots, 16$, is the preventive replacement of component i and costs $c_i^p + S$. The failure-repair cost of component i equals c_i^r. Let $r_i(\cdot)$ denote the rate of occurrence of failures, then $c_i^r \int_0^x r_i(y)dy$ are the expected repair costs incurred in the interval $[0, x]$ due to failures (see, for example, Dekker [4]). For $r_i(\cdot)$ we take a polynomial rate of occurrence of failures with scale parameter $\lambda_i > 0$ (in days), and shape parameter $\beta_i > 1$ (this is a Weibull process):

$$r_i(x) = \frac{\beta_i}{\lambda_i}\left(\frac{x}{\lambda_i}\right)^{\beta_i - 1}. \tag{1}$$

For the scale parameter λ_i we took as time unit a day, since the aim is to plan the activities in absolute time and not, for example, in machine running time. However, if running hours are considered, the parameter is easily converted to days through multiplication by a utilisation factor.

The expression of $r_i(\cdot)$ above yields the following expression for the expected deterioration costs $M_i(x)$ (in this case repair costs) for component i in $[0, x]$:

$$M_i(x) = c_i^r \left(\frac{x}{\lambda_i}\right)^{\beta_i}. \tag{2}$$

The optimal value x_i^* can be found by setting the derivative of $\Phi_i(\cdot)$ to zero. For a minimal-repair model we thus find the following analytical expression for x_i^*:

$$x_i^* = \sqrt[\beta_i]{\frac{(c_i^p + S)\lambda_i^{\beta_i}}{c_i^r(\beta_i - 1)}}.$$

Since $\Phi_i^* = \Phi_i(x_i^*) = M_i'(x_i^*)$, we have for Φ_i^*:

$$\Phi_i^* = \frac{(c_i^p + S)\beta_i}{x_i^*(\beta_i - 1)}.$$

In Table I we give the random data for the 16 maintenance activities that we use throughout this paper. For the set-up cost S we take $S = 15$, which is 5% of the average *individual* preventive-maintenance costs, i.e., $S = 0.05(1/16)\sum_{i=1}^{16}(c_i^p + S)$. The values of x_i^* and Φ_i^*, obtained by substitution of the data and $S = 15$ in the two equations above, are also given given in Table I.

Table I

Example data for 16 activities and the corresponding optimal values of Φ_i^* and x_i^*

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
λ_i	159	159	190	285	108	285	49	97	84	149	190	117	205	281	281	285
β_i	1.70	1.70	2.00	2.00	1.70	2.00	1.25	1.75	1.50	1.50	2.00	1.70	1.75	1.75	1.75	2.00
c_i^p	105	225	345	165	585	345	105	345	345	45	345	885	225	105	105	225
c_i^r	92	182	28	30	172	30	90	50	76	12	28	66	36	22	22	30
Φ_i^*	1.27	2.53	1.06	0.52	5.25	0.73	3.21	2.38	2.87	0.26	1.06	3.26	0.78	0.32	0.32	0.60
x_i^*	229	231	681	698	278	987	187	353	376	692	681	671	714	873	873	806

3.2 Phase 2: Penalty Functions

The individual maintenance rule of activity i according to Phase 1 provides a frequency at which activity i is executed. In Phase 3 a tentative planning is made based on this frequency and this tentative planning can be changed to enable grouping of activities in Phase 4. To this end we need to calculate the costs of shifting the execution time of an activity from its tentatively planned time. This will be done here.

According to the maintenance rule chosen in Phase 1, activity i is executed each x_i^* time units. Let t_i be the tentatively planned execution time of activity i, found by adding x_i^* time units to the latest execution date of the activity. We distinguish between two cases to determine the penalty costs $h_i(\Delta t)$ for a shift Δt from time t_i (where Δt may be positive or negative). In the first case, called Long-Term Shift (LTS), the execution interval is changed once, from x_i^* to $x_i^* + \Delta t$, while all future intervals remain x_i^*. This implies that all future execution *times* (after t_i) are shifted by Δt time units. In the latter case, called Short-Term Shift (STS), the execution interval is changed twice, first from x_i^* to $x_i^* + \Delta t$, while the following interval equals $x_i^* - \Delta t$. This implies that all future execution *times* (after t_i) remain the same.

In case of an LTS, the deterioration costs in the first interval (of length $x_i^* + \Delta t$) are given by $M_i(x_i^* + \Delta t)$, whereas otherwise $M_i(x_i^*)$ was paid. So the *extra* expected deterioration costs as a result of a shift Δt are given by $M_i(x_i^* + \Delta t) - M_i(x_i^*)$. As all following intervals remain of length x_i^*, all future executions times after t_i are deferred by Δt time units, which saves $\Delta t \Phi_i^*$. Altogether, the penalty function equals

$$h_i(\Delta t) = M_i(x_i^* + \Delta t) - M_i(x_i^*) - \Delta t \Phi_i^*, \qquad \Delta t > -x_i^*. \tag{3}$$

Notice that $h_i(\cdot)$ is strictly convex ($h_i''(\cdot) > 0$ since $M_i(\cdot)$ is strictly convex) and $h_i(0) = 0$. As $M_i'(x) - \Phi_i^* < 0$ for $x < x_i^*$ and $M_i'(x) - \Phi_i^* > 0$ for $x > x_i^*$, and $h_i'(\Delta t) = M_i'(x_i^* + \Delta t) - \Phi_i^*$, it follows that $h_i'(\Delta t) < 0$ for $\Delta t < 0$ and $h_i'(\Delta t) > 0$ for $\Delta t > 0$. Together with $h_i(0) = 0$ this implies that $h_i(\cdot) \geq 0$.

In case of an STS, the deterioration costs in the first two intervals (of length $x_i^* + \Delta t$ and $x_i^* - \Delta t$, respectively) are given by $M_i(x_i^* + \Delta t) + M_i(x_i^* - \Delta t)$, whereas otherwise in each of the two first intervals $M_i(x_i^*)$ was paid. As all future execution times after t_i remain unchanged, the penalty costs as a result of a shift Δt equal the *extra* expected deterioration costs, so that

$$h_i(\Delta t) = M_i(x_i^* + \Delta t) + M_i(x_i^* - \Delta t) - 2M_i(x_i^*), \qquad -x_i^* < \Delta t < x_i^*. \tag{4}$$

Notice that this function is also strictly convex ($h_i''(\cdot) > 0$), that $h_i(0) = 0$ and that $h_i(\cdot) \geq 0$. It even holds that $h_i(\cdot)$ is symmetric around zero.

Example

When we substitute (2) in (3) and (4), we obtain the following expressions for the penalty functions $h_i(\cdot)$. For an LTS we have:

$$h_i(\Delta t) = c_i^r \left(\frac{x_i^* + \Delta t}{\lambda_i}\right)^{\beta_i} - c_i^r \left(\frac{x_i^*}{\lambda_i}\right)^{\beta_i} - \Delta t \Phi_i^*, \qquad \Delta t > -x_i^*, \tag{5}$$

and for an STS:

$$h_i(\Delta t) = c_i^r \left(\frac{x_i^* + \Delta t}{\lambda_i}\right)^{\beta_i} + c_i^r \left(\frac{x_i^* - \Delta t}{\lambda_i}\right)^{\beta_i} - 2c_i^r \left(\frac{x_i^*}{\lambda_i}\right)^{\beta_i}, \qquad -x_i^* < \Delta t < x_i^*.$$

As $r_i(\cdot)$ is strictly increasing, $M_i(\cdot)$ is strictly convex. This implies that $h_i(\cdot)$ is strictly convex as well, both for an LTS and an STS. In case of an STS, $h_i(\cdot)$ is also symmetric around zero. In Figure 1 the penalty function of activity 1 is given for an LTS.

3.3 Phase 3: Tentative Planning

In this phase a tentative planning for each activity i is made at a certain time t; opportunities are incorporated (if any) and based on the individual maintenance rules of Phase 1 we determine at what time t_i activity i should be carried out. Under average conditions, this is x_i^* time units after the latest execution date of activity i. However, because of a varying use of component i this can be at another time, depending on the utilisation rate of component i since the latest execution of activity i.

We now take a finite planning horizon and we consider the activities that have to be executed in this horizon. There are many possibilities to choose a planning horizon and the choice depends very much on the specific problem and the requirements of the maintenance manager. Here we choose the horizon $[t, \max_i t_i]$ induced by the tentative execution times t_i, $i = 1, \ldots, n$, and we consider only the first occurrence of each activity. This implies that for each component there is exactly one planned activity. (Extending the model to deal with multiple occurrences of an activity in the planning horizon is obvious, but will not be done here.)

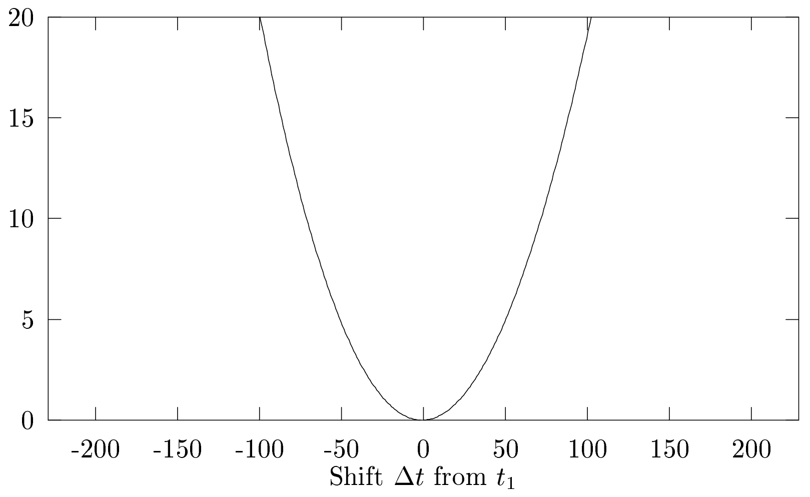

Figure 1. The penalty function $h_1(\cdot)$ of activity 1 according to an LTS (see equation (5)). Notice that $h_1(\cdot) \geq 0$, that $h_1(0) = 0$ and that $h_1(\cdot)$ is strictly convex.

Example

Suppose the system is considered at time (day) $t = 0$. Activity i is carried out each x_i^* days (see Table I). The time unit of these values x_i^* is a day, since the scale parameter λ_i is in days. As an example of how to incorporate certain short-term circumstances, we suppose here that the values of the scale parameter λ_i are based on an operating time of component i in *running hours* that are converted to days. Suppose the average utilisation factor is a_i hours a day. In practice however, utilisation factors fluctuate in time. We will show here how this can be incorporated in a short-term planning.

Suppose that at day $t = 0$ the latest execution time of activity i is x_i days ago and that during that period component i has been used for u_i hours a day. Consequently, at time $t = 0$ it is $u_i x_i$ *running hours* since the latest execution of activity i, whereas activity i is normally carried out each $a_i x_i^*$ running hours. This implies that the new execution time of activity i is $a_i x_i^* - u_i x_i$ running hours from now. Assuming that in the coming period activity i will again be used u_i hours a day (but this may also be another value), we have the following expression for the tentative execution day t_i of activity i:

$$t_i = \frac{1}{u_i}\left\{a_i x_i^* - u_i x_i\right\} = \frac{a_i}{u_i} x_i^* - x_i.$$

As an example, let the values of a_i, u_i and x_i for the sixteen activities be as given in Table II. The values of t_i are then easily calculated and are also tabulated in Table II (for completeness we repeat the values of x_i^* given in Table I). Notice that the components are indexed such that $t_1 \leq t_2 \leq \cdots \leq t_n$.

Table II

Values of a_i, u_i, x_i^*, x_i and t_i

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a_i	20	18	12	12	14	8	22	10	12	8	7	7	9	11	11	13
u_i	13	12	15	12	7	8	19	13	11	11	16	9	6	11	12	12
x_i^*	229	231	681	698	278	987	187	353	376	692	681	671	714	873	873	806
x_i	352	344	513	661	476	902	129	172	299	366	130	345	887	678	585	656
t_i	0	3	32	37	80	85	88	100	111	137	168	177	184	195	215	217

We want to stress here again that this way of incorporating certain short-term circumstances is only to illustrate the possibilities of our approach, not the limitations. What we present here is a general approach; we can apply many extensions or make alternative choices whenever a detail of our approach has to be filled in, but this will only blur the global picture. The details depend very much on the specific problem and they do not change the concept of our approach.

3.4 Phase 4: Grouping Maintenance Activities

Without loss of generality assume that the n activities that are tentatively planned at execution times t_i, $i = 1, \dots, n$, in Phase 3 are indexed such that $t_1 \leq t_2 \leq \cdots \leq t_n$ (notice that this order can be obtained by application of standard sorting algorithms with a time complexity of $\mathcal{O}(n \log n)$). Consequently, the planning horizon equals $[t, t_n]$.

In this subsection we first give a mathematical formulation of the problem of grouping the n activities in the finite planning horizon $[t, t_n]$. Subsequently, we derive reduction theorems that make the problem more tractable. Finally, based on these reduction theorems, we present an efficient dynamic-programming algorithm to solve the problem in polynomial time.

Mathematical Formulation of Grouping Problem

A *group* of activities is a subset of $\{1, \dots, n\}$. A *partition* of $\{1, \dots, n\}$ is a collection of mutually exclusive groups $G_1, \dots, G_m$, which cover all activities, i.e., $G_j \cap G_k = \emptyset \; \forall \; j \neq k$, and $G_1 \cup \cdots \cup G_m = \{1, \dots, n\}$. A *grouping structure* is a partition of $\{1, \dots, n\}$ such that all activities within each group are jointly executed at the same time.

Given for each activity i is its penalty function $h_i(\Delta t)$ expressing the additional expected costs of shifting the execution time of activity i Δt time units from its tentatively planned time t_i. This shift Δt may be positive and negative (forward and backward in time). These penalty functions were derived in Phase 2. Here we assume the following with respect to the penalty functions $h_i(\cdot)$:

Assumption 1 $h_i(\cdot)$ is strictly convex, $i = 1, \dots, n$.

Assumption 2 $h_i(\cdot) \geq 0$, $i = 1, \dots, n$.

Assumption 3 $h_i(0) = 0$, $i = 1, \dots, n$.

Notice that the penalty functions derived in Phase 2 satisfy these assumptions. Assumption 1 is more restrictive than necessary. In fact, in the sequel we only need to assume that every sum of penalty functions has a unique minimum, and that this sum is decreasing left of its minimum and increasing right of it. However, for reasons of simplicity, we use Assumption 1.

For any group G of activities, $H_G(t) := \sum_{i \in G} h_i(t - t_i)$ is the *cumulative penalty function*, expressing the additional expected costs when executing group G at time t. The group G should be executed when $H_G(\cdot)$ is minimal, say at time t_G^*, which we call the optimal execution time of group G. Let H_G^* denote the minimal value of $H_G(\cdot)$, then $H_G^* = H_G(t_G^*)$.

Each activity has the same set-up cost S, which implies that combining the execution of m activities yields a cost reduction of $(m - 1)S$. If for a group G the set-up cost reduction $(|G| - 1)S$ is larger than or equal to the minimal value H_G^*, G is called *cost-effective*. Equivalently, in this case we say that the *savings* are greater than or equal to zero, where the savings are defined as $(|G| - 1)S - H_G^*$, the set-up cost reduction minus the sum of the penalty costs.

An optimal grouping structure (partition) can be obtained by solving a set-partitioning problem (see also Dekker, Smit and Losekoot [6]), where the objective is to maximise the total savings (that is, the sum of the savings of the groups in the grouping structure). However, set partitioning is NP-complete (see Garey and Johnson [8]) and with n the number of activities, there are $2^n - 1$ possible groups (ignoring the empty set). In some cases groups can be excluded beforehand, which decreases the size of the set-partitioning problem. This is possible by using the reduction theorems presented below. To this end we need the following definition.

Define intervals I_i, $i = 1, \dots, n$, such that $I_i = [t_i + \Delta t_i^-, t_i + \Delta t_i^+]$, with Δt_i^- the smallest and Δt_i^+ the largest solution of the equation $h_i(\Delta t) - S = 0$. The interval I_i clearly shows the maximum allowable shift (backward and forward) of the execution time of activity i, to form a group with one of the other activities such that the penalty costs of activity i do not exceed the reduction in set-up costs.

Problem Reduction

We will now present theorems that reduce the number of groups that have to be considered for identifying an optimal grouping structure. The first two originate from Dekker, Smit and Losekoot [6]. The proofs of the other theorems are given in Appendix A.

If for two activities i and j the intersection of the corresponding intervals is empty (i.e., $I_i \cap I_j = \emptyset$), then i and j cannot be combined cost-effectively, since the penalty costs are greater than the reduction in set-up costs. Therefore, the group $\{i, j\}$ cannot be part of an optimal grouping structure. If for more than two activities the corresponding intersection of intervals is empty, combining these activities *can* be cost-effective but never optimal. This is stated in Theorem 1.

Theorem 1 *A group G can only be part of an optimal grouping structure if $\cap_{i\in G} I_i \neq \emptyset$. Furthermore, the optimal execution time t_G^* of G is in $\cap_{i\in G} I_i$.*

We say that a group F contains a *cluster* $G \subset F$, if $\forall\ i \in F \setminus G$ the following holds: $i < \min_{j\in G} j$ or $i > \max_{j\in G} j$. For instance, {3,5} and {3,5,7} are clusters in {1,3,5,7,8}, whereas {3,7} and {3,5,8} are not.

Theorem 2 *A group F cannot be part of an optimal grouping structure if it contains a cluster G that can be split up more effectively into two clusters of activities.*

There is some redundancy between the results of Theorems 1 and 2. If $\cap_{i\in G} I_i = \emptyset$ for a group G, then it can be proved, analogously to Theorem 1, that G can be split up more effectively into two clusters of activities. According to Theorem 2 this implies that G cannot be part of an optimal grouping structure. A group excluded by Theorem 1 will thus also be excluded by Theorem 2. We will, however, still use Theorem 1, since it provides a better understanding of the problem and it will appear to be useful for the properties defined below and for the dynamic-programming algorithm presented later.

Dekker, Smit and Losekoot [6] present an algorithm that first eliminates groups that cannot be part of an optimal grouping structure according to Theorems 1 and 2, and that subsequently finds the best structure by applying set partitioning to the remaining groups. A great disadvantage of this method is that its time complexity strongly depends on the data, since the number of groups excluded by Theorems 1 and 2 are dependent on the data. Theorems 1 and 2 might not eliminate any group at all, which implies that the number of groups left for the set-partitioning problem can be $\mathcal{O}(2^n)$.

In case the penalty functions are of a special form, however, the grouping problem has a very nice structure. The total number of groups can be reduced to $\mathcal{O}(n^2)$ if there is an optimal grouping structure in which every group has *consecutive* activities. A group G is said to contain consecutive activities if G is a cluster in $\{1, \ldots, n\}$. This very important result will be established in Theorem 3, assuming one of the following properties. (A similar consecutiveness result is obtained by Van Dijkhuizen and Van Harten [14], but for a different problem and in an infinite horizon.)

Property 1 (Symmetry) *All $h_i(\cdot)$ are symmetric, i.e., $h_i(\Delta t) = h_i(-\Delta t)$, $i = 1, \ldots, n$, $\forall\ \Delta t > 0$.*

Property 2 (Congruency) *All $h_i(\cdot)$ are congruent, i.e., $\alpha_i h_i(\cdot) = h_1(\cdot)$, $\alpha_i > 0$, $i = 2, \ldots, n$.*

Property 3 (Dominance) *For all $i = 1, \ldots, n-1$ $h_i(\cdot)$ dominates $h_{i+1}(\cdot)$ right of t_{i+1}, i.e., $h_i(t_{i+1} - t_i + \Delta t) > h_{i+1}(\Delta t)$ for $\Delta t > 0$, and for all $i = 2, \ldots, n$ $h_i(\cdot)$ dominates $h_{i-1}(\cdot)$ left of t_{i-1}, i.e., $h_i(t_{i-1} - t_i - \Delta t) > h_{i-1}(-\Delta t)$ for $\Delta t > 0$.*

Notice that these properties need only be true for each penalty function $h_i(\cdot)$ on its interval I_i. This is due to the fact that each activity i must be executed within its corresponding interval I_i. Any group G that contains an activity i such that the optimal execution time t_G^* of G is not in I_i cannot be part of an optimal grouping structure according to Theorem 1.

Theorem 3 (Consecutive Activities) *If Property 1, 2 or 3 holds there exists an optimal grouping structure with consecutive activities.*

In the proof of Theorem 3 (see Appendix A) it is shown that if one of the properties holds for two activities i and j it is optimal to execute these activities in the order of their tentatively planned execution times. This is independent of any activity other than i and j, so that the following corollary holds.

Corollary 1 *If Property 1, 2 or 3 holds for all activities in a subset A of $\{1, \dots, n\}$, then it is optimal to execute the activities in A in consecutive order (that is, in the order as tentatively planned).*

This also implies that adding whatever activities to the problem does not change the order of execution of the activities in A.

Due to Theorem 3, only groups with consecutive activities need to be considered. This implies that the maximum number of groups to be considered reduces from $2^n - 1$ to $(1/2)n(n+1)$, as there are 1 group of n consecutive activities, 2 groups of $n-1$ consecutive activities, ..., and n groups of 1 (consecutive) activity.

It is still an open question whether there are more properties that guarantee the existence of an optimal grouping structure with consecutive activities. In Appendix A it is shown that if none of the properties holds, there may be a unique optimal grouping structure in which the activities are not consecutive.

The following theorem also provides a reduction of the number of groups to be considered; for this theorem we implicitly assume that Property 1, 2 or 3 holds, so that Theorem 3 applies.

Theorem 4 *If in an optimal grouping structure of the first s activities activity s is executed in another group than an activity p $(1 \leq p < s)$, then for any $r > s$ there is an optimal grouping structure of the first r activities in which activity s is also executed in another group than activity p.*

If we take $s = p + 1$ in Theorem 4 we have the following corollary:

Corollary 2 *If in an optimal grouping structure of the first $p+1$ activities activity $p+1$ is executed on its own, then for any $r > p+1$ there is an optimal grouping structure of the first r activities in which $p+1$ is also executed separately from the activities $1, \dots, p$. Consequently, the groups found in an optimal grouping structure of the first p activities are then also optimal in any optimal grouping structure of the first r activities.*

This implies that if it is optimal for the first $p+1$ activities to execute activity $p+1$ on its own, adding activities i with $t_i \geq t_{p+1}$ does not influence the grouping of activities $1, \dots, p$. In that case the grouping structure of activities $1, \dots, p$ is optimal independently of the length of the time horizon, as long as no activity with tentative execution time smaller than t_{p+1} is added. This result is important since it implies that we can decide that a certain solution is stable, so that finite-horizon effects are eliminated. Such results cannot always be obtained for finite-horizon models. In inventory theory, these stability results are usually referred to as *planning-horizon* or *turnpike theorems* (see, for example, Wagner [15]).

The next corollary is a more complicated application of Theorem 4 and provides another result with respect to the stability of the grouping structure in the finite planning horizon.

Corollary 3 *Suppose that for $i = p-1, \dots, q-1$ group $\{l, \dots, i\}$ $(l \leq p-1 \leq q-1)$ is part of an optimal grouping structure GS_i of the first i activities and group $\{p, \dots, q\}$ is part of an optimal grouping structure GS_q of the first q activities. For any $r > q$ there is an optimal grouping structure GS_r of the first r activities, in which the activities that are combined in GS_{p-1} are also combined in GS_r.*

Notice that Corollary 3 immediately follows from Corollary 2 for the case $p = q$.

Dynamic-Programming Algorithm

If Property 1, 2 or 3 holds, there is by Theorem 3 an optimal grouping structure in which the activities in each group are consecutive. Every partitioning problem with an optimal solution that is consecutive can be solved following a shortest-path approach (Chakravarty, Orlin and Rothblum [2]). This yields a solution in $\mathcal{O}(n^2)$ time. However, in order to make use of Theorems 1, 2 and 4 as well, we shall present a special algorithm that uses the principle of dynamic programming. This algorithm terminates after n

iterations, while in each iteration j a best group with last activity j is found. The array entry $\mathit{First}[j]$ indicates the first activity of this best group. That is, if $\mathit{First}[j] = i$, then $\{i, \ldots, j\}$ is the best group found in iteration j. The total savings of the corresponding optimal grouping structure is stored in the array entry $\mathit{TotalSavings}[j]$. Thus we have the following approach.

Initialisation: $\mathit{TotalSavings}[0] := 0$.

Iteration 1: The best group with last activity 1 is $\{1\}$, with corresponding optimal grouping structure $\{1\}$. $\mathit{First}[1] := 1$. $\mathit{TotalSavings}[1] := 0$.

FOR $j := 2$ TO n DO Iteration j: Consider the groups with last activity j in the following order: $\{j\}$, $\{j-1, j\}$, ..., $\{1, \ldots, j\}$. Find the group for which the corresponding grouping structure covering activities $1, \ldots, j$ has greatest savings. This is the group $\{i, \ldots, j\}$ for which $\mathit{TotalSavings}[i-1]$ + *savings of* $\{i, \ldots, j\}$ is maximal. $\mathit{First}[j] := i$. $\mathit{TotalSavings}[j] := \mathit{TotalSavings}[i-1]$ + *savings of* $\{i, \ldots, j\}$.

The best grouping structure can be found by backtracking. The corresponding total savings equal $\mathit{TotalSavings}[n]$.

In Appendix B it is shown how Theorems 1, 2 and 4 can be incorporated in this approach and we present the resulting dynamic-programming algorithm. This dynamic-programming algorithm has in the worst case a time complexity of $\mathcal{O}(n^2)$, whereas in the best case an optimal grouping structure is found in linear time.

Example

In Table II we have the tentative execution times t_i, $i = 1, \ldots, 16$, within the planning horizon $[t, t_n] = [0, 217]$. To find the optimal grouping structure of the activities, we can apply the dynamic-programming algorithm. This is optimal when an optimal grouping structure with consecutive activities exists. When using the minimal-repair model as described before, there are different situations in which this is the case.

1. As $h_i(\cdot)$ is symmetric around zero if we apply an STS, Property 1 holds if each penalty function is derived according to an STS.

2. All β_i are equal to 2. It is easily shown that if $\beta_i = 2$, $h_i(\cdot)$ is symmetric also when an LTS is applied.

3. All components are equal. In that case the penalty functions are equal and, a forteriori, congruent, which implies that Property 2 holds. It is easily shown that the same result is true when all c_i^p and all c_i^r differ a positive factor α_i (i.e., $\alpha_i c_i^r = c_1^r$ and $\alpha_i c_i^p = c_1^p$, $\alpha_i > 0$, $i = 2, \ldots, n$) and all other parameters are equal.

4. Property 3 holds. Here it is convenient to recall the definition of the intervals I_i in the beginning of this section: $I_i = [t_i + \Delta t_i^-, t_i + \Delta t_i^+]$, with Δt_i^- the smallest and Δt_i^+ the largest solution of the equation $h_i(\Delta t) - S = 0$. Due to Theorem 1, Property 3 needs only be true for each penalty function $h_i(\cdot)$ on its interval I_i. This implies that we can check for each $h_i(\cdot)$ whether it intersects $h_{i+1}(\cdot)$ on the interval $[t_{i+1}, t_i + \Delta t_i^+]$, or $h_{i-1}(\cdot)$ on the interval $[t_i + \Delta t_i^-, t_{i-1}]$. If this is not so, then Property 3 holds.

If none of these situations occurs, there can still be a consecutive solution. In Appendix C it is shown which strategies can be followed to check the existence of a consecutive solution. When the existence of a consecutive optimal solution cannot be proved, there are several other possibilities (see Appendix C), one of which is to apply the dynamic-programming algorithm as a heuristic. In Appendix C it is shown that the optimal total savings are always less than twice as large as the total savings of the solution found with dynamic programming. This implies that application of the dynamic-programming algorithm yields a solution that in the worst case is almost twice as bad as the optimal total savings. However, to obtain such a worst case, we have to define very pathetic penalty functions that are not likely to occur in practice. Fortunately, the dynamic-programming algorithm can also be used to obtain an upper bound that is mostly much better, using a technique described in Appendix C.

We will illustrate here how dynamic programming can be used as a heuristic. To do so, we choose an example that is as general as possible, but such that none of the properties holds for all activities (otherwise the dynamic-programming algorithm is optimal). Consequently, we will not take the penalty functions according to an STS, since these are symmetric so that Property 1 holds. Instead, we take for each of the 16 activities tentatively planned in $[0, 217]$ a penalty function according to an LTS (see (5)). In this case none of the properties holds for all activities. Application of the technique in Appendix C to obtain an upper bound on the optimal total savings yields that the optimal total savings cannot be larger than 191.26.

When we apply the dynamic-programming algorithm (as a heuristic) to the data of Table I, we find that the total savings equal 191.24. Consequently, we can conclude that this solution -if not optimal- is at least very good (the maximum deviation from the optimal value is 0.01 %). The dynamic-programming algorithm found for each iteration j a (best) group with last activity j, as given in Table III.

The grouping structure given by the algorithm is found with backtracking. In the last iteration (iteration 16) group {10,11,12,13,14,15,16} is optimal. To cover the first 9 activities it is optimal in iteration 9 to take group {5,6,7,8,9}. Finally, in iteration 4 group {1,2,3,4} is optimal. The corresponding grouping structure is given in Table IV. The reduction in set-up costs is $13 \times 15 = 195$ (there are only three groups, and consequently three set-ups), whereas the total penalty costs amount to 3.76. Consequently, the total savings are 191.24, which is about 4.03 % of the total individual preventive maintenance costs $\sum_{i=1}^{16}(c_i^p + S) = 4740$. Applying Theorem 3 reduced the total number of groups from $2^n - 1 = 2^{16} - 1 = 65535$ to $(1/2)n(n+1) = (1/2)16(16+1) = 136$. Through application of Theorems 1, 2 and 4, 72 of these 136 groups needed to be investigated. Altogether it took 0.11 seconds of CPU time on a 66 Mhz PC-AT to identify the grouping structure.

Table III
The best group in each iteration

Iteration	Best group in this iteration	Savings of this group	Total savings
1	{1}	0.00	0.00
2	{1,2}	14.99	14.99
3	{1,2,3}	29.37	29.37
4	{1,2,3,4}	44.03	44.03
5	{5}	0.00	44.03
6	{5,6}	14.99	59.02
7	{5,6,7}	29.89	73.92
8	{5,6,7,8}	44.26	88.29
9	{5,6,7,8,9}	58.22	102.25
10	{5,6,7,8,9,10}	73.01	117.04
11	{5,6,7,8,9,10,11}	83.53	127.56
12	{10,11,12}	29.83	132.08
13	{10,11,12,13}	44.79	147.04
14	{10,11,12,13,14}	59.73	161.98
15	{10,11,12,13,14,15}	74.52	176.77
16	{10,11,12,13,14,15,16}	88.99	191.24

Notice that in iteration 5 activity 5 is executed on its own. According to Corollary 2 this implies that activity 5 will never be executed with one of the activities $1, \ldots, 4$. Consequently, the group {1,2,3,4} is optimal in every grouping structure of the first r activities, $r \geq 4$.

Notice also that for $i = 9, 10, 11$, group $\{5, 6, 7, 8, \ldots, i\}$ is optimal in iteration i and, consequently, part of an optimal grouping structure of the first i activities, while in iteration 12 group {10,11,12} is

Table IV
Grouping structure for the data in Table I

Group	Savings	Day
{1,2,3,4}	44.03	7.2
{5,6,7,8,9}	58.22	89.6
{10,11,12,13,14,15,16}	88.99	181.1
Total savings	191.24	

optimal. An (imaginary) activity 17 cannot be combined with activity 9 or lower according to Theorem 4, thus it will be in group $\{i, \ldots, 17\}$, with $10 \leq i \leq 17$. If $i = 10$ then group {5,6,7,8,9} is optimal (see Table III), if $i = 11$ then {5,6,7,8,9,10} is optimal, and if $i \geq 12$ then {5,6,7,8,9,10,11} is optimal. This argument can be repeated for an activity 18 and so on. We thus applied Corollary 3, which implies that there for any $r \geq 16$ it is optimal to group the activities {1,2,3,4} (which we knew already from Corollary 2) and to group the activities {5,6,7,8,9} (possibly with activity 10 and 11 if $r > 16$).

3.5 Phase 5: Rolling-Horizon Step

After application of Phase 4 we have a grouping structure for the n activities within the finite planning horizon $[t, t_n]$. The maintenance manager can change the planning in case he/she is not satisfied with it and then go back to Phase 3 or he/she can carry out one or more groups of activities according to the generated grouping structure and start with Phase 3 when a planning for a new period is required, for example because new information (like an opportunity) becomes available.

Example

Application of Phase 4 yields the grouping structure of the 16 activities within [0,217] as given in Table IV. Below we give examples of how an opportunity can be incorporated and how the maintenance manager can change the planning.

Suppose that an opportunity occurs at $t = 0$, for instance as a result of a failure of the system. This implies that the system is down and that other activities can be carried out at $t = 0$ without paying the set-up cost S. This opportunity can be incorporated in our approach by defining an activity *opp* with tentative execution time $t_{opp} = 0$, and with a penalty function $h_{opp}(\cdot)$ that is infinite for every shift unequal to zero ($h_{opp}(\Delta t) = \infty$ for $\Delta t \neq 0$) and with $h_{opp}(0) = 0$. When we include this opportunity in our example, we obtain the same structure as in Table IV, except that activities 1,2,3,4 are now executed at time $t = 0$ instead of at time $t = 7.2$. This involves larger penalty costs ($H_{\{1,2,3,4\}}(0) = 1.33$ whereas $H_{\{1,2,3,4\}}(7.2) = 0.97$), but we obtain extra savings of 15 cost units since now for group {1,2,3,4} no set-up cost has to be paid. Consequently, the savings of group {1,2,3,4} are now $4S - H_{\{1,2,3,4\}}(0) = 60 - 1.33 = 58.67$ instead of 44.03.

Incorporation of an opportunity can always be done like this; we only need to define an appropriate activity, in this case an activity that is fixed in time. In general, each activity that a maintenance manager wants to fix in time can be represented by a penalty function that is infinite for every shift unequal to zero.

Another possibility is to fix certain activities in a group; suppose for instance that the maintenance manager wants to execute activities 8,9,10,11 together (possibly with other activities). In that case we define a new activity *new* with tentative execution time $t^*_{new} = t^*_{\{8,9,10,11\}}$, and with a penalty function that equals the sum of the individual penalty functions minus the minimum value of the sum: $h_{new}(\Delta t) = H_{\{8,9,10,11\}}(t^*_{\{8,9,10,11\}} + \Delta t) - H^*_{\{8,9,10,11\}}$. This implies that $h_{new}(\cdot) \geq 0$, $h_{new}(0) = 0$ and that $h_{new}(\cdot)$ is strictly convex. Therefore, Assumptions 1, 2 and 3, are satisfied so that we can apply our approach. When we do so we find that activity *new* is executed together with activities 5, 6 and 7; the savings of the group {5,6,7,*new*} are 41.33. However, as activity *new* represents a group of four activities, it yields savings itself: there is a set-up cost reduction of $3S$. Since the penalty costs of executing activities 8,9,10,11 together amount to $H^*_{\{8,9,10,11\}} = 2.82$, the savings of activity *new* are $3S - H^*_{\{8,9,10,11\}} = 45 - 2.82 = 42.18$. When we add this to the savings of 41.33, we find that the actual savings of group {5,6,7,*new*}={5,6,7,8,9,10,11} are 41.33+42.18=83.51. The corresponding grouping structure is given in Table V.

Table V
Grouping structure in case activities 8, 9, 10 and 11 must be executed together

Group	Savings	Day
{1,2,3,4}	44.03	7.2
{5,6,7,8,9,10,11}	83.51	94.1
{12,13,14,15,16}	88.99	186.2
Total savings	186.92	

The total savings of this grouping structure are 186.92, which is slightly less than the (semi-) optimal value of 191.24 according to Table IV. The maintenance manager can now decide whether he/she indeed prefers this grouping structure with activities 8,9,10,11 executed together to the structure of Table IV.

4 Flexibility and Insight of the Approach

The approach presented in this paper has many advantages some of which will be discussed below.

Our approach can be applied to many preventive-maintenance optimisation models. In fact, it can be applied to all models described by Dekker [4] such as minimal repair, block replacement, inspection and efficiency models, but also to age-replacement kind of models (see Dekker, Wildeman and Van Egmond [7]). Not only preventive maintenance but also corrective maintenance can be incorporated [7].

Yet our approach is not dependent on the underlying maintenance models. This is due to the fact that for an activity only a tentative execution time is needed and a penalty function that indicates how much we have to pay for deviating from this time. This implies that for different activities we can chose different models; we can for instance combine activities modelled according to a minimal-repair model with activities modelled according to a block-replacement model. But it implies also that it is not even necessary to have an underlying model; tentative execution times and penalty functions can as well be estimated or specified directly (see, for example, Dekker, Smit and Losekoot [6]).

A problem with most finite-horizon models and, consequently, also with rolling-horizon approaches is that they require the definition of so-called *residual values*, which can be interpreted as the industrial value of the state of the system at the end of the horizon (compared to a brand-new system, for instance). The determination of residual values is rather arbitrary and depends on future strategies. The choice of a definition of the residual values can have substantial effects on the solutions that are generated (see, for example, Dekker, Wildeman and Van Egmond [7]). Our approach gets around the difficulty of defining residual values; the penalty functions give indications of how short-term decisions influence future costs.

The stability results that are derived in Section 3.4 are important as they show how the length of the planning horizon affects the generated planning. In practice it is favourable to chose a horizon such that a little change in this horizon hardly affects the planning, since it gives little insight and it would results in little confidence in the quality of the solution. Especially when a rolling horizon is applied it is often desirable to know if groups will not change (much) when a new horizon is chosen for a subsequent planning. By application of Theorem 4 and its corollaries we can often prove that groups will not change with another horizon. Consider for instance the example in Section 3.4: after execution of group {1,2,3,4} at day $t = 7.2$ a new planning can be made without losing the optimality of this group.

5 Conclusions

In this paper we presented a general rolling-horizon approach to group maintenance activities on a short-term basis. To this end a long-term tentative plan is made that is adapted to short-term information. This yields a dynamic grouping policy.

The approach presented in this paper enables interactive planning. According to the decisions of the maintenance manager, the planning of Phase 4 for a certain period can be adapted. Each time this is done, a dynamic-programming algorithm with a quadratic time complexity has to be applied. Therefore, the maintenance manager can easily and quickly see how decisions work out in the final maintenance plan. The stability results with respect to the length of the planning horizon also help in obtaining more insight in this process.

The flexibility of our approach and the insight it provides are presumably more important in practice than the 'optimality' of the solution. The complexity of practical situations makes it often impossible to find real optimal solutions; models are simplifications of practical situations and optimality is hard to achieve. Even if it is possible to find the 'optimal' solution, the resulting decision rules are often hard to implement and lack any structure.

Acknowledgement

The authors would like to thank S.W. Hadley and K.N. Srikanth of the Koninklijke/Shell-Laboratorium, Amsterdam for their useful comments.

A Reduction Theorems

For the theorems in this appendix we use Assumptions 1, 2 and 3 (see Section 3.4).

The following lemma is a trivial property of strictly-convex functions that will implicitly be used in the sequel.

Lemma 1 *If $f(\cdot)$ and $g(\cdot)$ are strictly-convex functions with minima t^*_f and t^*_g, where $t^*_f < t^*_g$, then $f(\cdot) + g(\cdot)$ is strictly convex and for its minimum we have $t^*_f < t^*_{f+g} < t^*_g$.*

Recall for the following theorems the three properties presented in Section 3.4.

Theorem 3 (Consecutive Activities) *If Property 1, 2 or 3 holds there exists an optimal grouping structure with consecutive activities.*

PROOF: Let F and G be two groups of activities. Consider two activities i and j, with $i \in G$ and $j \in F$ and $i < j$. Remark that $i < j$ implies that $t_i \leq t_j$. Let t^*_F, t^*_G be the optimal execution times of the groups F and G, respectively, and suppose that activity i and j are not executed in consecutive order, i.e., $t^*_F < t^*_G$.

We will prove that when $t_i < t_j$, each grouping structure that contains F and G cannot be optimal. For $t_i = t_j$ there sometimes can be an optimal grouping structure that contains F and G, but in that case there exists a grouping structure that is at least as good in which activity i is executed at t^*_F or activity j at t^*_G or both. Altogether this implies that there is always an optimal grouping structure in which each pair of activities is executed in consecutive order, which completes the proof.

Since it will be more convenient in this proof to think in terms of costs than in terms of savings, we take as objective the minimisation of the total costs instead of the maximisation of the total savings (the costs of a group are the penalty costs minus the reduction in set-up costs). We distinguish between three cases: $t^*_G \leq t_j$, $t_i \leq t^*_F$ and $t^*_F < t_i \leq t_j < t^*_G$.

$t^*_G \leq t_j$. Then we have $t^*_F < t^*_G \leq t_j$, and thus, using Assumptions 1, 2 and 3, it is cheaper to execute activity j at t^*_G than at t^*_F. This implies

$$\begin{aligned} & H^*_F - (|F|-1)S + H^*_G - (|G|-1)S = \\ & \quad = \sum_{l \in F} h_l(t^*_F - t_l) + \sum_{l \in G} h_l(t^*_G - t_l) - (|F| - 1 + |G| - 1)S \end{aligned}$$

$$\begin{aligned} &> \sum_{l \in F \setminus \{j\}} h_l(t_F^* - t_l) + \sum_{l \in G \cup \{j\}} h_l(t_G^* - t_l) - (|F| - 2 + |G|)S \\ &\geq H_{F\setminus\{j\}}^* - (|F| - 2)S + H_{G\cup\{j\}}^* - |G|S. \end{aligned}$$

(Note that $F \setminus \{j\}$ cannot be empty, and consequently that $|F| - 2 \geq 0$, since otherwise $F = \{j\}$ so that $t_F^* = t_j$, which is in contradiction with $t_F^* < t_G^* \leq t_j$.) Hence the groups $F \setminus \{j\}$ and $G \cup \{j\}$ are better than F and G. Consequently, F and G cannot be part of an optimal grouping structure.

$t_i \leq t_F^*$. Then we have $t_i \leq t_F^* < t_G^*$, and thus, using Assumptions 1, 2 and 3, it is cheaper to execute activity i at t_F^* than at t_G^*, and we can prove analogously that the groups F and G cannot be part of an optimal grouping structure, since the groups $F \cup \{i\}$ and $G \setminus \{i\}$ are better. (Note that $G \setminus \{i\}$ cannot be empty, since otherwise $G = \{i\}$ so that $t_i = t_G^*$, which is in contradiction with $t_i \leq t_F^* < t_G^*$.)

$t_F^* < t_i \leq t_j < t_G^*$. Suppose that it is *not* cheaper to execute activity i at t_F^* instead of at t_G^* or to execute activity j at t_G^* instead of at t_F^* (otherwise, F or G or both are not optimal as shown above). This implies that

$$\begin{aligned} h_i(t_G^* - t_i) &\leq h_i(t_F^* - t_i) \quad \text{and} & (6) \\ h_j(t_F^* - t_j) &\leq h_j(t_G^* - t_j). & (7) \end{aligned}$$

We now distinguish between Properties 1, 2 and 3 (in all cases Assumptions 1, 2 and 3 are used implicitly).

1. Property 1 (Symmetry).
   Equation (6) and the fact that $h_i(\cdot)$ and $h_j(\cdot)$ are symmetric imply that $t_G^* - t_i \leq t_i - t_F^*$ and consequently $t_G^* - t_j \leq t_G^* - t_i \leq t_i - t_F^* \leq t_j - t_F^*$. This implies that $h_j(t_G^* - t_j) \leq h_j(t_F^* - t_j)$.

2. Property 2 (Congruency).
   Suppose $\alpha h_i(\cdot) = h_j(\cdot)$, for some $\alpha > 0$. Equation (6) implies that $h_i(t_G^* - t_j) \leq h_i(t_G^* - t_i) \leq h_i(t_F^* - t_i) \leq h_i(t_F^* - t_j)$, and consequently $h_j(t_G^* - t_j) = \alpha h_i(t_G^* - t_j) \leq \alpha h_i(t_F^* - t_j) = h_j(t_F^* - t_j)$.

3. Property 3 (Dominance).
   Equation (6) and the fact that $h_i(\cdot)$ dominates $h_j(\cdot)$ right of t_j and $h_j(\cdot)$ dominates $h_i(\cdot)$ left of t_i imply that $h_j(t_G^* - t_j) < h_i(t_G^* - t_i) \leq h_i(t_F^* - t_i) < h_j(t_F^* - t_j)$.

If $t_i < t_j$ then we have strict inequalities in case of Properties 1 and 2 (we already have a strict inequality in case of Property 3). This implies that if equation (6) holds, equation (7) cannot be true if $t_i < t_j$. Analogously, if equation (7) holds, then equation (6) cannot be true (if $t_i < t_j$). Consequently, if $t_i < t_j$, both equations cannot be true at the same time. Hence, in that case it is more effective to execute activity i at time t_F^* or activity j at time t_G^* or both, which implies that F and G cannot be part of an optimal grouping structure. (Note that $F \setminus \{j\}$ and $G \setminus \{i\}$ cannot be empty since $t_F^* < t_i < t_j < t_G^*$.) If $t_i = t_j$ then we still have a strict inequality in case of Property 3, so that the same holds in that case. For Properties 1 and 2 it holds that strict inequality in one of the equations implies a contradiction with the other. So either one of the equations is an inequality, or both are equalities. In the first case it is more effective to execute one of the activities at the execution time of the group of the other. In the latter case one activity (or both) can be executed at the time of the other without increasing the costs. This implies that we can always find a grouping structure that is at least as good and in which the activities i and j are executed in consecutive order. □

In the following example it is shown that if none of the properties holds, there may be a unique optimal grouping structure in which the activities are not consecutive. Suppose that four activities are tentatively planned at execution times t_1, t_2, t_3 and t_4. Suppose further that the penalty functions of the activities are as drawn in Figure 2 (with asymptotes of $h_1(\cdot)$ in 0 and in $(t_1 + t_2)/2$, of $h_2(\cdot)$ in $(t_1 + t_2)/2$, of $h_3(\cdot)$ in $(t_3 + t_4)/2$, and of $h_4(\cdot)$ in $(t_3 + t_4)/2$ and in T). Notice that none of the properties holds for all activities.

Table VI shows the possible grouping structures in which only consecutive activities appear, and the corresponding savings (where $h_2(t_{2,3}^* - t_2) = h_3(t_{2,3}^* - t_3) = \epsilon$). The grouping structure {1},{2,3},{4} with savings $S - 2\epsilon$ is the best structure with consecutive activities. However, group {1,3} has an optimal execution time just right of t_1 with penalty costs S/M and group {2,4} has an optimal execution time just left of t_4 with the same penalty costs (i.e., S/M). Thus the grouping structure {1,3},{2,4} yields

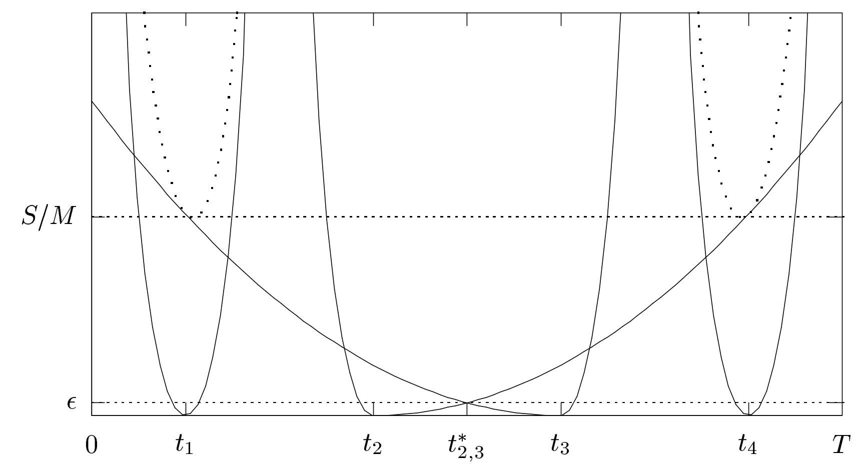

Figure 2. Penalty functions without Property 1, 2 or 3.

savings $S - S/M + S - S/M = 2S - 2S/M$, which is better than $S - 2\epsilon$ if M is sufficiently large. It is easily verified that the grouping structure {1,3},{2,4} is optimal and unique. Consequently, the optimal grouping structure does not consist of groups with consecutive activities.

Table VI
Grouping Structures with Consecutive Activities

Structure	Savings	Structure	Savings
{1},{2},{3},{4}	0	{1},{2},{3,4}	$-\infty$
{1,2},{3},{4}	$-\infty$	{1,2,3},{4}	$-\infty$
{1,2},{3,4}	$-\infty$	{1},{2,3,4}	$-\infty$
{1},{2,3},{4}	$S - 2\epsilon$	{1,2,3,4}	$-\infty$

Notice that in the example the quotient of the optimal total savings and the total savings of the best consecutive solution equals $(2S - 2S/M)/(S - 2\epsilon)$. This quotient can be arbitrarily close to the value 2 by making M sufficiently large and ϵ sufficiently small. In Appendix C we show that this is a worst case; the optimal total savings are always less than twice as much as the total savings of the optimal *consecutive* solution.

We will now prove Theorem 4.

Theorem 4 *If in an optimal grouping structure of the first s activities activity s is executed in another group than an activity p $(1 \leq p < s)$, then for any $r > s$ there is an optimal grouping structure of the first r activities in which activity s is also executed in another group than activity p.*

PROOF: In the following, GS_i always denotes an optimal grouping structure of the first i activities (i.e., activities $1, \ldots, i$).

Consider an optimal grouping structure GS_s of the first s activities in which activity s is executed separately from activity p. Accordingly, in GS_s there exists an activity $p' \geq p$ such that activity $p' + 1$ is executed in a group $\{p' + 1, \ldots, q\}$ with $p' + 1 \leq q \leq s$. This implies that the grouping structure GS_q defined by $GS_q := GS_{p'} \cup \{p' + 1, \ldots, q\}$ (i.e., an optimal grouping structure $GS_{p'}$ of the first p' activities extended with the group $\{p' + 1, \ldots, q\}$) is an optimal grouping structure of the first q activities. We will prove that for any $r > q$ there is an optimal grouping structure GS_r of the first r activities in which activity p' is executed separately from activity r. Then this holds a forteriori for p and any $r > s$.

As in the proof of Theorem 3 we take as objective the minimisation of the total costs instead of the maximisation of the total savings. Let the total costs of an optimal grouping structure of the first i

activities be denoted by $C(GS_i)$. Take $r > q$ and let GS'_r be an optimal grouping structure of the first r activities. If in this GS'_r activity r is executed separately from activity p', then we take $GS_r = GS'_r$ and we are ready. If in GS'_r activity r is not executed separately from activity p', then GS'_r contains a group $\{l, \ldots, r\}$ with $l \leq p'$. We shall show that the grouping structure GS_r with group $\{p'+1, \ldots, r\}$ instead of $\{l, \ldots, r\}$ is at least as good as GS'_r. We distinguish between the case that $t^*_{p'+1,\ldots,q} \leq t^*_{l,\ldots,r}$ and $t^*_{p'+1,\ldots,q} > t^*_{l,\ldots,r}$.

Suppose first that $t^*_{p'+1,\ldots,q} \leq t^*_{l,\ldots,r}$. For the costs $C(GS'_r)$ of GS'_r we have

$$\begin{aligned} C(GS'_r) &= C(GS_{l-1}) + H^*_{l,\ldots,r} - (r-l)S \\ &= C(GS_{l-1}) + H_{l,\ldots,r}(t^*_{l,\ldots,r}) - (r-l)S \\ &= C(GS_{l-1}) + H_{l,\ldots,p'}(t^*_{l,\ldots,r}) + H_{p'+1,\ldots,r}(t^*_{l,\ldots,r}) - (r-l)S. \end{aligned}$$

Assumptions 1, 2 and 3 in combination with the fact that $t^*_{l,\ldots,p'} \leq t^*_{p'+1,\ldots,q} \leq t^*_{l,\ldots,r}$ yield

$$H_{l,\ldots,p'}(t^*_{l,\ldots,r}) \geq H_{l,\ldots,p'}(t^*_{p'+1,\ldots,q}).$$

Using this and adding $0 = H_{p'+1,\ldots,q}(t^*_{p'+1,\ldots,q}) - H_{p'+1,\ldots,q}(t^*_{p'+1,\ldots,q})$ we have

$$\begin{aligned} C(GS'_s) &\geq C(GS_{l-1}) + H_{l,\ldots,p'}(t^*_{p'+1,\ldots,q}) + H_{p'+1,\ldots,q}(t^*_{p'+1,\ldots,q}) \\ &\quad - H_{p'+1,\ldots,q}(t^*_{p'+1,\ldots,q}) + H_{p'+1,\ldots,r}(t^*_{l,\ldots,r}) - (r-l)S \\ &= C(GS_{l-1}) + H_{l,\ldots,q}(t^*_{p'+1,\ldots,q}) - (q-l)S \\ &\quad - H_{p'+1,\ldots,q}(t^*_{p'+1,\ldots,q}) + H_{p'+1,\ldots,r}(t^*_{l,\ldots,r}) - (r-q)S. \end{aligned}$$

The costs $C(GS_{l-1}) + H_{l,\ldots,q}(t^*_{p'+1,\ldots,q}) - (q-l)S$ are the costs of a grouping structure of the first q activities, which are of course greater than or equal to $C(GS_q)$. Since GS_q contains group $\{p'+1, \ldots, q\}$, we have:

$$\begin{aligned} C(GS'_r) &\geq C(GS_{p'}) + H_{p'+1,\ldots,q}(t^*_{p'+1,\ldots,q}) - (q-p'-1)S \\ &\quad - H_{p'+1,\ldots,q}(t^*_{p'+1,\ldots,q}) + H_{p'+1,\ldots,r}(t^*_{l,\ldots,r}) - (r-q)S \\ &\geq C(GS_{p'}) + H^*_{p'+1,\ldots,r} - (r-p'-1)S. \end{aligned}$$

Thus we have a grouping structure of the first r activities with costs less than or equal to $C(GS'_s)$ and with group $\{p'+1, \ldots, r\}$ instead of group $\{l, \ldots, r\}$; for GS_s we take this grouping structure. Since GS'_r is optimal, GS_s must also be optimal.

If $t^*_{p'+1,\ldots,q} > t^*_{l,\ldots,r}$, then $H_{l,\ldots,r}(t^*_{l,\ldots,r})$ is not split up in $H_{l,\ldots,p'}(t^*_{l,\ldots,r}) + H_{p'+1,\ldots,r}(t^*_{l,\ldots,r})$ but in $H_{l,\ldots,q}(t^*_{l,\ldots,r}) + H_{q+1,\ldots,r}(t^*_{l,\ldots,r})$. Further the proof is analogous. □

B Dynamic-Programming Algorithm

Here we show how Theorems 1, 2 and 4 can be incorporated in the approach of Section 3.4 and we present the resulting dynamic-programming algorithm.

Although Theorem 1 is redundant when Theorem 2 is applied, it will be used because it takes less time (the intervals I_i can be stored, hence checking whether $\cap_{l \in G} I_l = \emptyset$ is easier than calculating H^*_G). Besides, having the intersection $\cap_{l \in G} I_l$ facilitates the calculation of H^*_G.

Let $G = \{i, \ldots, j\}$, $i < j$, be an arbitrary group with consecutive activities. If $\cap_{l \in G} I_l = \emptyset$, then G and every other group that contains G cannot be optimal according to Theorem 1. We use a parameter *Low*, indicating the lowest indexed activity that can be in an optimal group together with activity j in iteration j. If $\cap_{l \in G} I_l = \emptyset$, then *Low* is updated to $i+1$, because in later iterations $k > j$ no activity with index smaller than $i+1$ can be in a group that is part of the optimal grouping structure.

If G is not excluded by Theorem 1, then the savings of G, $(j-i)S - H^*_G$, and the corresponding optimal execution time t^*_G are calculated. Subsequently, G is compared to the groups $G_1 = G \setminus \{i\}$ and $G_2 = G \setminus \{j\}$. If G has less savings than G_1 or G_2 then G can be split up more effectively into two clusters of activities, viz. into $\{i\}$ and G_1 or into G_2 and $\{j\}$ (remember that the savings of an activity executed on its own are zero). Then G and every group that contains G as a cluster cannot be part of an optimal grouping structure according to Theorem 2. These are the groups $\{l, \ldots, k\}$, $l \leq i$ and $k \geq j$.

In that case *Low* is updated to $i+1$, since we do not need to consider groups with first activity lower than $i+1$ in later iterations. To avoid extra work while comparing G to G_1 and G_2, we use a variable with the savings of G_1 (the previous group considered) and, after checking G, update it with the savings of G. For G_2 we use an array with the savings of the groups considered in the previous iteration, and we update the i^{th} entry, after reading the savings of G_2, with the savings of G.

Suppose that $G = \{i, \dots, j\}$ is the best group in iteration j. This implies that G is part of an optimal grouping structure of the first j activities. In any following iteration $k > j$ there is by Theorem 4 a best group containing activity k, which does not contain any of the activities $1, \dots, i-1$. This implies that the variable *Low* can be updated to i.

Altogether, we have the following algorithm (for the definitions of the arrays *TotalSavings* and *First* see Section 3.4):

```
$Low := 1$;
$TotalSavings[0] := 0$;
FOR $j := 1$ TO $n$ DO
    $i := j$;
    $First[j] := j$;
    $TotalSavings[j] := TotalSavings[j-1]$;
    WHILE $i \geq Low$ DO
        Step 1 Let $C = \{i, \dots, j\}$.
               IF $i = j$
                   THEN the savings $S(C)$ of $C$ are zero; goto step 5;
        Step 2 IF $\cap_{l \in C} I_l = \emptyset$
                   THEN $Low := i + 1$ (Theorem 1); goto step 5;
        Step 3 Calculate $S(C)$ = the savings of $C$, and the corresponding
               optimal execution time. Let $C_1 = C \setminus \{i\}$, $C_2 = C \setminus \{j\}$,
               and $S(C_1)$ and $S(C_2)$ be the corresponding savings;
               IF $S(C) < S(C_1)$ OR $S(C) < S(C_2)$
                   THEN $Low := i + 1$ (Theorem 2); goto step 5;
        Step 4 IF $S(C) + TotalSavings[i - 1] > TotalSavings[j]$
                   THEN $First[j] := i$;
                        $TotalSavings[j] := S(C) + TotalSavings[i - 1]$;
        Step 5 $i := i - 1$;
    END WHILE
    $Low := First[j]$ (Theorem 4).
END FOR
```

In the worst case, e.g., when $t_i = t_0 + i\epsilon$ for all i, with ϵ sufficiently small, Theorems 1, 2 and 4 do not exclude any group at all, which implies that $(1/2)n(n+1)$ groups are considered. In the best possible case, e.g., when the distances between the tentative execution times are so large that combining is never cost-effective, only the groups consisting of one and two activities are considered, of which there are $n + (n-1) = 2n - 1$. This implies that in the worst case the algorithm has a time complexity of $\mathcal{O}(n^2)$, whereas in the best case an optimal grouping structure is found in linear time.

C Optimality of Dynamic Programming

In the example of Section 3.4 a number of situations for the minimal-repair model is mentioned in which there exists an optimal grouping structure with consecutive activities. When such a structure exists, it can be found with the dynamic-programming algorithm described before.

If none of these situations occurs, there can still be a consecutive solution. Corollary 1 states that activities in a set A satisfying one of the Properties 1, 2 or 3 are always executed in consecutive order. This result can be used to check the existence of an optimal consecutive solution. We shall clarify this with an example. Suppose there are four activities 1,2,3,4. Activities 1 and 2 have symmetric penalty functions and are therefore executed in consecutive order. The function $h_2(\cdot)$ dominates $h_3(\cdot)$ right of

t_3 and $h_3(\cdot)$ dominates $h_2(\cdot)$ left of t_2; therefore, activities 2 and 3 are executed in consecutive order. Activities 3 and 4 have congruent penalty functions and are therefore also executed in consecutive order. This implies that all activities are executed in consecutive order, so that a consecutive optimal solution exists.

When the existence of a consecutive optimal solution cannot be proved, then Corollary 1 can also be used in a branch and bound procedure in which enumeration (set partitioning) is applied on the remaining possibilities. Consider, for instance, the example of Figure 2 in Appendix A. The penalty function of activity 1 dominates that of activities 2 (right of t_2) and 4 (right of t_4), so that activity 1 is executed before activities 2 and 4. The penalty function of activity 4 dominates that of activities 1 (left of t_1) and 3 (left of t_3), so that activity 4 is executed after activities 1 and 3. By this result many grouping structures are eliminated. For the remaining possibilities enumeration can be applied. One can of course still use Theorems 1 and 2 while applying enumeration, to make further eliminations.

Another possibility is to apply the dynamic-programming algorithm as a heuristic in case the existence of an optimal consecutive solution cannot be proved. The dynamic-programming algorithm finds a grouping structure that is optimal among the grouping structures with consecutive activities. As we saw in the example of Figure 2 in Appendix A, this is not necessarily an overall optimal grouping structure when none of the properties holds for all activities. We constructed an example in which the optimal total savings can be arbitrarily close to twice as much as the best grouping structure with consecutive activities. Below we show that this is the worst case. The optimal total savings are never twice or more as much as the total savings of the best grouping structure with consecutive activities.

To this end we need the following lemma. In this lemma we use the notion of *intermediate* activities. We say that an activity i is intermediate for a group G if $i \in [\min_{j\in G} j, \max_{j\in G} j]$, but $i \notin G$. For example, for group {2,5,7,8,9,11} activities 3,4,6,10 are intermediate activities. If activity 3 is executed in group {1,3,4,6,10} then activities 2,5,7,8,9 are intermediate activities for this group. Lemma 2 states that in an optimal grouping structure each activity that is an intermediate activity for a certain group, is executed before the tentative execution time of the first activity in that group (that is, the activity in the group with the lowest index) or after the tentative execution time of the last activity in the group (the one with the highest index). For example, if group {2,5,7,8,9,11} is in an optimal grouping structure, then activity 3, which is an intermediate activity for this group, must be executed before t_2 or after t_{11}. Consequently, if activity 3 is in group {1,3,4,6,10}, then the optimal execution time $t^*_{\{1,3,4,6,10\}}$ of this group must be outside the interval $[t_2, t_{11}]$.

Lemma 2 *In an optimal grouping structure it holds that each activity i that is an intermediate activity for a group G must be executed in a group F_i with optimal execution time $t^*_{F_i} \notin [\min_{j\in G} t_j, \max_{j\in G} t_j]$.*

PROOF: Let activity i be an intermediate activity for group G: $i \in [\min_{j\in G} j, \max_{j\in G} j]$, but $i \notin G$. Let min denote the activity in group G with the lowest index and max the activity with the highest index and let t_{min} and t_{max} be the corresponding tentative execution times, then $t_{min} = \min_{j\in G} t_j$ and $t_{max} = \max_{j\in G} t_j$. Let activity i be in a group F_i with optimal execution time $t^*_{F_i}$. We will show that if $t^*_{F_i} \in [t_{min}, t_{max}]$ the group G cannot be optimal (notice that this also implies that the group F_i contains activities other than activity i, since otherwise $t^*_{F_i} = t_i \in [t_{min}, t_{max}]$).

Let t^*_G be the optimal execution time of group G, then $t^*_G \neq t^*_{F_i}$, since otherwise the group $G \cup F_i$ is a group with the same penalty costs and with an extra set-up cost reduction of S, so that the group $G \cup F_i$ is better than G and F_i, which implies that G and F_i cannot be optimal. Consequently, if $t^*_{F_i} \in [t_{min}, t_{max}]$ we have that $t^*_G \in [t_{min}, t^*_{F_i})$ or $t^*_G \in (t^*_{F_i}, t_{max}]$ (notice that this also implies that t_{min} is strictly smaller than t_{max}). Due to Assumptions 1, 2 and 3 it is in the first case cheaper to execute activity max at time $t^*_{F_i}$ instead of at t^*_G, and in the latter case it is cheaper to execute activity min at time $t^*_{F_i}$ instead of at t^*_G. As in both cases the penalty costs are smaller and the total set-up cost reduction does not change, we have that group G cannot be optimal. □

Let now $TS^*(P)$ be the total savings of an optimal grouping structure of a problem P and let $TS^{dp}(P)$ be the total savings of the grouping structure found with the dynamic-programming algorithm. Hence $TS^{dp}(P)$ are the total savings of a best grouping structure with consecutive activities. Using Lemma 2, we can now prove that $TS^*(P)/TS^{dp}(P) < 2$.

Theorem 5 *It holds that $TS^*(P)/TS^{dp}(P) < 2$.*

PROOF: Take an arbitrary problem P with n activities and with penalty functions that satisfy Assumptions 1, 2 and 3. Let GS_n be an optimal grouping structure and let $TS^*(P)$ be the corresponding optimal total savings. Assume that in GS_n not all activities are executed in consecutive order (otherwise $TS^*(P) = TS^{dp}(P)$ and hence $TS^*(P)/TS^{dp}(P) = 1 < 2$, so that the proof immediately follows). From this non-consecutive GS_n we will construct a consecutive grouping structure GS'_n such that the total set-up cost reduction is not smaller than half the total set-up cost reduction of GS_n and such that the total penalty costs of GS'_n are smaller than those of GS_n. This implies that GS'_n has total savings that are larger than $TS^*(P)/2$. Since GS'_n is a grouping structure with consecutive activities, its total savings are of course smaller than or equal to $TS^{dp}(P)$, so that $TS^{dp}(P)$ is larger than $TS^*(P)/2$ and the desired result follows.

How do we construct such a GS'_n from GS_n? Consider in grouping structure GS_n the group G_1 that contains activity 1. If there is no intermediate activity for group G_1, then all activities in G_1 are executed in consecutive order (group G_1 may then consist of activity 1 only). In that case we define $G'_1 = G_1$ for the grouping structure GS'_n. If there is at least one activity that is an intermediate activity for group G_1, let then activity i be such an activity with the lowest index. Then we know that activities $1, 2, \ldots, i-1$ are in group G_1 and are executed in consecutive order, and we define $G'_1 = \{1, 2, \ldots, i-1\}$ for the grouping structure GS'_n (it may be that i equals 2, so that $G'_1 = \{1\}$). Notice that this implies that G'_1 is the first consecutive subgroup in group G_1.

Let max be the activity in G_1 with the highest index, then $t_{max} = \max_{j \in G_1} t_j$. Furthermore, we have that $\min_{j \in G_1} t_j = t_1$. Application of Lemma 2 yields that all intermediate activities for group G_1 are executed before t_1 or after t_{max}. Execution before t_1 can never be cost-effective when Assumptions 1, 2 and 3 are satisfied, so that all intermediate activities for group G_1 are executed after t_{max}. Let now activity j be the first activity in group G_1 with an index higher than i, then activities $i, i+1, \ldots, j-1$ are all intermediate activities for group G_1 (this may concern activity i only if $j = i+1$), and are therefore executed after t_{max}. Let now k be the index of the first activity after activity j that is intermediate for group G_1. This implies that activities $j, j+1, \ldots, k-1$ are in group G_1 and are executed in consecutive order (it may be that $k = j+1$, so that it concerns activity j only). This implies that the group $\{j, j+1, \ldots, k-1\}$ is a consecutive subgroup of G_1 and that these activities can be grouped cost-effectively. Activities $i, i+1, \ldots, j-1$ are intermediate activities for group G_1 and, consequently, executed after t_{max}. Execution of these activities before t_{max} is always cheaper due to Assumptions 1, 2 and 3, so that these activities can cost-effectively be grouped with activities $j, j+1, \ldots, k-1$. Now we define a new group G'_i for grouping structure GS'_n: $G'_i = \{i, i+1, \ldots, j-1, j, j+1, k-1\}$. Notice that group G'_i, through the way it is constructed, contains *at least* two activities and that there are not more than two activities only if $j = i+1$ and $k = j+1$ (in that case $G'_i = \{i, j\}$). Notice further that for each activity in this group its penalty costs are smaller than or equal to those in grouping structure GS'_n, since the shifts for these activities in group GS'_i are equal or smaller and Assumptions 1, 2 and 3 are satisfied.

Subsequently, we start with a new group G'_k as above, where activity k plays the role of activity i. We continue until all activities $1, 2, \ldots, max$ are covered.

Then we go on with activity $max+1$ that is not in group G_1 but in a group G_{max+1}. Activity $max+1$ plays the role of activity 1 above. Notice that by our construction all activities with index higher than max in GS_n must be executed after t_{max+1}, so that the situation is indeed the same as with activity 1 above. We continue this process until all n activities are covered.

In the above construction, each group of the grouping structure GS_n is split up in subgroups and intermediate activities are added. Let there be m groups in GS_n, then the total set-up cost reduction of GS_n equals $(n-m)S$. Each time a group of GS_n is split up, the first activity (equivalent with activity 1 above) may be executed on its own in the new grouping structure GS'_n, but each of the other activities of the group is executed with at least one other activity. Consequently, as there are m groups in GS_n, there may be m activities that are executed on their own in GS'_n. However, the remaining $n-m$ activities are in the worst case paired. Therefore, the total set-up cost reduction of GS'_n is equal to $(n-m)S/2$ in the worst case. Summarising, in the worst case the total set-up cost reduction of the grouping structure GS'_n is half the total set-up cost reduction of the grouping structure GS_n.

The penalty costs of the activities in grouping structure GS'_n are equal to or smaller than the penalty costs in GS_n, since the shifts for each activity are equal or smaller. Altogether, this implies that the total savings of GS'_n are larger than half $TS^*(P)$, the total savings of GS_n. Since the total savings of GS'_n are smaller than or equal to the total savings $TS^{dp}(P)$ of a *best* grouping structure with consecutive activities, we have that $TS^{dp}(P) > TS^*(P)/2$, and consequently that $TS^*(P)/TS^{dp}(P) < 2$, which completes the proof. □

When we apply the dynamic-programming algorithm as a heuristic, this is according to Theorem 5 always less than twice as bad as solving the problem to optimality. This is a nice result. Yet we can obtain another bound that is mostly much better by using the dynamic-programming algorithm itself.

To this end we apply a technique of Wildeman [16]. We define for each penalty function $h_i(\cdot)$ a symmetric function $\underline{h}_i(\cdot)$ such that $\underline{h}_i(\cdot) \leq h_i(\cdot)$ and Assumptions 1, 2 and 3 are satisfied. Let P and $\underline{P}$ be the problems with $h_i(\cdot)$ and $\underline{h}_i(\cdot)$, respectively, and let $TS^*(P)$ and $TS^*(\underline{P})$ be the total savings of the corresponding optimal grouping structures of these two problems. Let $TS^{dp}(P)$ be the total savings of the grouping structure of P found by the dynamic-programming algorithm. It is obvious that $TS^*(\underline{P}) \geq TS^*(P) \geq TS^{dp}(P)$, since the savings are defined as the reduction in set-up costs minus the penalty costs. Now $\underline{P}$ can be solved to optimality with the dynamic-programming algorithm, since for this problem Property 1 holds ($\underline{h}_i(\cdot)$ is symmetric). This implies that the upper bound $TS^*(\underline{P})$ on the total savings can easily be determined. If $TS^*(\underline{P})$ is close to $TS^{dp}(P)$, then solving P with dynamic programming obtains a good approximation to an optimal solution $TS^*(P)$.

This approach is used in Section 3.4 to find an upper bound on the total savings of an optimal grouping structure of the 16 activities modelled according to a minimal-repair model. This is based on the following lemma:

Lemma 3 *For a penalty function $h_i(\cdot)$ evaluated according to an LTS (equation (5)) it holds that*

$$h_i(-|\Delta t|) \begin{cases} \geq h_i(|\Delta t|) & \textit{for } \beta_i \leq 2 \textit{ and} \\ \leq h_i(|\Delta t|) & \textit{for } \beta_i \geq 2. \end{cases}$$

Consequently, for $\beta_i \leq 2$, we can define $\underline{h}_i(\cdot)$ as follows: $\underline{h}_i(\Delta t) := h_i(|\Delta t|)$. For $\beta_i \geq 2$, we take $\underline{h}_i(\Delta t) := h_i(-|\Delta t|)$. Notice that $\underline{h}_i(\cdot)$ is symmetric (i.e., Property 1 holds), and satisfies Assumptions 1, 2, and 3. Application of the dynamic-programming algorithm to $\underline{P}$ for the 16 maintenance activities in Section 3.4 yields the following upper bound: $TS^*(\underline{P}) = 191.26$. Consequently, an optimal grouping structure of the 16 activities (problem P) has total savings $TS^*(P)$ smaller than or equal to 191.26.

References

[1] Bäckert, W. and D.W.T. Rippin. The determination of maintenance strategies for plants subject to breakdown. *Computers and Chemical Engineering*, **9**:113–126, 1985.

[2] Chakravarty, A.K., J.B. Orlin, and U.G. Rothblum. A partitioning problem with additive objective with an application to optimal inventory groupings for joint replenishment. *Operations Research*, **30**:1018–1022, 1982.

[3] Cho, D.I. and M. Parlar. A survey of maintenance models for multi-unit systems. *European Journal of Operational Research*, **51**:1–23, 1991.

[4] Dekker, R. Integrating optimisation, priority setting, planning and combining of maintenance activities. *European Journal of Operational Research*, **82**:225–240, 1995.

[5] Dekker, R. and I.F.K. Roelvink. Marginal cost criteria for preventive replacement of a group of components. *European Journal of Operational Research*, **84**:467–480, 1995.

[6] Dekker, R., A.C.J.M. Smit, and J.A. Losekoot. Combining maintenance activities in an operational planning phase: a set-partitioning approach. *IMA Journal of Mathematics Applied in Business and Industry*, **3**:315–331, 1992.

[7] Dekker, R., R.E. Wildeman, and R. van Egmond. Joint replacement in an operational planning phase. Technical Report 9438/A, Econometric Institute, Erasmus University Rotterdam (to appear in the *European Journal of Operational Research*), 1996.

[8] Garey, M.R. and D.S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[9] Goyal, S.K. and A. Gunasekaran. Determining economic maintenance frequency of a transport fleet. *International Journal of Systems Science*, **4**:655–659, 1992.

[10] Goyal, S.K. and M.I. Kusy. Determining economic maintenance frequency for a family of machines. *Journal of the Operational Research Society*, **36**:1125–1128, 1985.

[11] Howard, R.A. *Dynamic Programming and Markov Processes.* Wiley, New York, 1960.

[12] Stinson, J.P. and B.M. Khumawala. The replacement of machines in a serially dependent multi-machine production system. *International Journal of Production Research*, **25**:677–688, 1987.

[13] Van der Duyn Schouten, F.A. and S.G. Vanneste. Analysis and computation of (n, N)-strategies for maintenance of a two-component system. *European Journal of Operational Research*, **48**:260–274, 1990.

[14] Van Dijkhuizen, G. and A. van Harten. Optimal clustering of repetitive frequency-constrained maintenance jobs with shared setups. Technical report, University of Twente, The Netherlands (submitted for publication), 1996.

[15] Wagner, H.M. *Principles of Operations Research.* Prentice/Hall International, 1975.

[16] Wildeman, R.E. Combined maintenance scheduling. Master's thesis, Leiden University, 1991.

[17] Worm, J.M. and A. van Harten. Model based decision support for planning of road maintenance. To appear in Reliability Engineering & Systems Safety, 1996.