Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

European Journal of Operational Research 172 (2006) 783–797

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Discrete Optimization

Modeling and solving the periodic maintenance problem

Alexander Grigoriev [a,*], Joris van de Klundert [b], Frits C.R. Spieksma [c]

[a] *Department of Quantitative Economics, Operations Research Group, Faculty of Economics and Business Administration, University of Maastricht, P.O. Box 616, 6200 MD, Maastricht, The Netherlands*

[b] *Department of Mathematics, University of Maastricht, P.O. Box 616, 6200 MD, Maastricht, The Netherlands*

[c] *Department of Applied Economics, Katholieke Universiteit Leuven, Naamsestraat 69, 3000 Leuven, Belgium*

Received 5 May 2004; accepted 25 November 2004
Available online 11 January 2005

Abstract

We study the problem of scheduling maintenance services. Given is a set of m machines and integral cost-coefficients a_i and b_i for each machine i ($1 \leqslant i \leqslant m$). Time is discretized into unit-length periods; in each period at most one machine can be serviced at a given service cost b_i. The operating cost of machine i in a period equals a_i times the number of periods since the last servicing of that machine i. The problem is to find a cyclic maintenance schedule of a given length T that minimizes total service and operating costs. We call this problem the periodic maintenance problem or PMP.

In this work we are interested in computing optimal solutions to instances of PMP. We investigate several formulations for PMP. Two formulations, referred to as a flow formulation and a set-partitioning formulation, appear to have good linear programming relaxations. We exploit the problem structure by showing how the column generation subproblem can be solved in polynomial time. Our work leads to the first exact solutions for larger sized problem instances, and we present extensive computational results.

Keywords: Combinatorial optimization; Scheduling; Preventive maintenance; Branch and price

1. Introduction

The planning and scheduling of preventive maintenance activities is often crucial for the cost-effectiveness of many large industrial organizations. For instance, manufacturing organizations that have highly sophisticated and complex machinery have long recognized that efforts spent

* Corresponding author. Tel.: +31 43 3883853; fax: +31 43 3884874.

E-mail addresses: a.grigoriev@ke.unimaas.nl (A. Grigoriev), j.vandeklundert@math.unimaas.nl (J. van de Klundert), frits.spieksma@econ.kuleuven.ac.be (F.C.R. Spieksma).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2004.11.013

on preventive maintenance can contribute significantly towards an efficient running of the organization. Also in service organizations (like medical facilities or governmental institutions), preventive maintenance is regarded as an important activity that can help to reach the organization's performance goals. However, the costs associated with preventive maintenance can be significant: there are not only costs involved with the maintenance itself, also the costs of production losses during the maintenance have to be taken into account. Computerized maintenance management systems (CMMS's) are becoming increasingly popular as a tool to increase machine-availability and more generally, to improve control over the maintenance activities. Software vendors (see for instance http://www.plant-maintenance.com/index.shtml) offer packages that usually includes a scheduling module that suggests (among other things) when to service which unit (or machine). This decision is seen as a re-occurring event, i.e., it is expected that a schedule is of a cyclic nature, and hence will be executed repeatedly.

There is a huge amount of literature available on preventive maintenance. However, approaches in literature usually are of a stochastic nature where a probability distribution is used to describe the failure properties of a machine (see for instance Gertsbakh and Gertsbakh, 2000). In this work we take a different, completely deterministic, approach (see Wagner et al., 1964, for an early reference). More specifically, we deal with the problem of cyclically scheduling maintenance activities under a certain given cost-structure assuming a fixed cycle length. A precise description is given in the next subsection.

1.1. Problem description

We consider the following problem. There are a number of machines M_i, $i \in \{1,\ldots,m\}$, and there is a time interval $T = \{1,2,\ldots,T\}$ with $T \geqslant m$. During each period of the time interval T, at most one machine can be serviced. When machine M_i is serviced, a given, non-negative, servicing cost of b_i is incurred, regardless of the period. At time moment $t \in T$, a machine M_i that is not serviced during some period is in operation and incurs an operation cost of $j_i(t) \times a_i$, where a_i is a given positive integer, and where $j_i(t)$ is the number of periods elapsed since last servicing machine M_i, $i \in \{1,\ldots,m\}$. Observe that we assume here that the operating costs of a machine increase linearly with the number of periods elapsed since last servicing that machine. The problem is now to determine a maintenance schedule, i.e., to decide for each period $t \in T$ which machine to service (if any), such that total servicing costs and operating costs are minimized.

Below, when motivating the problem, we introduce several good reasons to view such problem in a cyclic context. In such a context, it is assumed that the maintenance schedule will be executed repeatedly. Thus, in period $k \times T + t$ ($k \in N$, $t \in T$), the same machine that was serviced in period t will be serviced again. In addition, the cost will be considered in this infinite horizon context. Consequently, the cost of a maintenance schedule is calculated by summing over all $t \in T$ the total of the servicing costs incurred in period t and the operating costs incurred by the machines which are not serviced in period t. These operating costs are defined in a cyclic context, i.e., the last maintenance service may lie in a previous execution of the maintenance schedule. We will refer to this problem as the *periodic maintenance problem* (*PMP*). Notice that in an optimal solution to PMP, each machine is served at least once. Finally, we notice here explicitly that in PMP T is considered to be an input parameter.

For ease of understanding, we now present a brief example.

Example. Let $T = 7$, $m = 3$ and the set of machines is $\{1,2,3\}$. Further, let $b_i = 1$, $i = 1,2,3$, and let $a_1 = a_2 = 10$ and $a_3 = 1$. Consider the solution $(1,2,1,2,1,2,3)$. This sequence of maintenance services is to be read as follows: in the first period, we service machine 1, in the second period machine 2, et cetera, until we service in the seventh period machine 3. Then, this sequence of maintenance services is repeated, i.e., in the 8th period we service machine 1 again, followed by machine 2 in period 9, and so on. The cost of this solution can be computed as follows. Since there is maintenance in each of the seven periods of T, and

since all service costs b_i are equal to one, the total servicing costs equal 7. For the first machine the operating costs are incurred in periods 2, 4, 6 and 7. In periods 2, 4 and 6, these costs equal 10, and in period 7 these costs amount to 20. Thus, machine 1 has a total operating cost of 50. Similarly, it can be checked that machine 2 has operating costs of $20 + 0 + 10 + 0 + 10 + 0 + 10 = 50$, and machine 3 has operating costs of $1 + 2 + 3 + 4 + 5 + 6 = 21$. Thus the total cost for this solution is 128. The reader can verify that the solution presented above is in fact optimal.

Apart from the application sketched in the introduction, PMP and variants of PMP have real-life applications with different origins such as the scheduling of maintenance services, multi-item replenishment of stock, and broadcasting of data messages over a communication channel (see the references in Section 2). In particular, the problem where the cycle length is not given, but instead a decision variable, has received quite some attention. In the remainder, we refer to the variant of PMP where T is considered to be a decision variable, as the *free periodic maintenance problem* (*FPMP*); we use T^* to denote the optimal cycle length in FPMP.

Our motivation for investigating PMP, rather than FPMP, is twofold. First of all, PMP is a practical problem. Especially in the context of constructing maintenance schedules, it is very natural to fix the cycle length to some constant such as 365, 52, 30, 7, 24 or 60. Indeed, an organization that implements a cyclic maintenance schedule will, for reasons of simplicity, ensure that the length of the cyclic schedule coincides with the size of a natural time interval such as the number of days per year, or the number of weeks per year, or the number of days per week. Further, in many practical settings, it is desirable that the cycle length T is not too large. In fact, even for instances of modest size, for example $m = 2$, $a_1 = 1$, $a_2 = a$, $b_1 = b_2 = 0$, the optimal cycle length T^* can be fairly large: for this case, $T^* \geqslant \lfloor \sqrt{2a} \rfloor$ (see Anily et al., 1998). Thus, one is interested in computing a cyclic schedule with a cycle length that is bounded from above by some reasonably small (given) integer B. In such a case, one can find the optimal $T \leqslant B$ by solving the PMP for each possible value of T not exceeding B. In both cases, the task is to find a solution of some specific cycle length that may differ from the optimal length T^*. As far as we are aware, the PMP has not been studied before.

A second motivation of our work is that we are interested in solving instances of the problem to optimality. As we shall see in Section 2, apart from Anily et al. (1998) which deals with a special case of FPMP, most research has focused on complexity results, and approximation for FPMP. From this point of view, we further explore the area of solving instances to optimality by solving them for a fixed, but not necessarily optimal, T. In addition, our results provide insight in the effect of varying T on the actual schedule and its solution, i.e., we investigate the sensitivity of the solution with respect to the cycle length.

This paper is organized as follows. In the next section we present a brief literature review. Section 3 discusses several models, and how they might be of use in solving the problem to optimality. Section 4 presents a branch and price algorithm that solves one of the models of Section 3 to optimality. In Section 5 we present computational results on instances with 3–10 machines and with a number of periods ranging from 10 to 100. Section 6 contains the conclusions.

2. Literature review

An area where preventive maintenance scheduling has been applied is in the operational planning of power generating plants. We refer to Kralj and Petrović (1988, 1995) for an overview of optimization techniques (including integer programming) in this field, and to Charest and Ferland (1993) for applying local search techniques to solve a model related to the set-partitioning model of Section 3.

Maintenance scheduling problems that involve coordinating a common resource to maintain a set of machines have been investigated by Duffuaa and Ben-Daya (1994), Hariga (1994), and Sule and Harmon (1979). An overview on these and several

other preventive maintenance scheduling problems can be found in Dekker et al. (1997).

Anily et al. (1998) consider the special case of FPMP, where $b_i = 0$ for all $i \in M$, and they describe an application in the multi-item replenishment of stock. They prove that there exists an optimal schedule that is cyclic. Further, they describe a network-flow based algorithm that has exponential complexity to solve the problem exactly. This approach allows them to solve instances with up to four machines exactly. In addition, the authors propose two lower bounds and a greedy heuristic, which performs very well. Notice however that in their problem setting, the cycle length is a decision variable, and therefore the solutions given by the heuristic may use a different cycle length then the cycle length of an optimal solution. The case with three machine and zero servicing costs is investigated in Anily et al. (1999). In this work the authors introduce an algorithm solving certain instances of the problem to optimality and for the other instances they present a heuristic algorithm with performance ratio of 1.0333.

Bar-Noy et al. (2002) and Kenyon et al. (2000) consider a generalized version of the FPMP where in each period at most M machines can be serviced. Their interest in the problem is motivated by applications that arise in broadcast scheduling. Bar-Noy et al. (2002) prove that FPMP is NP-hard. Further, they investigate lower bounds and propose a $\frac{9}{8}$-approximation algorithm. Kenyon et al. (2000) present a polynomial-time approximation scheme for FPMP with bounded service costs. The version of the problem with non-identical service times is studied in Kenyon and Schabanel (2001). Recently, Schabanel (2000) shows that the version of FPMP in which preemptions are allowed, is also NP-hard.

Brakerski et al. (2001) consider the problem of encoding a solution in such a way that the next machine to be serviced can always be found quickly, given that all service activities performed up till now are known. Brauner et al. (2001) address related scheduling problems that arise from compact encodings of solutions.

Another area that is related to the PMP is the so-called parallel machine replacement problem (see Jones et al., 1991; McClurg and Chand, 2002). This problem deals with a set of machines whose operational costs increase with age, while in each period there is the possibility to replace a machine at the expense of purchasing costs. The authors present a dynamic programming procedure to balance operational costs and purchasing costs. However, in contrast to the PMP, the parallel machine replacement problem has a fixed horizon, and is motivated from an economic perspective, incorporating salvage costs, and the discounting of costs.

We now briefly examine the PMP from a complexity viewpoint. First of all, notice that the input to PMP consists of $2m + 1$ numbers (the a_i, b_i and T). Thus, an algorithm which has the parameter T present in its running-time is not a polynomial-time algorithm for PMP. In fact, all models we present in this paper have (at least) a pseudo-polynomial number of variables. Second, the reduction in Bar-Noy et al. (2002) shows that FPMP is NP-hard even when T^* is known. This implies indeed that PMP is NP-hard as well, since it may be the case that $T = T^*$.

3. Modeling PMP

In this section we describe three formulations for PMP. Section 3.1 gives a quadratic programming formulation, Section 3.2 describes an integer programming based formulation, and Section 3.3 presents a set-partitioning formulation.

3.1. A quadratic programming formulation

Here we introduce a compact and natural, but non-convex quadratic program modeling PMP with operational costs only, i.e., we first assume $b_i = 0$ for all $i \in M$. The model uses a variable $x_{i,t} \in Z^+$, $i \in M$, $t \in T$, which represents the number of periods between the current period $t \in T$ and the last period before t when machine i has been serviced. Clearly, for any machine i, and any period t, the value of variable $x_{i,t}$ is obtained by either adding 1 to the value of $x_{i,t-1}$, or by setting it to 0. Setting the value of $x_{i,t}$ to 0 corre-

sponds to servicing machine i in period t. PMP can now be formulated as follows:

$$\min \sum_{i\in M}\sum_{t\in T} a_i x_{i,t}, \tag{1}$$

$$x_{i,t+1}(x_{i,t+1} - x_{i,t} - 1) = 0, \quad i \in M,\ t \in T \setminus T, \tag{2}$$

$$x_{i,1}(x_{i,1} - x_{i,T} - 1) = 0, \quad i \in M, \tag{3}$$

$$x_{i,t} + x_{k,t} \geqslant 1, \quad i \neq k,\ i \in M,\ k \in M,\ t \in T, \tag{4}$$

$$x_{i,t} \in Z^+, \quad i \in M,\ t \in T. \tag{5}$$

Eqs. (2) and (3) ensure the required behavior of the $x_{i,t}$ variables. Eqs. (4) imply that no two machines can be served simultaneously. Notice that if for some machine i one of the associated variables is integral, (2) and (3) together imply that all other variables corresponding to machine i are integral as well.

Since most of the available software for solving quadratic programming problems only solve the problems with quadratic objective function and linear constraints, we have not been able to solve problem instances through the formulation given above. Instead, we now linearize model (1)–(5) and take into account the servicing costs b_i:

$$\min \sum_{i\in M}\sum_{t\in T}(a_i x_{i,t} + b_i y_{i,t}), \tag{6}$$

$$x_{i,t+1} \geqslant x_{i,t} + 1 - N y_{i,t+1}, \quad i \in M,\ t \in T \setminus T, \tag{7}$$

$$x_{i,1} \geqslant x_{i,T} + 1 - N y_{i,1}, \quad i \in M, \tag{8}$$

$$\sum_{i\in M} y_{i,t} \leqslant 1, \quad t \in T, \tag{9}$$

$$x_{i,t} \in Z^+, \quad i \in M,\ t \in T, \tag{10}$$

$$y_{i,t} \in \{0,1\}, \quad i \in M,\ t \in T, \tag{11}$$

where N is a sufficiently big number (here, e.g., $N = T$ is large enough).

The binary variable $y_{i,t}$ simply takes on value 1 if we service the ith machine in period t and 0 otherwise. The objective (6) minimizes the total costs that now consist of operating costs and servicing costs. Eqs. (7) and (8) enforce the variables $x_{i,t}$ to behave in the same way as in the previous model. According to (9) we cannot service more than one machine in a single period. Restrictions (10) and (11) are the integrality constraints. We refer to the formulation (6)–(11) as QP.

Example. We illustrate model (6)–(11) with the following example. Let $T = 7$, $m = 3$ and the set of machines is $\{1,2,3\}$. A feasible solution of the formulation is depicted in Table 1.

Notice that formulation (6)–(11) involves a so-called *big N parameter* which renders the associated linear relaxation to be rather poor. For instance, by setting $y_{i,t} = 1/m$ and $x_{i,t} = 0$, $i \in M$, $t \in T$, we satisfy all constraints of the linear relaxation. The value of the objective function of this solution to the linear relaxation is equal to $T\sum_{i\in M} b_i/m$ which can be an arbitrary bad lower bound for the optimum. This explains the poor computational performance we obtained using the standard ILP-packages dealing with formulation (6)–(11), see Section 5.3.

Another weak point of this formulation is that we use the fact that the objective is to *minimize* the total operating and servicing costs. This means that not every solution that satisfies (7)–(11) is a meaningful solution to PMP. Thus, to solve the problem under maximization or mixed min–max criteria we cannot even use the linear model described above.

3.2. An integer programming formulation

We now present a formulation that contains $O(m \times T^2)$ binary variables. We introduce a

Table 1
A feasible solution

	Period ($t \in T$)						
	1	2	3	4	5	6	7
Sequence of maintenance services (machines)	1	3	1	2	1	3	2
$y_{1,t}$ (service indicator)	1	0	1	0	1	0	0
$y_{2,t}$ (service indicator)	0	0	0	1	0	0	1
$y_{3,t}$ (service indicator)	0	1	0	0	0	1	0
$x_{1,t}$ (state)	0	1	0	1	0	1	2
$x_{2,t}$ (state)	1	2	3	0	1	2	0
$x_{3,t}$ (state)	2	0	1	2	3	0	1

variable $x_i^{s,t}$, $i \in M$, $s,t \in T$, whose value equals 1 if machine i is serviced in period s, and serviced next (cyclically) in period $t+1$, and 0 otherwise. Notice that when s is the last service in T, we have that $t \leqslant s$, because of the cyclicity of the maintenance schedule. Using costs $c(s,t)$ defined as follows:

$$c(s,t) = \begin{cases} \dfrac{(t-s)(t-s+1)}{2} & \text{if } s \leqslant t, \\ \dfrac{(T-s+t)(T-s+t+1)}{2} & \text{if } s > t, \end{cases}$$

the problem can be modeled as follows:

$$\min_x \sum_{i\in M}\sum_{s\in T}\sum_{t\in T}(a_i c(s,t)x_i^{s,t} + b_i x_i^{s,t}), \tag{12}$$

subject to

$$\sum_{i\in M}\sum_{s\in T} x_i^{s,t} \leqslant 1, \quad t \in T, \tag{13}$$

$$\sum_{s\in T} x_i^{s,t} = \sum_{s\in T} x_i^{t+1,s}, \quad i \in M,\ t \in T \setminus T, \tag{14}$$

$$\sum_{s\in T} x_i^{s,T} = \sum_{s\in T} x_i^{1,s}, \quad i \in M, \tag{15}$$

$$\sum_{s\in T}\sum_{t\in T} x_i^{s,t} \geqslant 1, \quad i \in M, \tag{16}$$

$$x_i^{s,t} \in \{0,1\}, \quad i \in M,\ s \in T,\ t \in T. \tag{17}$$

Inequalities (13) express that in each period at most one machine can be serviced, equalities (14) and (15) imply that there is a next period in which a machine will be serviced, inequalities (16) say that each machine is serviced at least once, and finally (17) are the integrality constraints.

Again, the LP relaxation of this formulation is rather poor. For example, setting $x_i^{t,t} = \frac{1}{m}$ for all $t \in T$ and for all $i \in M$, and all other variables equal to 0, yields a feasible solution with zero operating costs. Notice how this solution resembles the example demonstrating the poor behavior of the LP relaxation of (6)–(11). The LP relaxation is strengthened considerably when we replace (16) by the following constraints (which are clearly valid for the ILP formulation above):

$$\sum_{s\leqslant u}\sum_{t<s} x_i^{s,t} + \sum_{s\leqslant u}\sum_{t\geqslant u} x_i^{s,t} + \sum_{t\geqslant u}\sum_{s>t} x_i^{s,t} = 1, \quad \text{for all } i \in M,\ 1 < u < T, \tag{18}$$

$$\sum_{s>1}\sum_{t<s} x_i^{s,t} + \sum_{s\leqslant T} x_i^{s,T} = 1, \quad \text{for all } i \in M, \tag{19}$$

$$\sum_{t<T}\sum_{s>t} x_i^{s,t} + \sum_{t\geqslant 1} x_i^{1,t} = 1, \quad \text{for all } i \in M. \tag{20}$$

Constraints (18)–(20) state that for every machine and for every period u, the sum of the variables corresponding to pairs (s,t) that contain period u, is one. Notice that the solution given above violates these constraints. One can view (18)–(20) as a (polynomially sized) set of valid inequalities for the formulation consisting of (13)–(17); adding these inequalities yields a strengthened formulation. Summarizing, we refer to the formulation consisting of constraints (12)–(15), (17)–(20) as the flow formulation (*FF*). In Section 5 we provide computational results showing that *FF* yields promising computational results when solving it using state of the art standard software CPLEX 7.5.

3.3. A set-partitioning formulation

Yet another formulation, using an exponential number of variables, concludes this modeling section.

Let S be the set of all non-empty subsets of T. Clearly, every $s \in S$ is a possible set of periods for servicing a machine $i \in M$. Let us call $s \in S$ a *service strategy* or simply *strategy*. For every pair consisting of a machine $i \in M$ and a strategy $s \in S$, we can compute the cost $c_{i,s}$ incurred when servicing machine i in the periods contained in s as follows: let p_s be the cardinality of s and let q_j, $j \in \{1,2,\ldots,p_s\}$, be the distances between neighboring services in s. For example, if $T = 7$ and $s = \{2,4,6\}$ then $p_s = 3$ and $q_1 = 4-2 = 2$, $q_2 = 6-4 = 2$, $q_3 = 7-6+2 = 3$. The total service and operating cost associated with machine $i \in M$ and strategy $s \in S$ is

$$c_{i,s} = b_i p_s + a_i \sum_{j=1}^{p_s}(q_j - 1)q_j/2.$$

So, in the example above the total costs of servicing machine i using strategy s is $c_{i,s} = 3b_i + a_i + a_i + 3a_i = 3b_i + 5a_i$.

Now we introduce a variable $x_{i,s}$ which has value 1 if machine $i \in M$ is serviced in the periods

contained in strategy $s \in S$, and 0 otherwise. This allows for the following set-partitioning formulation (SP):

$$\min_x \sum_{i\in M}\sum_{s\in S} c_{i,s}x_{i,s}, \tag{21}$$

subject to

$$\sum_{s\in S} x_{i,s} = 1, \quad i \in M, \tag{22}$$

$$\sum_{i\in M}\sum_{s\in S:t\in s} x_{i,s} \leqslant 1, \quad t \in T, \tag{23}$$

$$x_{i,s} \in \{0,1\}, \quad i \in M, \ s \in S. \tag{24}$$

Constraints (22) imply that one service strategy has to be selected for each machine, and constraints (23) ensure that no two strategies make use of a same period. Constraints (24) are the integrality constraints. Despite the exponential size of this integer linear program it has two important properties. First, its linear relaxation (obtained by replacing (24) by $x_{i,s} \geqslant 0$ for all i,s) is solvable in time polynomial in m and T (see Section 4). Second, computational experiments show that the linear relaxation of this integer problem is quite strong. In the next section we show how to solve SP using a branch and price algorithm.

We conclude this section by showing that the LP relaxation of SP is stronger than the LP relaxation of FF.

Theorem 1. *Let $v(FFLP)$, $v(SPLP)$ be optimal solutions of the linear relaxations of FF and SP respectively. We have $v(FFLP) \leqslant v(SPLP)$.*

Proof. Let $x^* = \{x_{i,s}: i \in M; s \in S\}$ be any solution to the LP relaxation of SP. Construct a solution $y^* = \{y_i^{u,v} : i \in M; u,v \in T\}$ to the LP relaxation of FF as follows. Let $S(u,v) = \{s \in S: u,v$ are consecutive periods in strategy $s\}$. We set $y_i^{u,v-1} = \sum_{s\in S(u,v)} x_{i,s}$.

Now let us first show that this solution is feasible. The solution y^* satisfies the flow conservation constraints (14) and (15) from its construction. Similarly, constraint (23) and the feasibility of x^* implies that (13) is satisfied. Further, it follows from constraint (22) and the construction of y^* that (18)–(20) is satisfied. We leave it to the reader to verify that the objective function values of x^* and y^* are equal.

Thus, any solution of the LP relaxation of SP can be converted to a corresponding solution of the LP relaxation of FF with the same value. This completes the proof. □

4. A branch and price algorithm for PMP

In this section we show how to solve SP using a branch and price algorithm. In Section 4.1 we show how column generation can be used to solve the LP relaxation of (21)–(24) without enumerating all variables $x_{i,s}$. Next, in Section 4.2 we propose a branching scheme that keeps the structure of the problem intact. We refer to Barnhart et al. (1998) for a general description of branch and price algorithms.

4.1. Column generation algorithm

Its linear relaxation (called $SPLP$) is obtained by replacing constraints (24) by $x_{i,s} \geqslant 0$ for all i,s. The corresponding dual problem (called SPD) is

$$\max_{u,v}\left(\sum_{i\in M} u_i + \sum_{t\in T} v_t\right), \tag{25}$$

subject to

$$u_i + \sum_{t\in s} v_t \leqslant c_{i,s}, \quad i \in M, \ s \in S, \tag{26}$$

$$v_t \leqslant 0, \quad t \in T. \tag{27}$$

The column generation procedure starts with finding a feasible solution for $SPLP$. To do that we can use, for example, a trivial integer solution where in the first T periods we service all machines one by one, and for all remaining periods we service only the machine with the largest coefficient a_i. So, in a initialization step, we generate the set of pairs $N = \{(i,s_i): i \in M\}$ where s_i is the set of periods when we service machine $i \in M$. Let us restrict the column set of $SPLP$ to N and let us call the problems restricted to N as $SPLP(N)$ and $SPD(N)$ respectively.

Next, we find an optimal solution for $SPLP(N)$ and $SPD(N)$ using an LP-solver. Thus, we obtain a primal–dual pair of solutions $(x(N),(u(N),v(N)))$. We can extend $x(N)$ to a solution of $SPLP$ by setting the remaining variables to zero. Establishing whether or not this extended solution is optimal for $SPLP$ can be done by analyzing the corresponding dual solution $(u(N),v(N))$. Optimality of $x(N)$ for $SPLP$ depends on the feasibility of $(u(N),v(N))$ in SPD. To verify whether all dual constraints are satisfied we have to solve the following pricing problem:

$$\text{Price}: \ \exists i \in M,\ s \in S \text{ such that } u_i + \sum_{t\in s} v_t > c_{is}?$$

If the dual solution $(u(N),v(N))$ satisfies all constraints of SPD, then $x(N)$ extended with zeros is an optimal solution of $SPLP$. If not, then we have found—by solving the pricing problem—a machine i and a strategy s whose reduced cost $c_{is} - u_i - \sum_{t\in s} v_t$ is negative. Thus bringing this variable into the basis will contribute to the objective function's value. Then we update N by adding this variable to it, and we iterate. The efficiency of this procedure depends to a large extent on the speed with which the pricing problem can be solved. We have the following theorem:

Theorem 2. *The pricing problem can be solved in $O(mT^3)$ time.*

Proof. We prove that for each i we need to solve an all-pairs shortest path problem on a directed graph with $O(T)$ nodes. Since this problem can be solved in $O(T^3)$ using the Floyd–Warshall algorithm (see Ahuja et al., 1993), the result follows.

Thus, let us now consider a specific machine i, and let us build the following graph $G=(V,A)$ with $V=T$ and $A=\{(p,q)\colon p\leqslant q, p,q\in V\}$.

For each arc $(p,q)\in A$ we define the following costs w:

$$w(p,q) = b_i + a_i\frac{(q-p)(q-p-1)}{2} - v_q \quad \text{if } p\neq q$$

$$\text{and } w(p,p) = b_i + a_i\frac{T(T-1)}{2} - v_p.$$

This completes the construction of G. Since by (27) $v_t \leqslant 0$ for all $t\in T$, all costs w are non-negative. Let us now establish a correspondence between a path P in G and a service strategy s for machine i. Indeed, consider any path $P=\{t_1,t_2,\ldots,t_k\}$ in G. We have the following:

Claim: If there exists a path in G from t_1 to t_k with costs less than $Q \equiv u_i + v_{t_1} - b_i - a_i\sum_{t=t_k+1}^{T+t_1-1}(t-t_k)$ then the current solution is not optimal.

Argument: Notice that Q depends only on t_1 and t_k. Consider now the cost of a path $\{t_1,t_2,t_3,\ldots,t_{k-1},t_k\}$ in V. Summing the appropriate coefficients w gives:

$$(k-1)b_i + a_i\sum_{l=1}^{k-1}\sum_{t=t_l+1}^{t_{l+1}-1}(t-t_l) - \sum_{l=2}^{k} v_{t_l}.$$

We now derive:

$$\begin{aligned}
&(k-1)b_i + a_i\sum_{l=1}^{k-1}\sum_{t=t_l+1}^{t_{l+1}-1}(t-t_l) - \sum_{l=2}^{k} v_{t_l} < Q\\
&\quad\Longleftrightarrow kb_i + a_i\sum_{l=1}^{k}\sum_{t=t_l+1}^{t_{l+1}-1}(t-t_l) - \sum_{l=1}^{k} v_{t_l} < u_i\\
&\quad\Longleftrightarrow c_{is} - \sum_{t\in s} v_t < u_i.
\end{aligned}$$

It follows that given the first and the last service period, computing a shortest path in G between the corresponding vertices determines whether there is a strategy to be added to the master problem. Hence, to solve the pricing problem for machine i we need to compute shortest paths between every pair of vertices in G. As mentioned above this can be done using Floyd–Warshall's algorithm in $O(T^3)$ operations (see Ahuja et al., 1993). □

Corollary 4.1. *The problem SPLP can be solved in time polynomial in m and T.*

Proof. The proof of the corollary straightforwardly follows from Theorem 2 and the well-known theorem by Grötschel et al. (1981), stating

> *There exists a polynomial-time algorithm for the separation problem for a family of polyhedra, if and only if there exists a polynomial-time algorithm for the optimization problem for that family.*

Since the pricing problem is nothing else but the separation problem for *SPD* we have that optimization problems *SPD* and *SPLP* are solvable in time polynomial in m and T. □

In the computations reported in Section 5 we do not use the approach by Grötschel et al. (1981). Instead, we apply the column generation procedure described above to solve the *SPLP*. Observe that this procedure goes through in case a given number of machines can be serviced in each period (instead of exactly one machine).

4.2. A branching scheme

To solve the original integer programming formulation *SP* let us introduce the following branching strategy. Notice that a traditional branching strategy that consists of setting a variable to 0 versus setting a variable to 1, would not preserve the efficient solvability of the pricing problem (see Barnhart et al., 1998). Given a linear programming solution $x_{i,s}$, define $sum_i(t) = \sum_{s\in S: t\in s} x_{i,s}$ for $i \in M$, $t \in T$.

Lemma 4.2. *If the solution is fractional, i.e., if there exists a machine $i_0 \in M$ and a strategy $s \in S$ with $0 < x_{i_0,s} < 1$, then there exists a $t \in T$ such that $0 < sum_{i_0}(t) < 1$.*

Proof. Consider machine $i_0 \in M$. Let $S(i_0)$ be the set of strategies s for which $0 < x_{i_0,s} < 1$. We say that strategy s_1 *contains* strategy s_2 if, for each period $t \in s_2$, we have that $t \in s_1$. Let $s_0 \in S(i_0)$ be a strategy that does not contain any other strategy from $S(i_0)$ (notice that such a strategy always exists). We argue by contradiction.

Assume that for all $t \in T$ the numbers $sum_{i_0}(t)$ are equal to either 0 or 1. This implies that $sum_{i_0}(t) = 1$ for all periods $t \in s_0$. Since, by (22), $\sum_{s\in S} x_{i_0,s} = 1$, and since for each $t \in s_0$ we have that $sum_{i_0}(t) = \sum_{s\in S: t\in s} x_{i_0,s} = 1$, it follows that $x_{i_0,s} = 0$ for each strategy $s \in S$ that uses a period t not used by strategy s_0. Due to the fact that s_0 does not contain any strategy from $S(i_0)$, it follows that for each $s \in S(i_0)\backslash s_0$, there exists a period $t \in s$ such that $t \notin s_0$. Consequently, $x_{i_0,s} = 0$ for all $s \in S(i_0)\backslash s_0$, and hence $x_{i_0,s} = 1$, which is a contradiction. □

Let us now describe how this branching scheme preserves the efficient computation of service strategies. Let the branching rule be simply to decide whether period $t \in T$ is used in a service strategy for machine $i_0 \in M$ (i.e., $sum_{i_0}(t) = 1$, we refer to this as branch 1) or not (i.e., $sum_{i_0}(t) = 0$, we refer to this as branch 2). Considering branch 1, this has the following consequences for the pricing problem: each arc passing t, i.e., going from some $t_1 < t$ to some $t_2 > t$ is deleted from the graph and from now on for every child node of the branching tree machine i_0 is serviced at period t. Moreover, in the graphs associated to the other machines, we delete all arcs entering node t. So, for these machines, no path will visit node t. Considering branch 2 is even easier: we simply delete from the graph all arcs entering t. Obviously, an optimal solution is not excluded by this branching rule and, from Lemma 4.2, we conclude that this rule excludes the current fractional solution.

5. Computational results

In this section we present computational results for all ILP models presented in the previous sections.

5.1. Technical details

All experimental results were obtained on an AMD Athlon computer with 2400 XP+/1GB RAM running Debian GNU/Linux 3.0 with kernel 2.4.18. All calculations, except the results reported in Table 2, were limited by 100 000 branching nodes and by 10 000 seconds CPU-time. To compute the optimal solutions for *QP* and *FF* we use the package ILOG OPL-Studio 3.5 using the CPLEX MIP Solver. In the calculations results we mean by *OPT*, *QP*- and *FF*-nodes, *QP*- and *FF*-time respectively: the average maintenance and operating cost of an optimal solution (the optimal objective value divided by T), the number of nodes in the branching tree needed by OPL-Studio for *QP* and *FF* (expressed by the parameter "MIP-nodes") and the CPU-time in seconds for *QP* and *FF* (expressed by the parameter "Solving time").

Table 2
Formulations *QP*, *FF*, and *SP* ($m = 3$, $b_i = 0$, $i \in M$)

T	a	OPT	QP-nodes	QP-time (second)	FF-nodes	FF-time (second)	SP-nodes	SP-time (second)
3	1, 1, 1	3.0	14	1	1	1	1	1
3	2, 1, 1	4.0	9	1	1	1	1	1
3	2, 2, 1	5.0	13	1	1	1	1	1
4	5, 1, 1	5.5	20	1	1	1	1	1
4	5, 2, 1	7.0	38	1	1	1	1	1
5	5, 5, 1	10.0	70	1	1	1	1	1
4	10, 1, 1	8.0	29	1	1	1	1	1
4	10, 2, 1	9.5	37	1	1	1	1	1
6	10, 5, 1	13.3333	156	1	3	1	9	1
16	10, 10, 1	17.25	197 040	114	1	1	1	1
8	30, 1, 1	14.5	194	1	1	1	1	1
17	30, 2, 1	17.2941	142 837	89	1	1	33	2
8	30, 5, 1	22.25	437	1	2	1	1	1
9	30, 10, 1	28.4444	1169	1	33	1	15	1
13	30, 30, 1	42.9231	17 099	9	1	1	1	1
10	50, 1, 1	19.0	351	1	1	1	1	1
21	50, 2, 1	22.6667	766 220	604	1	1	1	1
10	50, 5, 1	29.5	1397	2	1	1	1	1
10	50, 10, 1	36.5	1377	1	1	1	1	1
15	50, 30, 1	55.0	44 664	27	17	1	27	1
17	50, 50, 1	66.8235	184 068	114	1	1	1	1

The computational results for *SP* are obtained using the aforementioned branch and price approach. To compute optimal solutions for the linear programs *SPLP* and *SPD* we use the standard package ILOG CPLEX 7.5. The programs were coded in C++. In the following sections we denote by *SP*-nodes the number of nodes in the branching tree created by the algorithm described in Section 4 and we denote by *SP*-time the CPU-time in seconds rounded up.

5.2. On the column generation

Here we mention two important details concerning the implementation of the branch and price algorithm described in Section 4. Let us first comment on the choice of an initial feasible solution.

In the initialization phase of the algorithm we are free to choose any set of pairs (columns of LP) $N = \{(i, s_i): i \in M\}$. We have tested two sets of initial LP columns in our implementation. The first one contains the pairs (i, s_i) such that $s_i = \{i\}$ for any $i \neq 1$ and for $i = 1$ we have $s_1 = \{1, m+1, m+2, \ldots, T\}$. This set corresponds to the trivial feasible solution of *PMP* where we first service all the machines in order 1 up to m and then from time interval $m+1$ onwards, we service machine 1 only. We shall refer to this set of initial columns as the *simple solution*. Another set of initial columns is formed by the *greedy solution*, see Anily et al. (1998). Recall that the *greedy solution* can be obtained by the following simple rule: at each time interval t we service the machine which would have the maximal aggregated operating cost in time interval $t+1$. In our experiments we have noticed that the choice of an initial solution can have a large impact on the resulting computation time. For example, to solve the LP relaxation of *SP* in case $m = 4$, $T = 33$, $a = (10, 10, 10, 1)$, $b = (0, 0, 0, 0)$, the algorithm starting with the *simple solution* generates 312 columns and stops within 7 seconds, whereas the algorithm starting with the *greedy solution* generates 4383 columns and stops only after 488 seconds. In another instance, $m = 3$, $T = 21$, $a = (50, 2, 1)$, $b = (0, 0, 0)$, the algorithm starting with the *simple solution* generates 41 nodes in the branching tree and stops in 13 seconds, while the algorithm starting with *greedy solution* provides the integer solution in the first node of the branch-

ing tree and stops in 1 second. We conclude that the choice of an initial column set has a significant impact on the running times achieved. In the tables describing the experiments, we report the calculation results for *SP* with the best running time from the two starting solutions. To specify the initial set of columns we use the following notation: by default we use the *greedy solution* and we mark solutions provided by the algorithm starting with the *simple solution* by a superscript "*s*".

Secondly, in a column generation approach there is freedom concerning what variables with negative reduced costs (as found by the solution of the *pricing problem*) to add to the set N of columns active in the current LP. For instance, one could add all variables with negative reduced costs. For reasons of convenience we have opted in our implementation to consider two strategies. In the first strategy, we add one column at each iteration, namely the one that has the smallest reduced costs (this corresponds to the most violated constraint in the dual problem). In the second strategy, we add at most m columns per iteration: for each machine we find a column with the smallest reduced costs. In the computational results we use by default the first version of the algorithm and we mark results obtained by the second version by superscript "*m*". Again, we report the calculation results with the best running time.

5.3. Different formulations

First of all, as promised in Section 3, we illustrate the difference between formulation *QP*, flow

Table 3
Instances with four machines ($m = 4$, $b_i = 0$, $i \in M$)

T	a	OPT	*FF*-nodes	*FF*-time (second)	$v(FFLP)$	*SP*-nodes	*SP*-time (second)	$v(SPLP)$
4	1, 1, 1, 1	6.0	1	1	6.0	1	1	6.0
9	2, 1, 1, 1	7.3333	1	1	7.3333	1	1	7.3333
10	2, 2, 1, 1	8.8	1	1	8.8	1	1	8.8
15	2, 2, 2, 1	10.4	1	1	10.4	1	1	10.4
6	5, 1, 1, 1	10.0	1	1	10.0	1	1	10.0
16	5, 2, 1, 1	11.75	1	1	11.75	1	1	11.75
22	5, 2, 2, 1	13.7273	99	6	13.5	1	3	13.7273
6	5, 5, 1, 1	15.0	1	1	15.0	1	1	15.0
6	5, 5, 2, 1	17.5	1	1	17.5	1	1	17.5
24	5, 5, 5, 1	22.25	1	2	22.25	15^{s}	3^{s}	22.25^{s}
6	10, 1, 1, 1	12.5	1	1	12.5	1	1	12.5
6	10, 2, 1, 1	15.0	1	1	15.0	1	1	15.0
6	10, 2, 2, 1	17.5	1	1	17.5	1	1	17.5
8	10, 5, 1, 1	19.5	1	1	19.5	1	1	19.5
6	10, 5, 2, 1	22.5	1	1	22.5	1	1	22.5
8	10, 5, 5, 1	27.875	1	1	27.25	19	1	27.25
8	10, 10, 1, 1	24.5	1	1	24.5	1	1	24.5
6	10, 10, 2, 1	27.5	1	1	27.5	1	1	27.5
9	10, 10, 5, 1	34.0	10	1	32.8889	15	1	32.8889
33	10, 10, 10, 1	40.4545	3	9	40.4545	17^{s}	17^{s}	40.4545^{s}
8	30, 1, 1, 1	21.75	1	1	21.75	1	1	21.75
8	30, 5, 1, 1	29.5	1	1	29.5	1	1	29.5
10	30, 5, 5, 1	40.5	3	1	39.5	17	1	39.5
8	30, 10, 1, 1	37.0	1	1	37.0	1	1	37.0
12	30, 10, 5, 1	49.6667	88	2	48.0	23	1	48.0
30	30, 10, 10, 1	58.3333	123	14	57.9231	19^{sm}	19^{sm}	58.3333^{sm}
26	30, 30, 1, 1	55.8462	1	3	55.8462	1	3	55.8462
24	30, 30, 5, 1	70.5	36	5	70.4231	19^{s}	4^{s}	70.5
14	30, 30, 10, 1	81.5	20	1	80.4231	23	1	80.7857
19	30, 30, 30, 1	108.4737	1	1	108.4737	1	1	108.4737

formulation *FF* and the set-partitioning formulation *SP*. Table 2 presents computational results on the three machine instances introduced in Anily et al. (1998), where $b_i = 0$ for all $i \in M$, and we have chosen T to be the optimal schedule length as computed in Anily et al. (1998). Recall that, in order to be able to compare results with a different T, we express the optimum value (OPT) as the average operating cost per period.

We conclude from Table 2 that as the schedule length increases, the number of nodes in *QP*, as well as the computation times, increase enormously. However, each model is able to deal with the smaller sized instances ($T \leqslant 5$). We also observe that the computation times for the other two formulations are much better than *QP*. Therefore we concentrate in the remainder on the formulations *SP* and *FF* only.

5.4. The quality of the lower bound

Now, we focus on the general performance of the column generation algorithm for *SPLP* and the branch and price algorithm for *SP* versus the LP based branch and bound algorithm that the CPLEX MIP Solver uses to solve *FF*. Again, we consider instances from Anily et al. (1998) on four machines, with servicing costs $b_i = 0$, and we report the solution value in terms of the average operating cost per period. We have chosen T to be the optimal schedule length as computed by Anily et al. (1998).

The computational results depicted in Table 3 show that the lower bounds provided by the two linear programming relaxations are very good; in particular the LP relaxation of formulation *SP* misses the integral optimum 5 times and the LP relaxation of formulation *FF* misses the integral optimum 8 times (out of 30). Observe also that these values are obtained in a very short time, usually within a second.

Notice further that even in case of a positive integrality gap OPL-Studio can provide an integral solution for *FF* analyzing only one node of the searching tree. The reason for this is that the OPL-Studio MIP-solver is based on a branch and cut algorithm which creates a number of cuts (actually there are nine types of different cuts) that can lead to an integer solution right in the root-node of the searching tree (see Table 3).

Finally, we point out that for the instance with $m = 4$, $a = (30, 10, 10, 1)$, $b = (0, 0, 0, 0)$, and cycle length $T = 30$ (Table 3) we find a solution with $OPT = 58.3333$. This value is better than the solution of $OPT = 58.42$ reported in Anily et al. (1998).

5.5. Symmetry

In order to test the proposed solution approaches for large values of T, we have composed symmetrical instances where $a_i = 1$ and $b_i = 0$ for all machines $i \in M$. The structure present in these instances ensures that optimal solutions are not hard to come by, however, we are interested in the performance of the algorithms for these instances. Table 4 displays computational results for $m = 3$. For all these instances, the integrality gap of formulation *SP* equals zero; in contrast, $v(FFLP) = 3$ for all instances considered.

Table 4
Symmetric instances ($m = 3$, $a_i = 1$, $b_i = 0$, $i \in M$)

T	OPT	*FF*-nodes	*FF*-time (second)	*SP*-nodes	*SP*-time (second)
50	3.04	57	33	1^{sm}	51^{sm}
51	3.0	1	13	1^{s}	21^{s}
52	3.0385	111	42	7^{s}	154^{s}
53	3.0377	75	40	7^{s}	247^{s}
54	3.0	1	14	1^{s}	32^{s}
55	3.0364	94	50	1^{sm}	117^{sm}
56	3.0357	156	55	11^{s}	590^{s}
57	3.0	1	19	1^{s}	46^{s}
58	3.0345	69	52	1^{s}	366^{s}
59	3.0339	61	57	3^{s}	407^{s}
60	3.0	1	22	1^{s}	170^{s}
61	3.0328	104	66	3^{sm}	715^{sm}
62	3.0323	107	75	1^{sm}	1437^{sm}
63	3.0	1	28	1^{s}	195^{s}
64	3.0313	61	77	5^{s}	1431^{s}
65	3.0308	80	94	3^{sm}	1098^{sm}
66	3.0	1	31	1^{sm}	470^{sm}
67	3.0299	135	102	1^{sm}	546^{sm}
68	3.0294	162	134	1^{sm}	783^{sm}
69	3.0	1	46	1^{s}	601^{s}
70	3.0286	110	149	7^{s}	5150^{s}
80	3.025	182	258	1^{sm}	1167^{sm}
90	3.0	1	771	1^{s}	1093^{s}
100	3.02	217	1655	1^{sm}	2618^{sm}

We conclude from the results in Table 4 that these instances are not so easy to solve, especially for the branch and price algorithm. Although the integrality gap of formulation *SP* for these instances equals zero, and solving the problem in the root-node is often sufficient, many calls to the column generation procedure are needed to prove optimality. The computation times for formulation *FF* are better, despite the fact that it uses much more nodes in the search tree. Thus, we conclude that the column generation algorithm spends relatively much time on solving the LP relaxations of these instances.

We notice also that in these symmetrical instances the algorithm based on solving formulation *SP* performs better if we start with the *simple solution* as an initial set of columns in LP rather than starting with the *greedy solution*.

5.6. Maintenance costs

In this section we investigate the impact of strictly positive servicing costs b_i. To construct the instances considered in Table 5 we use a subset of the instances from Anily et al. (1998) on five machines; we set $T = 24$ for all the instances (notice that this may not correspond an optimal cycle length), and we choose the servicing costs b as indicated in Table 5. More in particular, for each choice of a we compared three choices of b, namely $b = 0$, $b = a$, and $b = (30, 10, 5, 2, 1)$.

Since for all instances of Table 5 the branch and price algorithm based on solving formulation *SP* performs worse than the implementation based on formulation *FF* we do not report here the results for *SP*.

Table 5
Instances with positive maintenance costs ($m = 5$, $T = 18$)

a	b	OPT	*FF*-nodes	*FF*-time (second)	$v(FFLP)$
5, 1, 1, 1, 1	0, 0, 0, 0, 0	15.0	1	3	15.0
5, 1, 1, 1, 1	5, 1, 1, 1, 1	17.3333	1	3	17.3333
5, 1, 1, 1, 1	30, 10, 5, 2, 1	27.0417	86	10	26.9333
5, 5, 1, 1, 1	0, 0, 0, 0, 0	21.9583	3289	20	21.75
5, 5, 1, 1, 1	5, 5, 1, 1, 1	25.4167	15858	66	25.1429
5, 5, 1, 1, 1	30, 10, 5, 2, 1	33.8333	469	9	33.7143
5, 5, 5, 1, 1	0, 0, 0, 0, 0	29.5	1	2	29.5
5, 5, 5, 1, 1	5, 5, 5, 1, 1	33.5	1	2	33.5
5, 5, 5, 1, 1	30, 10, 5, 2, 1	41.125	542	7	40.8214
5, 5, 5, 5, 1	0, 0, 0, 0, 0	40.375	59750	260	39.5
5, 5, 5, 5, 1	5, 5, 5, 5, 1	44.875	82879	357	44.10
5, 5, 5, 5, 1	30, 10, 5, 2, 1	50.375	25289	127	49.35
10, 5, 1, 1, 1	0, 0, 0, 0, 0	26.75	1	7	26.75
10, 5, 1, 1, 1	10, 5, 1, 1, 1	32.125	310	38	31.8333
10, 5, 1, 1, 1	30, 10, 5, 2, 1	41.0	5394	169	40.4167
10, 10, 5, 1, 1	0, 0, 0, 0, 0	43.5	4515	208	42.9091
10, 10, 5, 1, 1	10, 10, 5, 1, 1	50.9583	27114	829	50.2
10, 10, 5, 1, 1	30, 10, 5, 2, 1	56.125	3223	180	55.3833
30, 10, 5, 1, 1	0, 0, 0, 0, 0	61.4167	1443	71	60.8462
30, 10, 5, 1, 1	30, 10, 5, 1, 1	77.4167	912	61	76.5909
30, 10, 5, 1, 1	30, 10, 5, 2, 1	77.5	1042	67	76.6818
30, 30, 1, 1, 1	0, 0, 0, 0, 0	69.0	177	25	68.7692
30, 30, 1, 1, 1	30, 30, 1, 1, 1	91.75	61	18	91.6364
30, 30, 1, 1, 1	30, 10, 5, 2, 1	84.6667	528	54	83.7436
30, 30, 30, 1, 1	0, 0, 0, 0, 0	129.5	24292	122	126.9474
30, 30, 30, 1, 1	30, 30, 30, 1, 1	155.875	13947	74	153.7647
30, 30, 30, 1, 1	30, 10, 5, 2, 1	142.7917	24652	152	139.3860
30, 30, 30, 30, 1	0, 0, 0, 0, 0	207.75	18793	89	204.0
30, 30, 30, 30, 1	30, 30, 30, 30, 1	236.5417	23764	120	232.7826
30, 30, 30, 30, 1	30, 10, 5, 2, 1	218.2917	15191	67	214.5326

Table 6
Instances with many machines ($m = 10$, $T = 18$, $b_i = 0$, $i \in M$)

a	OPT	FF-nodes	FF-time (second)	$v(FFLP)$	SP-nodes	SP-time (second)	$v(SPLP)$
1, 1, 1, 1, 1, 1, 1, 1, 1, 1	49.0	$\gg$100 000	–	45.0	1^{s}	1^{s}	49.0^{s}
10, 9, 8, 7, 6, 5, 4, 3, 2, 1	232.0	$\gg$100 000	–	225.76	93^{sm}	29^{sm}	232.0^{sm}
10, 10, 10, 10, 10, 10, 10, 10, 10, 1	413.5	$\gg$100 000	–	393.5	3^{s}	2^{s}	413.5^{s}
100, 1, 1, 1, 1, 1, 1, 1, 1, 1	126.5	1	5	126.5	1^{m}	1^{m}	126.5^{m}
1000, 1, 1, 1, 1, 1, 1, 1, 1, 1	576.5	1	3	576.5	23^{sm}	7^{sm}	576.5^{sm}

It is hard to infer general statements from the results presented in Table 5; however, it is safe to conclude that having positive servicing costs makes the problem more difficult to solve. Only for the case with $a = (30, 10, 5, 1, 1)$ the instance with $b = 0$ is the most difficult one to solve; for all other choices of a either $b = a$ or $b = (30, 10, 5, 2, 1)$ is the more difficult one. Also, the integrality gap increases in the absence of servicing costs $b = 0$. Thus, the results in Table 5 indicate that the impact of having different servicing costs on the running time can be significant.

5.7. Cases with many machines

Finally, we investigate how the number of machines affects the performance of the algorithms. We have selected five instances with ten machines introduced in Anily et al. (1998) with servicing costs $b = 0$, and we define a relatively modest cycle length of $T = 18$. The results are described in Table 6.

We conclude from these results that the algorithm based on the formulation SP performs better than the OPL implementation based on formulation FF. We explain this as follows. First, from the experiments with these instances we find that the linear relaxation provided by formulation SP is much stronger than the linear relaxation of formulation FF. Second, when the number of machines is increased by one, the size of formulation FF is enlarged with at least $2T$ rows (constraints), while the size of formulation SP grows with just one single row. Indeed, for some instances, for example the one with $a = (10, 10, 10, 10, 10, 10, 10, 10, 10, 1)$, we could not even solve formulation FF in 12 hours. We see also that if the integrality gap is not zero then the algorithm based on formulation FF needs much more processing time and branching nodes than for instances with a small number of machines (and a non-zero integrality gap).

6. Conclusions

In this paper we have proposed several models for a periodic maintenance scheduling problem that has applications in many different areas. In contrast to previous research, our approach has been to fix the length of the period to a given constant T. We describe several natural mathematical programming formulations, most of which are integer linear programs. We have investigated the computational behavior of these formulations when solving them exactly using LP based branch and bound. One of the formulations is a set-partitioning formulation, that contains a number of variables that is exponential in the cycle length T. Among the formulations considered, this formulation has the strongest linear relaxation. We have shown how this formulation can be solved using a column generation approach, and how the corresponding pricing problem can be solved efficiently. This results in a branch and price algorithm. When comparing the computational results of this approach to the results obtained through a flow formulation, we conclude that for instances with many machines the branch and price algorithm seems more suited, whereas for instances with positive servicing costs the flow formulation dominates.

References

Ahuja, R.K., Magnanti, T.F., Orlin, J.B., 1993. Network Flows—Theory, Algorithms, and Applications. Prentice-Hall, Englewood Cliffs, NJ.

Anily, S., Glass, C.A., Hassin, R., 1998. The scheduling of maintenance service. Discrete Applied Mathematics 82, 27–42.

Anily, S., Glass, C.A., Hassin, R., 1999. Scheduling of maintenance services to three machines. Annals of Operations Research 86, 375–391.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, P.H., Vance, P.H., 1998. Branch-and-price: Column generation for solving huge integer programs. Operations Research 46, 316–329.

Bar-Noy, A., Bhatia, R., Naor, J.S., Schieber, B., 2002. Minimizing service and operation costs of periodic scheduling. Mathematics of Operations Research 27, 518–544.

Brakerski, Z., Dreizin, V., Patt-Shamir, B., 2001. Dispatching in perfectly-periodic schedules. Manuscript, Department of Electrical Engineering, Tel-Aviv University.

Brauner, N., Crama, Y., Grigoriev, A., van de Klundert, J.J., 2001. On the complexity of high multiplicty scheduling problems. Working paper GEMME 0110, University of Liège.

Charest, M., Ferland, J.A., 1993. Preventive maintenance scheduling of power generating units. Annals of Operations Research 41, 185–206.

Dekker, R., van der Duyn Schouten, F.A., Wildeman, R.E., 1997. A review of multi-component maintenance models with economic dependence. Mathematical Methods of Operations Research 45, 411–435.

Duffuaa, S.O., Ben-Daya, M., 1994. An extended model for the joint overhaul scheduling problem. International Journal of Operations and Production Management 14, 37–43.

Gertsbakh, I.B., Gertsbakh, E., 2000. Reliability Theory with Applications to Preventive Maintenance. Springer-Verlag, Heidelberg.

Grötschel, M., Lovasz, L., Schrijver, A., 1981. The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1, 169–197.

Hariga, M., 1994. A deterministic maintenance-scheduling problem for a group of non-identical machines. International Journal of Operations and Production Management 14, 27–36.

Jones, P., Zydiak, J., Hopp, W., 1991. Parallel machine replacement. Naval Research Logistics 38, 351–365.

Kenyon, C., Schabanel, N., 2001. The data broadcast problem with non-uniform transmission times. Research Report 2001-43, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon.

Kenyon, C., Schabanel, N., Young, N., 2000. Polynomial-time approximation scheme for data broadcast. In: 32nd ACM Symposium on Theory of Computing (STOC 2000), pp. 659–666.

Kralj, B.L., Petrović, R., 1988. Optimal preventive maintenance scheduling of thermal generating units in power systems—a survey of problem formulations and solution methods. European Journal of Operational Research 35, 1–15.

Kralj, B.L., Petrović, R., 1995. A multiobjective optimization approach to thermal generating units maintenance scheduling. European Journal of Operational Research 84, 481–493.

McClurg, T., Chand, S., 2002. A parallel machine replacement model. Naval Research Logistics 49, 275–287.

Schabanel, N., 2000. The databroadcast problem with preemption. In: 17th International Symposium on Theoretical Aspects of Computer Science (STACS 2000), Lecture Notes in Computer Science, vol. 1770, pp. 181–192.

Sule, D.R., Harmon, B., 1979. Determination of co-ordinated maintenance scheduling frequencies for a group of machines. AIIE Transactions 11, 48–53.

Wagner, H.M., Giglio, R.J., Glaser, R.G., 1964. Preventive maintenance scheduling by mathematical programming. Management Science 10, 316–334.