IMA Journal of Mathematics Applied in Business & Industry (1992) **3**, 315–331

Combining maintenance activities in an operational planning phase: a set-partitioning approach

ROMMERT DEKKER†, ADRIAAN SMIT, AND JEROEN LOSEKOOT‡
Koninklijke/Shell-Laboratorium, Amsterdam

[Received 23 September 1991 and in revised form 9 December 1991]

Joint execution of maintenance activities can reduce costs because preparative activities, like opening a machine, may be shared. Combining execution, however, also implies that activities are carried out at times other than originally planned. In this paper, we analyse the problem of determining which activities should be combined at which moments in time. We develop a methodology to represent the cost-effectiveness of combining activities, and to identify an optimal combination plan. The method consists of three phases: one in which penalty cost functions are derived, another in which combinations are evaluated, and finally one in which the optimal combination is obtained through a set-partitioning algorithm.

1. Introduction

Maintenance management problems in the operational phases of technical systems may be broken down into several subproblems that relate to various timescales, as follows.

- Define maintenance activities and procedures which determine how and how often equipment is to be maintained in the longer term, say two to five years. This exercise is the so-called maintenance concept.
- Plan and prepare the major activities such as large overhauls and shutdowns. Match the need and availability of maintenance resources (e.g. manpower and materials) over shorter periods, six months say.
- Allocate (known) work to available manpower by setting priorities, resulting in time-ordered work schedules that cover one or more weeks and satisfy all (production) requirements.

To set up a maintenance concept, a technical installation should first be functionally analysed, breaking equipment down into systems, subsystems, units, and components or elements, down to the level where individual failure modes can be recognized. Then, on the basis of a failure-mode, effect, and criticality analysis, preventive maintenance activities can be defined to anticipate unwanted consequences. In the maintenance concept, all these activities are individually defined, evaluated, and possibly optimized, assuming unlimited resources. To reduce the resulting multitude of activities, some combination of related activities into packages may have been applied already. Yet it is likely that the number of packages remains large,

† Present address: Erasmus University, Rotterdam.
‡ Master's student at Twente University of Technology.

interspersed with corrective maintenance activities. Also, since resources and requirements change with time, operational maintenance planning becomes an intricate and unsurveyable task which is often overthrown by urgent corrective work that has higher priority. Consequently, maintenance planners would be much helped by quantitative techniques that can provide decision support.

One issue in cost-effective maintenance planning arises from the fact that maintenance activities often require the same preparative work. If a maintenance job has to be done on a remote unmanned platform, then simultaneous execution of other maintenance activities saves travel costs. If a production system has to be shut down to do some maintenance work, then other work may be done at the same time, without incurring shutdown costs. These aspects can hardly be incorporated in the concept phase because they would make the problem too complex, or because a long-term combination may be undesirable. The objective of this paper is to formulate and analyse a mathematical model that can assist a maintenance manager in combining activities during his planning. Here we assume that some maintenance management information system is present, in which essential data on maintenance activities is stored (i.e. results from the maintenance concept phase).

This paper is organized as follows. Section 2 starts with a review of the open literature, focusing on contributions to (cost-effective and) operational maintenance planning techniques, and in particular, to the maintenance activity combination problem resulting therefrom. Section 3 provides a mathematical problem formulation, and presents penalty functions which allow for cost-centred maintenance planning. Given a limited time horizon, these penalties are utilized to address the activity combination problem, indicating how timing can be optimized and the cost-effectiveness of combinations can be evaluated. The cost-optimal combinations (which constitute a partition of the original set of maintenance activities) can be obtained by applying set-partitioning methods (Section 4). As there are usually numerous maintenance activities to deal with, one encounters large computational problems if all possible combinations have to be evaluated. In setting up our approach, we therefore assume an appropriate savings structure. Consequently we prove three lemmas that considerably reduce the problem size. Section 5 presents a numerical example to clarify the proposal's potential. Finally, the merits and drawbacks of the approach are discussed in Section 6, resulting in some conclusions (Section 7).

2. Literature survey

From the literature surveys of Sherif & Smith (1981) or Valdez-Flores & Feldman (1989), for instance, it appears that most publications deal with so-called maintenance optimization models. More specifically, the core of the literature focuses on models aimed at identifying some long-term optimal maintenance strategy, be that an individual, a collective, or even a dynamic one. The problem of cost-effective and operational maintenance planning is rarely addressed, whereas references to the activity combination problem (the topic of this paper) are even harder to find.

In one paper, Liang's (1985) so-called 'piggyback approach' defines time windows for each activity, within which execution dates may be varied. Optimization is pursued

only with respect to the number of combined activities, possibly resulting in many 'optimal' solutions between which one has to choose. Altogether this constitutes a pragmatic but rather crude way of dealing with the combination problem. In two other contributions, maintenance planning problems are tackled using LP-like job-scheduling and combinatorial optimization approaches. In this context, Phillips (1979) allows four different job types to be scheduled on the basis of minimal backlog, minimal flow time, or maximal efficiency. Notice that, so far, costs are not considered at all. Mann & Bostock (1983) apply network planning tools, postulating penalties/preferences derived from what they refer to as downtime cost factors. These penalties are both empirical and relative, being built up from factors relating to fixed and variable costs, from the reliability of redundant equipment, and from service time or product inventory. Combinations can be analysed, but only if maintenance activities include pure waiting time.

The combining of maintenance work also occurs in opportunity maintenance models. According to a review in Dekker & Smeitink (1991), models either consider combination at one given opportunity only and then lack the planning aspect, or they apply Markov decision chains, which are tractable for very few (say 3 to 4) activities only. Thus, no real contribution from that side is found either.

Finally, combination may also pertain to general activities: e.g. joint ordering of stock items to save preparative costs, or combining jobs in flexible manufacturing systems to reduce tool-switching times. Although closely related, the maintenance activity combination problem has a special structure which requires a specific analysis (in the joint-ordering problem, one also has to decide on the number to be ordered; in the job-grouping problem, the timing aspect is not important).

Our approach first of all focuses on cost-effectiveness. This way, any combination can arise only when it yields savings gained from a supposed overlap in the individual maintenance activities (such as savings on travelling to and from an oil-producing platform). Secondly, we derive a methodology which is consistent with maintenance optimization in the concept phase, in the sense that equivalent data, models, and characteristics are used. Finally, as will be seen in the forthcoming sections, we rely on only mild mathematical assumptions, which allows quite a wide application.

3. Problem formulation and analysis

3.1 *Definition of the problem*

Consider n independent maintenance activities which have been individually planned at dates $t_1, \ldots, t_n$ within a planning horizon $[\tau_0, \tau_1]$. In the combination problem of this paper, one has the freedom to alter any of these planned execution dates, so as to ensure joint execution of one or more activities. Each activity, however, should be executed within the planning horizon. The time required to execute a single activity is short compared to the planning horizon, and is therefore left out of consideration. Once the plan has been made, one implements the results. Later on, one may make a new plan, but that is a new instance of the problem.

We first consider a heuristic approach to this problem which is similar to Liang's (1985) approach. Suppose that each activity has a window within which the execution date may be varied to enable combination.

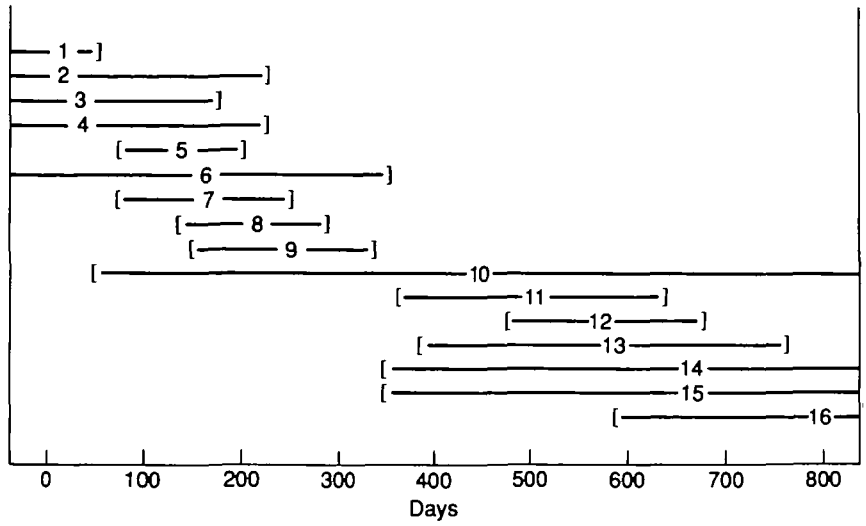


FIG. 1.

Figure 1 presents an example of 16 activities; the activity number is positioned at the planned execution date. Although the figure is helpful as a first step, it is unclear which activities should be combined. One may optimize the number of activities grouped, but this may produce several solutions (compare e.g. $\{\{1, 2, 3, 4, 6\}, \{5, 7, 8, 9, 10\}, \{11, ..., 16\}\}$ with $\{\{1\}, \{2, ..., 9\}, \{10, ..., 16\}\}$). Next, one may minimize the sum of absolute deviations from the planned execution dates, but this can no longer be done by hand and is somewhat arbitrary. Altogether, such a simple approach does not allow a balancing between deviations from the planned execution times with the savings of combined execution and it does not come up with a 'best' solution. Therefore we introduce a mathematical model which structures the problem and allows a cost optimization.

3.2 *A mathematical approach to the combination problem*

Here, two types of costs are considered in the planning: execution costs and penalty costs for deviating from original execution dates; the approach yields the optimal balance between both. We will later show how the penalty costs can be derived from so-called deterioration costs. These represent the expected costs incurred from deterioration of the system(s) to be maintained, and constitute the reason for doing the maintenance activity. They allow us to capture a number of optimization models. We make the following assumptions:

- Cost savings, because of joint execution, are non-negative and depend only on which activities are combined (and so not on when they are executed).
- The penalty cost for activity i for deviating x time units (x is positive when the activity is delayed and negative when it is brought forward) from the planned execution date t_i is $h_i(x)$, where $h_i(\cdot)$ is a non-negative convex function with $h_i(0) = 0$.

Let A denote the set of all activities considered, numbered from 1 to n following their planned execution date. Any combination of activities corresponds to a subset of A. A *partition* $\mathcal{P}$ is a collection of mutually exclusive subsets $S_1, \ldots, S_k$ which cover all activities, i.e. $S_i \cap S_j = \varnothing$, for all $i \neq j$, and $S_1 \cup \cdots \cup S_k = A$. A *combination structure* is a partition of A with the requirement that all activities within each subset are jointly executed. For each subset $S \subseteq A$ we denote by Δc_S^g the savings obtained from joint execution of all activities in S. The total cost incurred by combining execution of all activities in S and executing them at time t, is expressed by the function $h_S(\cdot)$, which is defined by

$$h_S(t) \equiv \sum_{i \in S} h_i(t - t_i) - \Delta c_S^g.$$

Notice that $h_S(\cdot)$ is also a convex function. Also, since it is decreasing left of $\min_{i \in S} t_i$ and increasing right of $\max_{i \in S} t_i$, it has a unique minimum between these dates. Denote the minimum by t_S^* and the corresponding cost value by h_S^*. We will call combining the activities in S *cost-effective* if $h_S^* < 0$. Analogously, splitting S up into two subsets S_1 and S_2 is called *cost-effective* if $h_S^* > h_{S_1}^* + h_{S_2}^*$. The total costs associated with a combination structure $\mathcal{P}$ are given by $\sum_{S \in \mathcal{P}} h_S^*$ and a combination structure will be called *optimal* if it minimizes this sum, over all partitions.

The combination problem consists of finding an optimal combination structure. Since a combination structure corresponds to a set partition, of which there are a finite number, there should be an optimal one. Hence, the problem can be formulated as a set-partitioning problem. In principle, such a problem can be solved by enumerating all set partitions and determining the associated cost for each combination in the partition. The number of set partitions, however, grows exponentially in the number of activities; so this is not a practical approach when there are many activities. In the sequel, we therefore investigate how to reduce this number.

3.3 *Determination of the penalty costs*

The combination method presented in this paper requires only convexity of the penalty functions, but not a specific form. Penalty functions represent the expected costs for deviating from the original plan, and the convexity property implies that they are larger the more one deviates. Penalty functions are crucial to our method, yet they may be difficult to obtain. The simplest solution is to use the absolute deviation from the planned execution date, multiplied by a scaling factor, as the penalty (note that this yields a convex function). If the scaling factor is taken low enough, the method will not only maximize the number of activities combined, but also minimize the sum of the deviations from the planned execution dates. In this way the heuristic method from Section 3.1 is incorporated.

Scaled time, however, is a somewhat arbitrary penalty. One may ask questions to obtain real penalty costs—a technique applied by Christer (1982) to optimize inspection intervals (one should be careful in the question formulation, and all consequences should be taken into account; it may be better to ask for the deterioration costs). For instance, one may ask for the utmost deviations from the

planned execution date for joining a combination cost-effectively (at these points, the penalty costs equal Δc^p) and use a linear interpolation and extrapolation for all other points. This approach may be applied to individual activities, but also to classes of activities, where data for one characteristic activity are used for all other activities in the class.

Another approach is to derive the penalty functions from the optimization model which has been used to determine the planned execution date. Here we take a long-term perspective and assume that each activity is carried out regularly, against costs c_i^p, and that $\mu_i(x)$ represents the expected deterioration cost, i.e. the expected cost incurred during the x time units since the last execution of the activity. Long-term optimization of the interval between executions is easily achieved by applying renewal theory, obtaining the long-term average cost $\phi_i(t)$ as function of the interval length t:

$$\phi_i(t) = \frac{c_i^p + \mu_i(t)}{t}.$$

Let t^* denote the optimum interval and $\phi^* = \phi(t^*)$ the associated long-term average cost. To specify the penalty cost for deviating from t^*, we distinguish between two cases. In the first case, called short-term shift, we change the execution interval only once from t^* to $t^* + x$ (x may be either positive or negative), implying that the next execution date remains the same and that the time interval to it equals $t^* - x$. In the second case, called long-term shift, we replan all future execution dates as well, so that all following execution intervals again equal t^*. In the first case, the penalty cost amounts to

$$h_i^A(x) = \mu_i(t^* + x) + \mu_i(t^* - x) - 2\mu_i(t^*),$$

while in the second it is

$$h_i^B(x) = \mu_i(t^* + x) - \mu_i(t^*) - x\phi^*.$$

The latter is explained as follows. The quantity $\mu_i(t^* + x) - \mu_i(t^*)$ represents the expected deterioration cost incurred by deferring the activity from t^* to $t^* + x$. The term $x\phi^*$ represents the savings gained because we can defer all future executions by a time x, which we value with the minimum average cost rate ϕ^*. Notice that, for both cases, we have $h(0) = 0$. The convexity of $h_i^A(\cdot)$ and $h_i^B(\cdot)$ is induced by the convexity of $\mu(\cdot)$; further, $h_i^A(\cdot)$ is even symmetric around zero.

Several models fit within this structure. For example, in the block-replacement model (BRM; see Barlow & Proschan 1965) we have $\mu(t) = c^f h(t)$, where $h(\cdot)$ indicates the renewal function which, under not too stringent conditions (c^f/c^p large enough, Weibull lifetime distribution with shape parameter > 1), is locally convex ($h'(t)$ is increasing in t up to about the mean life; if c^f/c^p is large enough, then the block replacement minimum will be well before that value). The minimal-repair model fits the specification as well. In this case we have

$$\mu(t) = c^r \int_0^t r(x)\,dx,$$

where $r(\cdot)$ denotes the failure rate (of the failures requiring a minimal repair) and c^r the failure repair costs. The convexity of $\mu(\cdot)$ is implied by the increasingness of $r(\cdot)$.

The final example of a model that fits into the structure, is the so-called modified block replacement model (MBRM; see Dekker & Roelvink 1991). This is a mixture of the BRM and the Age Replacement Model (ARM) for one component. The MBRM takes the BRM as long-term reference model, yielding the optimum block replacement interval t_b^* and associated minimum costs ϕ_b^*; however, the MBRM takes into account the actual age (which is set back to zero upon a failure replacement), to calculate local deterioration costs and the replacement time. Whereas, in the BRM, we would replace a component preventively if

$$c^f \mu'(t) - \phi_b^* = 0,$$

it is replaced in the MBRM, if

$$c^f r(x(t)) - \phi_b^* = 0,$$

where $r(\bullet)$ denotes the failure rate and $x(t)$ the component age t time units since the last preventive replacement. Let x^{rc} be the replacement age. In the short-term plan, we take the age at time τ_0, say x^0, as a basis, and suppose that no failure occurs within the planning horizon. Hence the planned execution time equals $\tau_0 + x^{rc} - x^0$. Should a failure occur, a new plan is made with the new age. The associated penalty cost is

$$h^A(x) = \int_0^x c^f [r(x^{rc} + y) - r(x^{rc} - y)] \, dy.$$

In case of a long-term shift, the penalty cost is

$$h^B(x) = \int_0^x c^f r(x^{rc} + y) \, dy - x\phi_b^*.$$

The MBRM has two advantages. First it yields a lower average cost than the optimal block-replacement policy, and it can be extended to group replacement (see Dekker & Roelvink 1991). Secondly, the penalty functions are much more easily calculated than those for the BRM model, since the latter require calculation of the renewal function, which in general has to be approximated.

The pure age replacement model is difficult to capture within the framework presented here, since planning has to be conditioned on whether a component fails within an interval or not, and if it fails, on when. Even for a few components, a large number of cases have to be considered. In this respect, the MBRM is the preferred compromise between the ARM and the BRM.

3.4 *Discussion of the combination savings*

We assumed that the combination savings are a function of the activities combined. However, specifying the saving costs for each combination is a considerable administrative task, because realistic numbers of activities are tens or more. In fact, it is a major problem to identify potential execution savings. Therefore we make a further simplifying assumption. We assume that all the maintenance activities of interest can be divided into groups, where each group shares the same preparative

work, unique for that group, while activities from different groups have no common elements, and therefore have no savings because of joint execution. This assumption implies that we need to consider combining activities only within one group, and that the savings from combining k activities equal $(k-1)\Delta c^p$, where Δc^p is the cost involved in the preparative work. This savings structure is not that difficult to implement, if one can identify the preparative work (e.g. one groups the activities per unit). Signalling the combination potential can be done by defining one extra table in the database (for the group composition) or by using an already existing hierarchy. The number of activities within each group may still be considerable, but not too large (say up to 100).

3.5 *Execution windows*

Another advantage of the constant-savings assumption is that it allows us to define a number of simple rules to indicate whether combination needs to be considered. We will develop them in the following lemmas. We start by recalling an immediate property of convex functions which we will use implicitly in the sequel.

LEMMA 1 Suppose that f and g are convex functions with minima at t_f^* and t_g^*, *where* $t_f^* < t_g^*$. Then $f+g$ is convex as well and, for its minimum at t_{f+g}^*, we have $t_f^* < t_{f+g}^* < t_g^*$.

Notice that, since $h_i(\bullet)$ is convex and the execution savings per activity are constant, we can define an interval $I_i \equiv [t_i + x^-, t_i + x^+]$ around t_i, by solving for x^+ and x^- from $h_i(x) = \Delta c^p$. This interval can be considered as an execution window for the combination to remain cost-effective, which justifies the heuristic approach from Section 3.1. Notice that the interval width is independent of t_i, so it can be stored in a database. The interval can be used to reduce the number of combinations that need to be considered, as the next lemmas show.

LEMMA 2 Combining activities i and j can only be cost effective if the intervals I_i and I_j overlap.

Proof. See the Appendix.

Notice that, for a combination $\{i,j\}$ to be cost-effective, it is not required that $t_i \in I_j$. The following lemma extends Lemma 2 to a set of activities.

LEMMA 3 A combination S can only be part of an optimal combination structure if all activity intervals within the set overlap, i.e. if $\bigcap_{i \in S} I_i \neq \varnothing$. The optimum execution time t_S^* lies within the intersection.

Proof. See the Appendix.

In the following, we assume that all activities have been numbered following their originally planned execution date. Thus $i < j$ implies $t_i \leqslant t_j$.

LEMMA 4 A combination $V = \{i_1, \ldots, i_n\}$, with $i_j < i_h$ for $j < h$, cannot be part of an optimal combination structure if it has a subset S of successive activities, say

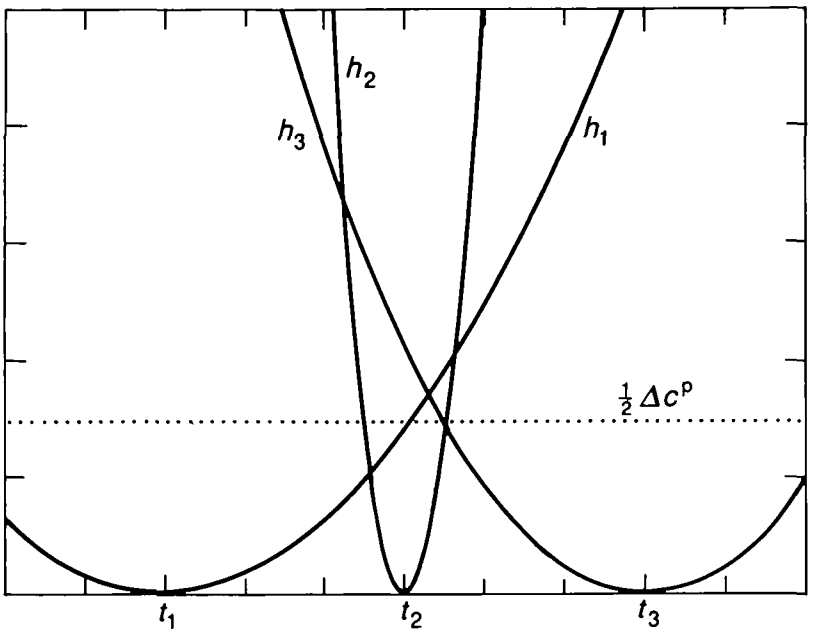


FIG. 2.

$S = \{i_j, \ldots, i_l\}$, which can be split up cost-effectively into two subsets $S_1 = \{i_j, \ldots, i_k\}$ and $S_2 = \{i_{k+1}, \ldots, i_l\}$ of successive activities.

Proof. See the Appendix.

This lemma implies that, if a combination can be split up cost-effectively into two groups of successive activities, then it cannot be extended with earlier or later planned activities to form an optimal combination. The restriction to later or earlier planned activities is necessary, as the example illustrated in Fig. 2 shows. It will be clear that combining activities 1 and 3 is not cost-effective (the penalty functions h_1 and h_3 intersect above the line $\frac{1}{2}\Delta c^p$). Yet adding activity 2 to the combination of 1 and 3 yields the optimal combination structure (notice that combination $\{1, 2, 3\}$ is better than e.g. $\{1, 2\}$, since h_3 is well below Δc^p around t_2).

Similarly it is not true that combinations that are part of the optimal structure always consist of consecutive activities, as the counterexample depicted in Fig. 3 shows. Notice that combining 1 with 2, or 3 with 4, is not cost-effective. Combining 2 and 3 is, but its total cost savings are less than Δc^p. Combining 1 and 3 has a total cost savings larger than $\frac{1}{2}\Delta c^p$, and the same holds for 2 and 4. Hence $\{\{1, 3\}, \{2, 4\}\}$ is the optimal combination structure.

4. A set-partitioning algorithm to determine the optimal combination structure

In this section, we present an algorithm to determine an optimal combination structure. We assume that, for each activity i, the planned execution date t_i and the interval I_i have already been determined. The algorithm consists of the following two phases:

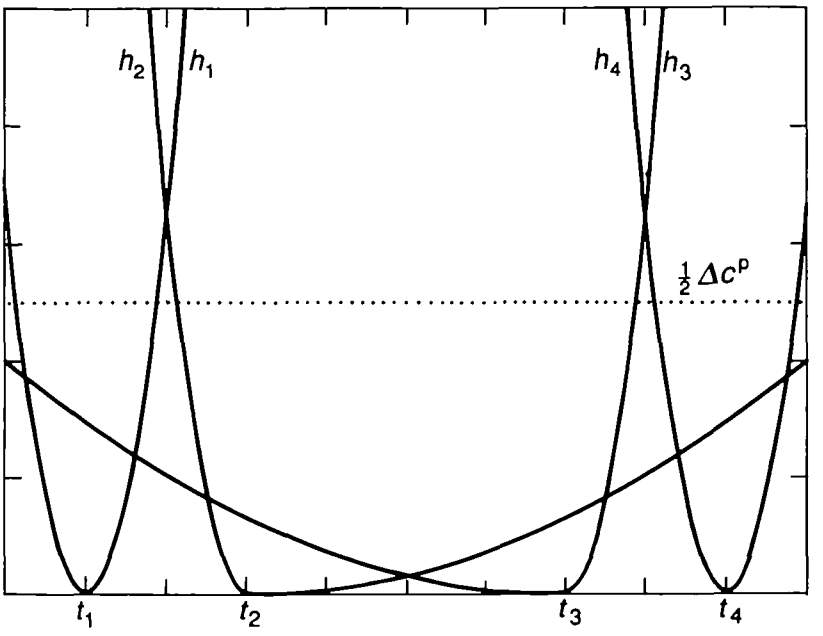


FIG. 3.

(1) Generate a list of combinations that may be present in the optimal combination structure, and determine for each combination the total associated costs.
(2) Apply a set-partitioning algorithm to this list to determine the optimal combination structure.

Phase 1 is achieved by enumerating combinations according to an increasing number of activities, concurrently applying Lemmas 2 to 4. More formally, we state the algorithm as follows.

(a) Let list 1 consist of all single-activity combinations.
(b) Suppose that list $k-1$, consisting of candidate combinations with $k-1$ activities, has already been constructed, and let

$$I_S \equiv \bigcap_{i \in S} I_i$$

for each combination S. List k is constructed as follows.
  (b1) For each combination $S = \{i_1, \ldots, i_{k-1}\}$ from list $k-1$, consider extension with every activity i_k for which $i_k > i_{k-1}$, and let $V = \{i_1, \ldots, i_k\}$.
  (b2) If $I_S \cap I_{\{i_k\}} = \varnothing$, then continue with the next extension. Otherwise check whether $\{i_2, \ldots, i_k\}$ is in list $k-1$. If it is not, then continue with the next extension; if it is, then evaluate h_V^*. If $h_V^* \leqslant h_S^*$, then add V to list k; otherwise, continue with the next extension.
  (b3) If all extensions of S have been considered, then continue with the next combination of list $k-1$.
(c) If all combinations of list $k-1$ have been dealt with, then list k is completed. If $k < n$, then start making list $k+1$; otherwise stop.

Any set S of the optimal combination structure, say $S = \{i_1, ..., i_k\}$ can be generated by successively adding i_l to $\{i_1, ..., i_{l-1}\}$, for $l = 2, ..., k$. Following Lemmas 2–4, all these additions have to be cost-effective. Thus the algorithm generates all sets that can be part of the optimal combination structure. The largest part of the computational effort is in determining h_S^* for a set S, which can be done by applying e.g. bisection or Newton–Raphson (if an explicit expression for $h_S'(t)$ is available). The number of sets that need to be considered depends on the problem at hand, and could still equal the number of possible sets generated by n elements, namely 2^n, minus $n + 1$ (i.e. n single-activity sets and the empty set). In the trivial case when all execution dates are equal, the algorithm has to generate all combinations (Section 6 presents a way to circumvent this).

Following phase 1, the optimal combination structure is determined from the remaining number of candidate combinations by solving a set-partitioning problem. Let A denote the set of original activities, as before, and let C be the set of remaining combinations, with m the total number of elements in C. Now recall that optimal execution times t_S^* for the combinations S in C are already known, since these have been computed in step (b2) of phase 1. Consequently the minimal costs g_S^* are known as well; let g denote the the $m \times 1$ vector of all these costs. Applied to maintenance activity combination, the set-partitioning problem then may be defined as that of identifying a partition $\mathcal{P}$ of k combinations $S_i \in C$ $(i = 1, ..., k)$ satisfying

$$\bigcup_{i=1}^{k} S_i = A, \qquad S_i \cap S_j = \varnothing \quad \text{for all } i \neq j \in \{1, ..., k\},$$

such that $\sum_{i=1}^{k} g_{S_i}^*$ is minimal (see also Syslo *et al.* 1983). In general, this is achieved by representing C as an $n \times m$ boolean matrix $\boldsymbol{C}$, where rows denote individual activities and columns indicate the activities that are contained in each combination of C. For instance, let $n = 4$ and $C = \{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$, so that $m = 7$. Then

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Note that the identity matrix is always contained in $\boldsymbol{C}$, ensuring the existence of a feasible solution (though against zero savings). Now the optimal partition $\mathcal{P}$ of A is found from $\boldsymbol{C}$ by solving for the boolean vector $\boldsymbol{x}$ in the problem

$$\text{minimize} \quad \boldsymbol{g}^{\top}\boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{C}\boldsymbol{x} = \mathbf{1}_n,$$

where $\mathbf{1}_n$ indicates the vector with all components equalling 1. Clearly the above constitutes a 0–1 programming problem, which could, to begin with, be solved by using ordinary branch-and-bound algorithms. However, the special structure of the problem allows one to apply dedicated algorithms, as can be found in Syslo *et al.* (1983). Some results of the algorithm are shown in the following example.

5. Example

Consider 16 maintenance activities which have been planned over a two-year period. Associated with each activity is a Weibull lifetime distribution for the component addressed, with scale parameter α_i (in days) and shape parameter β_i. Each activity is individually optimized with respect to its execution interval, following the MBRM. Here we consider an actual realization of the individually planned activities, where activity 1 is just due, activity 2 three days from now, and so on (column t_i); the age at the planned date is given by x_i^{rc}. The costs of corrective maintenance are given by c_i^f (repair plus down-time), and those of preventive replacement by c_i^p (repair only). Table 1 summarises the data (based on real data, though somewhat simplified). We consider combining under the short-term-shift assumption. In this case, the penalty function $h(\cdot)$ is given by

$$h(x) = \int_0^x c^f[r(x^{rc} + y) - r(x^{rc} - y)]\,dy,$$

where $r(x) = (\beta/\alpha)(x/\alpha)^{\beta-1}$ $(x > 0)$, for a Weibull (α, β) distribution.

With an assumed saving of 15 cost units (i.e. about 10% of the average preventive-maintenance cost) for a combination of 2 activities (so 30 for three, 45 for four, etc.), the corresponding penalty cost functions determine the same admissible execution windows as depicted in Fig. 1 (the example is the same as the one in Section 3.1). One may interpret this figure as follows, taking activity 1 as an example. The more execution of activity 1 is deferred with respect to its optimal timing (day 0), the more this will cost, because corrective maintenance costs increase (in expectation). This increase is outweighed by the savings of combining, as long as we remain within the interval of 0 to 20 days. Outside this interval, activity 1 cannot be combined

TABLE 1
Example data for combining 16 activities

Activity	α_i (days)	β_i	c_i^f	c_i^p	t_i (day)	x_i^{rc} (days)
1	1620	1.70	4610	60	0	167
2	2380	1.70	9090	120	3	239
3	1900	2.00	1380	180	32	731
4	2850	2.00	1510	90	37	736
5	1620	1.70	8600	300	135	302
6	2850	2.00	1510	180	160	1062
7	1950	1.25	4530	60	168	202
8	1350	1.75	2480	180	212	379
9	1800	1.50	3780	180	231	398
10	3200	1.50	580	30	448	746
11	1900	2.00	1380	180	495	731
12	1750	1.70	3310	450	575	742
13	2850	1.75	1810	120	580	747
14	3900	1.75	1130	60	668	904
15	3900	1.75	1130	60	668	904
16	2850	2.00	1510	120	795	848

cost-effectively with any other activity. On the other hand, cost-effective combination possibilities involving activity 1 must have an execution moment that is part of activity 1's admissible execution window.

Applying the three lemmas that reduce the number of promising combinations, only 200 out of $2^{16} - 17 = 65519$ possibilities need to be considered. Evaluating these by our algorithm results in:

- the optimal combinations and the savings per combination
- the corresponding optimal moments of combined execution.

In this case, we calculated that optimal combining allows a reduction of $11 \times 15 = 165$ cost units on the preventive maintenance budget of 2370 (so 7.0%). The corresponding penalties for changing execution times amount to 10.40 cost units, which results in a total savings minus penalty costs of 154.60 (i.e. 6.5%). To accomplish this, the following schedule should be executed:

activities 1 to 4 at day 6, saving $3 \times 15 - 1.00 = 44.00$ cost units (1.9%),
activities 5 to 7 at day 144, saving $2 \times 15 - 1.80 = 28.20$ cost units (1.2%),
activities 8 and 9 at day 222, saving $1 \times 15 - 0.40 = 14.60$ cost units (0.6%),
activities 10 to 13 at day 547, saving $3 \times 15 - 4.60 = 40.40$ cost units (1.7%),
activities 14 to 16 at day 732, saving $2 \times 15 - 2.60 = 27.40$ cost units (1.2%).

Compare this solution with the heuristic solution from Section 3.1: {1, 2, 3, 4, 6}, {5, 7, 8, 9, 10}, and {11 ,..., 16}, which are executed at days 12, 171, and 589 against penalty costs at 9.6, 26.0, and 24.7. Hence the total savings minus the penalty costs amount to 134.7, which is 13% less than optimal.

All calculations were performed with a Turbo PASCAL implementation, using a set-partitioning algorithm from Syslo *et al.* (1983). To obtain a rough indication of the reduction power (with respect to the combinations remaining) and elapsed execution time (on an 8 MHz PC-AT machine), the above example was evaluated for different values of the savings. Table 2 summarizes the results so far.

TABLE 2
Some global results for 16 example activities.

Δc^p	# comb. evaluated	Total savings	Response time (sec)
10	143	94.7 (4.0%)	97
15	200	154.6 (6.5%)	108
20	297	209.8 (8.9%)	181

6. Discussion and extensions

The method presented here provides an optimal solution for the combination problem, if penalty functions can be defined for each activity, and if combination savings consist of fixed preparative costs. Two objections may be made against the method.

First, one may argue that in practice it is difficult to obtain real penalty functions

for the individual activities, and that no planned execution date is determined via an optimization model. The latter is certainly true for present practice, but not necessarily for the future. The first aspect is primarily a result of balancing the time spent on optimization of the activity against the benefits obtained. The method presented is flexible enough to work with various penalty functions—even a function that simply represents scaled time deviations. The method further shows which data one needs to solve the combination problem. In general, data shortage is often a chicken-and-egg problem: if you do not know the value of data, you will not spend much effort on getting them. Decision support systems, incorporating models like the one presented here, help in breaking through that circle, as they can also indicate the value of data.

Another objection against the method is that the number of combinations to be evaluated may grow exponentially in the number of activities. This is true: although Lemmas 2–4 can reduce the number significantly, there may still remain a very large number of possibilities which all have to be stored in memory before set-partitioning can take place. Yet this disadvantage primarily originates from the problem rather than from the method, and one has to look for heuristics unless more simplifying assumptions are made.

Despite these objections, there are almost no alternatives to the method proposed here. The heuristic method presented in Section 3.1 works well in combining few activities, but falls short if there are a number of them. Furthermore, the mathematical method has some advantages: consistency with maintenance optimization models, an explicit focus on cost-effectiveness, and quite a wide applicability. Also, the method is easily extended in the following ways.

- Constraints on how many activities may be combined or on forbidden combinations can both be incorporated by deleting the sets violating the constraints in the first phase of the algorithm. The only requirement is that a constraint formulated on a set should hold equally for all supersets.
- Constraints can be imposed on execution times. Here, we restrict the optimization of the combined execution time t_S of a combination S to some time constraints. It is easily checked that Lemmas 2 and 3 remain valid, while Lemma 4 needs further study; checking all combinations is always an alternative.
- Corrective maintenance activities can be included by planning them at the start of the planning horizon and defining a cost rate (which should be increasing) for deferring the repair.

Finally, the total cost function $h_S(\bullet)$ for a given combination S can also be used to determine the importance of delaying the execution of S. Splitting $h_S(\bullet)$ into the individual penalty functions gives additional information on which activities contribute most to the savings of the combination. In fact, $h_S(\bullet) - h_S^*$ serves as a penalty cost function for S

One can therefore fix certain combinations beforehand and use S, t_S^*, and $h_S(\bullet)$ as input, instead of data on the individual activities. This opens the possibility for various heuristics, e.g. applying the method first to a subset and then combining the outcomes with the remaining activities. It also allows interactive planning, in which the user defines some combinations and gets feedback on the total associated costs.

7. Conclusions

In this paper we analysed a problem of combining maintenance activities, and presented a general method to determine optimal combination structures, which can enhance the cost-efficiency of maintenance planning. The method is fully consistent with several optimization models to determine individual execution times. Application depends on whether future maintenance management information systems can provide the necessary data.

Appendix: proofs

Proof of Lemma 2

Suppose that the intervals do not overlap, so that any point t falls outside one interval, say interval I_i. Then $h_i(t - t_i) > \Delta c^p$ and therefore $h_{\{i,j\}}(t) > 0$ for all t; hence $h^*_{\{i,j\}} > 0$. □

Proof of Lemma 3

Suppose that $\bigcap_{i \in S} I_i = \varnothing$. We will show that any combination structure containing the set S can be improved by splitting S up. Because the intervals do not overlap, the minimum t^*_S lies outside at least one of them, say I_j. Hence

$$h_j(t^*_S) - \Delta c^p > 0.$$

Splitting S up into $\{j\}$ and $S \backslash \{j\}$ lowers the costs, since (notice that $h^*_{\{i\}} = 0$, for all $i \in A$)

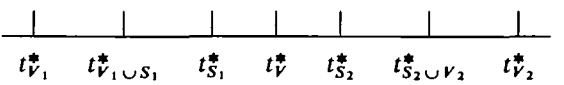
$$h^*_{S\backslash\{j\}} + h^*_{\{j\}} < h_{S\backslash\{j\}}(t^*_S) + h_j(t^*_S) - c^p = \sum_{j \in s} h_j(t^*_S) - (|S| - 1)\Delta c^p = h^*_S. \quad \square$$

Proof of Lemma 4

Let $V_1 = \{i_1, \ldots, i_{j-1}\}$ and $V_2 = \{i_{l+1}, \ldots, i_n\}$. Then $V = V_1 \cup S_1 \cup S_2 \cup V_2$. We will show that V can be split up cost-effectively into $V_1 \cup S_1$ and $S_2 \cup V_2$—in other words, that the total penalty costs reduce by more than Δc^p if we shift the execution dates of activities in $V_1 \cup S_1$ from t^*_V to $t^*_{S_1 \cup V_1}$, and of activities in $V_2 \cup S_2$ from t^*_V to $t^*_{S_2 \cup V_2}$. To this end, first notice that

$$t^*_{V_1} \leqslant t^*_{S_1 \cup V_1} \leqslant t^*_{S_1},$$

from Lemma 1. A similar result holds for S_2 and V_2. The next step is to consider the position of t^*_V with respect to $t^*_{S_1}$ and $t^*_{S_2}$. Let us first assume that $t^*_{S_1} \leqslant t^*_V \leqslant t^*_{S_2}$. All the relationships are shown in the following diagram.



In this case, we will first shift the activities of $V_1 \cup S_1$ from t^*_V to $t^*_{S_1}$ and then to $t^*_{S_1 \cup V_1}$. The activities of $V_2 \cup S_2$ are shifted in an analogous way. Notice that shifting execution

of activities in V_1 from t_V^* to $t_{S_1}^*$ is cost-effective:

$$\sum_{i \in V_1} h_i(t_V^* - t_i) \geqslant \sum_{i \in V_1} h_i(t_{S_1}^* - t_i),$$

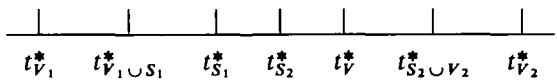
and the same holds for shifting activities of V_2 from t_V^* to $t_{S_2}^*$. Since S_1 and S_2 do not combine cost-effectively, we have

$$\sum_{i \in S} h_i(t_V^* - t_i) - \Delta c^p > \sum_{i \in S_1} h_i(t_{S_1}^* - t_i) + \sum_{i \in S_2} h_i(t_{S_2}^* - t_i).$$

Hence

$$\begin{aligned} h_V^* &= \sum_{i \in V} h_i(t_V^* - t_i) - (|V| - 1)\Delta c^p \\ &> \sum_{i \in V_1} h_i(t_{S_1}^* - t_i) + \sum_{i \in S_1} h_i(t_{S_1}^* - t_i) + \sum_{i \in S_2} h_i(t_{S_2}^* - t_i) \\ &\quad + \sum_{i \in V_2} h_i(t_{S_2}^* - t_i) - (|V| - 2)\Delta c^p \\ &\geqslant \sum_{i \in S_1 \cup V_1} h_i(t_{S_1 \cup V_1}^* - t_i) - (|S_1 \cup V_1| - 1) \\ &\quad + \sum_{i \in S_2 \cup V_2} h_i(t_{S_2 \cup V_2}^* - t_i) - (|S_2 \cup V_2| - 1) \\ &= h_{S_1 \cup V_1}^* + h_{S_2 \cup V_2}^* \end{aligned}$$

So splitting up is cost effective. We next consider the case when $t_V^* > t_{S_2}^*$, which is shown in the following diagram (the case when $t_V^* < t_{S_1}^*$ follows from symmetry arguments).



In this case, we can no longer shift activities of V_2 to $t_{S_2}^*$. So we leave all activities of S_2 and V_2 in first instance at t_V^* and shift them later to $t_{S_2 \cup V_2}^*$. Since S_1 and S_2 do not combine cost-effectively, executing them together at $t_{S_2}^*$ involves penalty costs larger than Δc^p; hence

$$\sum_{i \in S_1} h_i(t_{S_2}^* - t_i) - \Delta c^p > \sum_{i \in S_1} h_i(t_{S_1}^* - t_i).$$

Since $t_V^* > t_{S_2}^*$ in this case it follows that

$$\sum_{i \in S_1} h_i(t_V^* - t_i) - \Delta c^p > \sum_{i \in S_1} h_i(t_{S_1}^* - t_i).$$

Activities from V_1 can also be shifted to $t_{S_1}^*$, and the remainder of the proof is similar to that of the first case. □

References

BARLOW, R. E., & PROSCHAN, F. 1965 *Mathematical Theory of Reliability.* Wiley, New York.

CHRISTER, A. H. 1982 Modelling inspection policies for building maintenance, *J. ORS* **33**, 723–32.

DEKKER, R., & SMEITINK, E. 1991 Opportunity-based block replacement. *Euro. J. Opl Res.* **53**, 46–63.

DEKKER, R., & ROELVINK, I. F. K. 1991 Marginal cost criteria for group replacement. KSLA Publication no 91.090. Submitted for publication.

LIANG, T. Y. 1985 Optimum piggyback preventive maintenance policies. *IEEE Trans. on Reliab.* **34**, 529–38.

MANN, L., & BOSTOCK, H. H. 1983 Short-range maintenance planning scheduling using network analysis. *Hydrocarbons Proc.* **60**, 97–101.

PHILLIPS, K. 1979 Aspects of job scheduling. *J. of Eng. for Ind.* **101**, 17–22.

SHERIF, Y. S., & SMITH, M. L. 1981 Optimal maintenance models for systems subject to failure—a review. *Nav. Res. Logist. Q.* **28**, 47–74.

SYSLO, M. M., DEO, N., & KOWALIK, J. S. 1983 *Discrete Optimization Algorithms with PASCAL Programs.* Prentice-Hall, Englewood Cliffs, New Jersey.

VALDEZ-FLORES, C., & FELDMAN, R. M. 1989. A survey of preventive maintenance models for stochastically deteriorating single-unit systems. *Nav. Res. Logist. Q.* **36**, 419–46.