Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 899–906

ISBN 978-83-60810-14-9
ISSN 1896-7094

On a class of periodic scheduling problems: models, lower bounds and heuristics

Philippe Michelon
and Dominique Quadri
Université d'Avignon
Laboratoire d'Informatique d'Avignon
F-84911 Avignon Cedex 9, France
Email: {philippe.michelon,
dominique.quadri}@univ-avignon.fr

Marcos Negreiros
Universidade Estadual do Ceará
Departamento de Estatística e Computação
Av. Paranjana, 1700 Fortaleza, Brasil
Email: negreiro@uece.br

***Abstract*—We study in this paper a generalization of the basic strictly periodic scheduling problem where two positive integer constants, associated to each task, are introduced such that to replace the usual strict period. This problem is motivated by the combat of the dengue which is one of the major tropical disease. We discuss general properties and propose two integer mathematical models of the problem considered which are compared theoretically. We also suggest a lower bound which is derived from the structure of the problem. It appears to be quickly obtained and of good quality. Three greedy algorithms are proposed to provide feasible solutions which are compared with the optimum (when it can be obtained by the use of ILOG-Cplex10.0). It is shown that for special instances greedy algorithms are optimal.**

I. Introduction

DENGUE is a flu-like viral disease spread by the bite of infected mosquitoes and occurs in most tropical areas of the world. One of a severe complication of this disease is dengue hemorrhagic fever which is often fatal. Unfortunately, there is no specific treatment for it. Consequently at the present time the only way of preventing or of fighting the dengue is to eliminate the vector mosquitoes which are located in breeding sites. More specifically equipped vehicles are sent in each infested and detected areas to pulverize insecticide. In practice those sites are divided in sub-areas which are computed so as to exactly obtain one working day for each vehicle (formally each task takes a duration of unite time). Unfortunately, the pulverized product only kills mosquitoes but leaves the larves in life. Indeed, it takes between 7 and 9 days for a larve to become an adult mosquito. Therefore, each sub-area has to be treated repetitively with a delay of at least 7 and at most 9 days between two pulverizations to achieve the best efficiency. We refer to [22] for more details on the logistic aspect of prevention and combat of the dengue. More formally, we consider the problem of minimizing the number of vehicles required to make periodic single destination equipped vehicle to a set of infested sub-areas. We therefore face to a periodic scheduling problem (which is in this basic form NP-hard [10]) where the periodicity is not strict but represented by both a minimum and a maximum delay (respectively, in this special case 7 and 9). We show in this paper that if all the maximum periods are equals for all tasks then the problem relative to our application of dengue prevention can be solved in polynomial time. We will then also study the more general case where all the maximum periods are different.

We therefore extend our work to a generalization of a strict periodic scheduling problem which is basically concerned with processing, on a set of identical machines (or identical unitary resources), periodic tasks or activities over an infinite horizon. Each activity i is characterized by a duration d_i. We are actually concerned with the following generalization of the strict periodicity requirement: two positive integer numbers $\underline{F_i}$ and $\overline{F_i}$ are associated with task i and correspond, respectively, to a minimum and a maximum delay for the repetition of activity i. Thus, if the k^{th} execution of i has been scheduled on time $t_{i,k}$ then $t_{i,k+1}$ must belong to $[t_{i,k} + d_i + \underline{F_i}, t_{i,k} + d_i + \overline{F_i}]$. We also consider a finite time horizon, all the task durations equals to 1 (i.e. $d_i = 1 \ \ \forall i$) and unary resources. Our aim is to minimize the number of resources (or machines or vehicles) so as to execute all the activities.

In this paper, we examine the structure of this general problem (denoted by $GSPS$), giving some properties. We propose two integer linear formulations which we name "weak and strong" formulations for $GSPS$. We compare theoretically those formulations that asserts the named of each model. From the strong formulation we derived a lower bound denoted by $\lceil Z[\overline{GSPS}] \rceil$. We then suggest a second lower bound of $GSPS$ easier to be computed and of good quality which appears to be optimal for special cases, including when all the maximum delay are equals. Finally, we present three greedy algorithms which provide feasible solutions. Those upper bounds are evaluated and compared with both the optimal value of the integer linear model when it can be obtained in a competitive CPU time given by CPLEX10.0. and the lower bounds we proposed.

The paper is organized as follows. In Section II, we formally define the problem $GSPS$ and give necessary notations. Section III describes the relevant literature. We establish in Section IV some properties relative to the considered scheduling problem and provide a trivial lower bound. Section V is dedicated to the formulation of $GSPS$ by two integer linear programs.

In Section VI we propose three greedy algorithms so as to obtain good feasible solutions. The computational results are reported in Section VII. In Section VIII we summarize the main results of this paper and we point out some directions for future research.

II. Problem definition

We consider in this paper a problem denoted by $GSPS$ which is a generalization of both the basic strictly periodic scheduling problem [10] and the problem derived from the combat of the dengue described in the introduction.
Formally, it can be stated as follows. We consider J types of activities and associate at each type j ($j = 1, .., J$) the following parameters:

- n_j, the number of activities of type j.
- $\underline{F}_j$ the minimum delay between two executions of an activity of type j.
- $\overline{F}_j$ the maximum delay between two executions of an activity of type j.

Each activity of each type requires an unary resource (at each time it is processed) and has a duration of a unit time. The resources are identical and renewable (i.e. right after having processed a task, the resource is available for another task).

The objective of the problem is to find a feasible schedule, with respect to the minimum and maximum delays between two executions of the same activity, over a time horizon H while minimizing the number of used resources. The activities of type j are required to be executed at least once in the first $\overline{F}_j$ units of time.

The following proposition establishes the complexity of the problem we study.

Proposition 1: The problem $GSPS$ considered here is NP-Hard.

Proof: If $\underline{F}_j = \overline{F}_j,\ \forall j = 1...J$ then $GSPS$ corresponds to the strictly periodic problem which has been shown as being NP-Hard (see [10] and [24]). Consequently, the strictly periodic scheduling problem is a special case of $GSPS$. Since it is NP-hard then $GSPS$ remains NP-hard. ■

To get a clearer reading, let us provide an example of a feasible solution (which actually happens to be optimal) represented by Figure 1. This solution uses 4 units of resource or machines (or vehicles if we refer the dengue application) over an horizon of 20 units of time, for the problem corresponding to the following data:

type	n	$\underline{F}$	$\overline{F}$	activities
1	3	0	2	1,2,3
2	2	2	3	4,5
3	1	2	4	6
4	2	3	4	7,8
5	1	6	6	9

where the first column corresponds to the type identifier, the second column reports the number of activities for the relative type, the third and fourth columns give respectively the minimum and maximum delay between 2 executions of an activity of this type and the last ones indicates which are the activities of this type.

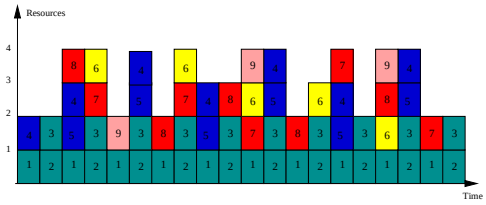

Fig. 1. A feasible solution

III. Literature review

We review in this section the literature relative to periodic scheduling problems. More specifically, we begin by presented the state of art concerning the strictly periodic scheduling (and/or routing) problem. We note that the major attention with regard to this problem has been addressed to heuristics methods. We then pursue the literature concerning variant and generalization of the previous problem. We finally briefly describe other applications of periodic scheduling problems.

In a basic version of strictly periodic scheduling problem, all durations are equals to 1 and a period T_i is also associated with each of the task so that if task i ($\forall i = 1, ..., n$) is scheduled on time t then it must also be strictly scheduled on times $t+d_i+T_i$, $t+2d_i+2T_i$ and so on. The problem consists then in minimizing the number of machines to process periodically the tasks. This basic problem has been shown as being NP-hard ([10], [24]) although it has also been shown that it is sufficient to compute a schedule over a time horizon equal to the lowest common multiplier of the T_i (since the latest can be duplicated for larger time horizons [10]).

Generalizations of this basic periodic problem have then been introduced. Jan Korst's Ph.D. thesis [19] considers a periodic scheduling problem very similar but with general integral execution times. In Park and Yun [23] and Gaudioso et. al. [11], unitary duration times are considered with task i requires w_i units of resources, while being executed. Both problems are identical and do correspond with the basic problem when all d_i and w_i values equal one. A large integer program was then introduced by [23] and [12] to solve the resulting load minimization problem, however Park and Yun [23] also describe a method for decomposing the initial problem into smaller subproblems. They divide the set of activities into sets $N_1, N_2, \cdots, N_d$, where for any $i \in N_q$ and any $j \in N_r$, p_i and p_j (periods) are relatively prime. They then use integer programming to solve each resulting subproblem, minimizing the number of resources required for each subset of customer. If K_q represents the minimum number of resources required for customer set N_q, Park and Yun [23] showed that the minimum number of vehicles required to service the customers is exactly $K_1+K_2+\cdots+K_d$ since the decomposition does not increase the number of vehicles required. Finally, Gaudioso et al. [12] also present a branching heuristic with several possible branching rules, where the greedy algorithm presented

in [10] corresponds to one of the choice of these rules. Another generalization has been considered in [5], also in the strictly periodic case, by considering non unitary resources and non unary demands associated with every activities have a duration of 1 time unit. The concept of tree scheduling is then introduced, which is a methodology for developing perfectly periodic schedules based on hierarchical round-robin, where the hierarchy is represented by trees. Some optimal (exponential time) and efficient heuristic algorithms which generate schedule trees are presented.

Some authors relax the strict periodicity requirement ([14]): rather than exactly scheduling activity i on times $t + d_i + T_i$, $t + 2d_i + 2T_i$, ..., it is allowed to be scheduled on time slots centered in $t + d_i + T_i$, $t + 2d_i + 2T_i$, ... Namely, the second execution of task i must occur in $[t+d_i+T_i-a_i, t+d_i+T_i+b_i]$, the third in $[t + 2d_i + 2T_i - a_i, t + 2d_i + 2T_i + b_i]$, etc. where a_i and b_i are non negative real numbers associated with i. Baruah et al. [8] introduce the notion of Proportionate fairness (PFair) Scheduling. PFair scheduling differs from more conventional real-time scheduling approaches in that tasks are explicitly required to execute at steady rates and have been deeply studied [9], [7] and [3]. This (multiple-resource) periodic scheduling problem was first addressed by Liu[21]. Baruah et al. [8] showed that PFair scheduling can be solved in polynomial time.

Other closer works are found also in the periodic routing problems (PRP). In [10], the authors consider the problem of minimizing the number of vehicles required to make strictly periodic, single destination deliveries to a set of customers, under the initial assumption that each delivery requires the use of a vehicle for a full day. A greedy algorithm that is optimal in some special cases is proposed. A variant of this problem is also considered when the restriction of a full day for each delivery is relaxed. The same relaxation proposed in this article is also considered in [11], however, the authors address conjointly the routing problem for each day, which makes the problem much more difficult to solve due to the routing component and its interactions with the periodic scheduling problem.

Early work on periodic schedules was also motivated by Teletext Systems [2], [13] and [1], maintenance problem [4], [25] and [20] and broadcast disks [6], [18], [16] and [17]. This latter issue gained a lot of attention recently since they are used to model backbone communication in wireless systems, Teletext systems and efficient web caching in satellite systems. Finally in [15], the authors address how to sequence the movements of robots in order to minimize the number of robots required to complete a series of tasks over a fixed time horizon. In this case, the periodicity comes from the sequence of each robot's movement rather than from the jobs being processed.

IV. Some properties and a trivial lower bound

We dedicated this section to the study of the structure of the problem $GSPS$. We thus first exhibit some observations and properties. We then provide a simple way to compute a lower bound of good quality for $GSPS$ derived from the structure of the problem which becomes to be optimal when all the maximum periods are equals.

Observation 1: As proved in [10] if the problem is a strictly periodic scheduling then to obtain a solution over an infinite time horizon, it is necessary only to find a solution over a time horizon equal to the lowest common multiplier of the periods. On the opposite, if we consider our problem $GSPS$, it is not feasible to duplicate over an infinite horizon a schedule computed over a horizon equal to the lowest common multiplier of the maximum period. Indeed, since the period is not strict any more then the previous property can not be established.

Observation 2: The decomposition procedure of the basic periodic scheduling problem, into sub-problems easier to be solved, suggested by Park and Yun [23] can not be extended in our context because the periodicity is not any more strict.

The two previous observations are straightforwardly validated by counter-examples, which are easy to be found.

In spite of the fact that the more general periodicity seems to be inconvenient for using the lowest common multiplier of the periods to establish properties, this notion still plays a key role in the computation of lower and upper bounds for our problem $GSPS$. Indeed, we present now a simple technique to compute a lower bound of $GSPS$ which becomes easier if the horizon time H is greater than the common multiplier of the maximum delays $\overline{F_j},\ \forall j = 1...J$. The upper bounds will be provided by greedy algorithms in Section VI.

A simple way to under-evaluate the number of needed resource units is to find how much time each activity has to be processed. For this purpose, let us consider a time t ($\leq H$). Between times 1 and t, an activity of type j will be processed at least $\lfloor \frac{t}{\overline{F_j}} \rfloor$ and therefore, between times 1 and t, we have to process at least $\sum_{j=1}^{J} n_j \lfloor \frac{t}{\overline{F_j}} \rfloor$ tasks, hence an average of:

$$\frac{\sum_{j=1}^{J} n_j \lfloor \frac{t}{\overline{F_j}} \rfloor}{t} \tag{1}$$

per time unit. Since the average of a set of values is smaller than the maximum, the smallest integer greater than quantity (1) (that is the ceil of (1)) is a lower bound to the number of resources needed to schedule all the activities between times 1 and t. In order to get the best (over the set of those lower bounds) lower bound for scheduling all the activities of all the types over the time horizon, we then have to compute;

$$TLB = \max \left\{ \left\lceil \frac{\sum_{j=1}^{J} n_j \lfloor \frac{t}{\overline{F_j}} \rfloor}{t} \right\rceil / \quad t = 1, ..., H \right\} \tag{2}$$

The above quantity will be referred as the "Trivial Lower Bound" (TLB).

The following proposition establishes the role of the lowest common multiplier of the $\overline{F}_j$ ($j = 1, ..., J$) in the computation of the (TLB).

Proposition 2: If the time horizon H is greater or equal than the lowest common multiplier of the $\overline{F}_j$ ($j = 1, ..., J$), then the (TLB) is actually equal to:

$$TLB = \left\lceil \sum_{j=1}^{J} \frac{n_j}{\overline{F}_j} \right\rceil \tag{3}$$

Proof: Note first that $TLB \leq \left\lceil \sum_{j=1}^{J} \frac{n_j}{\overline{F}_j} \right\rceil$: since $\lfloor \frac{t}{\overline{F}_j} \rfloor \leq \frac{t}{\overline{F}_j}$, we obviously have:

$$\frac{\sum_{j=1}^{J} n_j \lfloor \frac{t}{\overline{F}_j} \rfloor}{t} \leq \frac{\sum_{j=1}^{J} n_j \frac{t}{\overline{F}_j}}{t} = \sum_{j=1}^{J} \frac{n_j}{\overline{F}_j} \tag{4}$$

It then follows that:

$$TLB = \max \left\{ \left| \frac{\sum_{j=1}^{J} n_j \lfloor \frac{t}{\overline{F}_j} \rfloor}{t} \right| / \quad t = 1, ..., H \right\} \leq \left\lceil \sum_{j=1}^{J} \frac{n_j}{\overline{F}_j} \right\rceil \tag{5}$$

Let now F denote the lowest common multiplier of the $\overline{F}_j$ ($j = 1, ..., J$). If $H \geq F$, we can consider inequality (4) for $t = F$. Since, $\frac{F}{\overline{F}_j}$ is integer, the inequality $\lfloor \frac{t}{\overline{F}_j} \rfloor \leq \frac{t}{\overline{F}_j}$ becomes an equality and, thus,

$$TLB = \max \left\{ \left| \frac{\sum_{j=1}^{J} n_j \lfloor \frac{t}{\overline{F}_j} \rfloor}{t} \right| / \quad t = 1, ..., H \right\} \geq \left| \frac{\sum_{j=1}^{J} n_j \frac{F}{\overline{F}_j}}{F} \right| \tag{6}$$

Since,

$$\left| \frac{\sum_{j=1}^{J} n_j \frac{F}{\overline{F}_j}}{F} \right| = \left\lceil \sum_{j=1}^{J} \frac{n_j}{\overline{F}_j} \right\rceil \tag{7}$$

we have,

$$TLB = \max \left\{ \left| \frac{\sum_{j=1}^{J} n_j \lfloor \frac{t}{\overline{F}_j} \rfloor}{t} \right| / \quad t = 1, ..., H \right\} = \left\lceil \sum_{j=1}^{J} \frac{n_j}{\overline{F}_j} \right\rceil \tag{8}$$

■

We show now, as mentioned in the introduction, that the application of the combat of the dengue, which represents a particular case of $GSPS$ can be solved in polynomial time.

Proposition 3: If all the types have the same maximum delay $\overline{F}$ (i.e. $\overline{F}_j = \overline{F} \ \ \forall j = 1, ..., J$) then the problem $GSPS$ can be solved to optimality in $\lfloor \frac{H}{\overline{F}} \rfloor \sum_{j=1}^{J} n_j$ operations, that is in polynomial time with respect to the total number of tasks to be scheduled. In addition, the optimal value is exactly equal to the value provided by the Trivial Lower Bound.

Proof: With no loss of generality, we can assume that $H \geq \overline{F}$ (otherwise, it is not necessary to execute the activities). Let us consider a particular schedule that we are going to show as being feasible and which provides a objective function value equal to (TLB). As a consequence, this schedule will be shown as being optimal. This particular schedule is builded by the following greedy method:

- If $\sum_{j=1}^{J} n_j \leq \overline{F}$ then the problem is rather easy to solve : assign activity 1 on times 1, $1 + \overline{F}$, $1 + 2\overline{F}$ and so on; activity 2 on times 2, $2 + \overline{F}$, $2 + 2\overline{F}$,... etc ...
  It is then direct to observe that this procedure will use only one unit resource and that each activity will be executed with a strict period of $\overline{F}$ units of time. Thus, the produced schedule is feasible and optimal since no less than 1 resource unit can be used.
- Otherwise, i.e. if $\sum_{j=1}^{J} n_j > \overline{F}$, then we take the first $\overline{F}$ activities and assign activity 1 on times 1, $1+\overline{F}$, $1+2\overline{F}$ and so on, activity 2 on instants 2, $2 + \overline{F}$, $2 + 2\overline{F}$,... etc ...
  Consequently, each of those activities are executed according to a strict period of $\overline{F}$ units of time and on one unit of resource.
  Repeat the same process for the next $\overline{F}$ activities and so on until that the number of remaining activities is no greater than $\overline{F}$. Then apply the same procedure as above. Once again, each of the activities are executed exactly $\overline{F}$ units of time (hence, the schedule is feasible) and after an easy calculation it can be straightforwardly established that this schedule requires $\left\lceil \frac{\sum_{j=1}^{J} n_j}{\overline{F}} \right\rceil$ units of resource. Since this quantity is exactly equal to the Trivial Lower Bound, the schedule is actually optimal.

■

V. INTEGER LINEAR MODELS

In this Section, we first introduce two integer mathematical programs modelling $GSPS$ for the general periodic scheduling problem. Then, a theoretically comparison of the models is presented *via* corollary 1. Consequently, we will refer to a "weak formulation" and to a "strong formulation". We begin by establishing the so-called "weak formulation".

First are introduced some necessaries notations. Let $J(i) \in \{1, 2, ..., J\}$ be the type of activity i and $n = \sum_{j=1}^{J} n_j$ the total number of activities to process.

Let us now define the following decision variables:

$$x_{it} = \begin{cases} 1 & \text{if task } i \text{ is processed on time } t \\ 0 & \text{otherwise} \end{cases}$$

where $i = 1, ..., n$ and $t = 1, ..., H$.

R = the number of unit resources that we want to minimize

Let also $J(i) \in \{1, 2, ..., J\}$ be the type of activity i and $n = \sum_{j=1}^{J} n_j$ the total number of activities to process. Then, the "weak formulation" can be stated as follows:

$$GSPS \begin{cases} \min\ R \\ s.t. \left| \begin{array}{l} (9), (10)\ or\ (12), (11) \\ x_{it} \in \{0, 1\},\ \forall i = 1, ..., n,\ \forall t = 1, ..., H \\ R \geq 0 \end{array} \right. \end{cases}$$

where contraints (9), (10), (11) and (12) are respectively defined as follows.

At each time t, the number of used resources is greater or equal than the number of scheduled activities. Consequently we have constraint (9):

$$R \geq \sum_{i=1}^{n} x_{it} \quad \forall t = 1, ..., H \tag{9}$$

If activity i is scheduled on instant t, then it must also be scheduled between instants $t + 1 + \underline{F}_{j(i)}$ and $\overline{F}_{j(i)}$. It follows constraint (10):

$$\sum_{l=t+1+\underline{F}_{j(i)}}^{t+\overline{F}_{j(i)}} x_{il} \geq x_{it} \qquad \forall i = 1, ..., n \quad \forall t = 1, ..., H - \overline{F}_{j(i)} \tag{10}$$

Finally, each activity can be scheduled at most once on each period of $\underline{F}_{j(i)}$ consecutive days. We are then face to constraint (11):

$$\sum_{l=t}^{t+\underline{F}_{j(i)}} x_{il} \leq 1 \qquad \forall i = 1, ..., n \quad \forall t = 1, ..., H - \underline{F}_{j(i)} \tag{11}$$

The "strong formulation" we proposed is obtained by substituting constraint (10) by:

$$\sum_{l=t+}^{t+\overline{F}_{j(i)}} x_{il} \geq 1 \qquad \forall i = 1, ..., n \quad \forall t = 1, ..., H - \overline{F}_{j(i)} \tag{12}$$

Actually, this constraint states that, on each period of $\overline{F}_{j(i)}$ units of time, the activity must be executed at least once.

In the remainder of this paper, we will denoted by $\lceil Z[\overline{GSPS}] \rceil$ the smaller integer greater than or equal to the value optimal value of the LP-relaxation of $GSPS$ ("strong formulation").

The following proposition addresses the lower bound provided by the LP-relaxation of the "strong formulation".

Proposition 4: If the time horizon H is greater or equal than the lowest common multiplier of the $\overline{F}_j$ ($j = 1, ..., J$), then $\overline{x}$ defined by $\overline{x_{it}} = \dfrac{1}{\overline{F}_{j(i)}} \quad \forall i = 1, ..., n,\ t = 1, ..., H$, is an optimal solution of the LP-relaxation of the "strong formulation" and, therefore, its optimal value is $\overline{R} = \sum_{j=1}^{J} \dfrac{n_j}{\overline{F}_j}$.

Proof: Let us show that $(\overline{R}, \overline{x})$ is feasible for the linear relaxation of the initial problem.

- $\sum_{i=1}^{n} \overline{x}_{it} = \sum_{j=1}^{J} \dfrac{n_j}{\overline{F}_j}$ and, hence, constraint (9) is satisfied by $(\overline{R}, \overline{x})$.
- $\sum_{l=t}^{t+\overline{F}_{j(i)}} \overline{x}_{il} = \dfrac{\overline{F}_{j(i)}}{\overline{F}_{j(i)}} = 1$ and constraint (12) is thus verified.
- $\sum_{l=t}^{t+\underline{F}_{j(i)}} \overline{x}_{il} = \dfrac{\underline{F}_{j(i)}}{\overline{F}_{j(i)}} \leq 1$ and constraint (11) is also verified.

Therefore, $(\overline{R}, \overline{x})$ is feasible for the linear relaxation. Let us now show that it is optimal. For this purpose, consider, for any (continuous) feasible solution (R, x) of the linear relaxation the following quantity and any activity i : $\sum_{t=1}^{F} x_{it}$ where F is the lowest common multiplier of the $\overline{F}_j$. Then, from constraint 12, we have:

$$\sum_{t=1}^{F} x_{it} = x_{i1} + \ldots x_{i\overline{F}_{j(i)}} + x_{i\overline{F}_{j(i)}+1} \ldots + x_{iF} \geq \frac{F}{\overline{F}_{j(i)}} \tag{13}$$

since (12).

On the other hand, constraints (9) holds for any t from 1 to F. Thus :

$$F \times R \geq \sum_{t=1}^{F} \sum_{i=1}^{n} x_{it} \tag{14}$$

and therefore (from(13):

$$F \times R \geq \sum_{j=1}^{J} F \frac{n_j}{\overline{F}_j} \tag{15}$$

that is

$$R \geq \sum_{j=1}^{J} \frac{n_j}{\overline{F}_j} = \overline{R} \tag{16}$$

It follows that the value provided by the objective function using any feasible solution is at least the value given by $\overline{x}$. As a consequence, $(\overline{R}, \overline{x})$ is optimal for the LP-relaxation of the "strong formulation" whenever the horizon is greater or equal to F. ■

Corollary 1: If the time horizon H is greater or equal than the lowest common multiplier of the $\overline{F}_j$, then the two formulations are well named, that is the lower bound provided by the "weak formulation" is smaller or equal to the bound provided by the linear relaxation of the "strong formulation".

Proof: It is sufficient to establish that $(\overline{R}, \overline{x})$ is feasible for the "weak formulation", or in other words, that $\overline{x}$ satisfies constraint (10), using a simple calculation. ■

Corollary 2: If the time horizon H is greater or equal than the lowest common multiplier of the $\overline{F}_j$ ($j = 1, ..., J$), then the linear relaxation bound of the "strong formulation" is equal to the trivial lower bound.

Proof: Straightforward from propositions 2 and 4. ■

VI. The proposed heuristics

We propose in this Section, three greedy algorithms so as to obtain good feasible solutions for $GSPS$. The main idea of the three heuristics is based on both the trivial lower bound and the possibility to schedule a task as late as it can be done. We successively give the main idea of the three greedy algorithms.

Heuristics I and II. First of all, Heuristics I and II only differ by a sorting criteria of the maximum periods. For both heuristics, the idea is to schedule first the most difficult types. The Heuristic I and II correspond to two different measures of what is a "difficult type". In Heuristic I, a type can be considered difficult if it is supposed to induce a large consumption of resources, which can be measured, from Proposition 2, by $\frac{n_j}{\overline{F_j}}$. Consequently, in Heuristic I, the types are sorted by decreasing order of the $\frac{n_j}{\overline{F_j}}$ whereas in Heuristic II, a type is "difficult" if we have only few possibilities to schedule a task after having chosen its first execution, that is if $\overline{F_j} - \underline{F_j}$ is "small". Thus, in Heuristic II, the types are sorted by increasing order of $\overline{F_j} - \underline{F_j}$. An outline of those two heuristics is given by Algorithm I and II.

Algorithm 1 Heuristic I

```
Sort the types according to decreasing n_j / F̄_j
for j = 1 to J do
  Compute TLB for the first j types
  if it is possible then
    Schedule successively the activities of type j as late as
    possible within the open resources
  else
    open a new resource
  end if
end for
```

Heuristic III. We keep for this heuristic the sorting criteria relative to Heuristic II. Based on this we first schedule the tasks concerning the "easier" type and solve exactly the corresponding sub-problem on the whole horizon, using a branch-and-bound algorithm provided by Cplex10.0. We repeat this previous approach for all the types. The main steps of Heuristic III are reported in Algorithm 3.

Algorithm 2 Heuristic II

```
Sort the types according to increasing F̄_j − F_j
for j = 1 to J do
  Compute TLB for the first j types
  if it is possible then
    Schedule successively the activities of type j as late as
    possible within the open resources
  else
    open a new resource
  end if
end for
```

Algorithm 3 Heuristic III

```
Sort the types according to increasing F̄_j − F_j
for j = 1 to J do
  Successively schedule on the whole horizon each of the
  activities of the current type by minimizing R taking into
  account the activities already scheduled.
end for
```

VII. Computational results

The objective of the computational study we conducted in this paper is to determine and/or assert:

- if the "strong formulation" allows a solver like ILOG-Cplex10.0 to solve to optimality large scale instances;
- if the lower bound ($\lceil Z[\overline{GSPS}] \rceil$) provided by the use of the "strong formulation" is of good quality and obtained quickly;
- if our (TLB) provides a good lower bound for $GSPS$ for both if H is greater than the lower common multiplier of the maximum delays or not;
- the quality of the three upper bounds obtained utilizing the three algorithms we suggest, comparing those performance between each other and between the lower bounds;
- the approaches (I), (II) and (III) we proposed behave well and in a very fast CPU time.

Since no benchmark for $GSPS$ is available nowadays, we consider different two major types of randomly generated instances endowing each a particular structure:

1) ($H < lcm$): H is lower than the lower common multiplier (lcm) of the maximum delays;
2) ($H \geq lcm$): H is greater than or equal to the lower common multiplier (lcm) of the maximum delays. Actually, we set $H = lcm + 1$ (otherwise (lcm) takes very large values).

The rationale for utilizing instances (1) and (2) stems from the theoretical study of $GSPS$ which shows that the lower common multiplier (lcm) of the maximum period plays a key role in the evaluation of lower bound and in the resolution of $GSPS$.

In addition with regard to (1) and (2) problems, parameters $\overline{F_j}$, $\underline{F_j}$ and n_j were respectively uniformly drawn at random in the range $\{1, ..., 20\}$, $\{1, ...\overline{F_j}\}$ and $\{1, ..., 20\}$ (cf. (1)) and

such that $\{1, ..., 10\}$, $\{1, ...\overline{F_j}\}$ $\{1, ..., 10\}$ (cf. (2)). We reduce to 10 the integer value possibility concerning instances (2) because of the largest value of the corresponding (lcm) which implies a very large value for H. Indeed, H is a parameter which depends on the value of (lcm) which is computed after generation of the maximum delays. We also generated two sub-types of instances as follows:

(1.a) ($H < lcm$) and all the parameters $\overline{F_j}$ and $\underline{F_j}$ are randomly generated in the range defined previously;
(1.b) ($H < lcm$) and a half of the parameters $\overline{F_j}$ and $\underline{F_j}$ are equals and otherwise generated basically;
(2.a) ($H \geq lcm$) and all the parameters $\overline{F_j}$ and $\underline{F_j}$ are randomly generated in the range defined previously;
(2.b) ($H \geq lcm$) and a half of the parameters $\overline{F_j}$ and $\underline{F_j}$ are equals and otherwise generated basically.

The coefficient J (i.e the number of types), is given. We chose to get $J = 5$ and $J = 10$. Consequently, the total number of the tasks equal to $\sum_{j=1}^{J} n_j$ is obtained by a simple calculation. On average concerning instances (1), (2) the number of total tasks is respectively equal to 55, 30 (cf. $J = 5$) and 106, 120 (cf. $J = 10$). We also play on the gap between the values of $\overline{F_j}$ and $\underline{F_j}$ so as to create more difficult instances. Indeed, smaller the gap is so more difficult is $GSPS$ to be solved to optimality in a fast CPU time (see instances (1.b et 2.b)). For example, the relative gap between the maximum and minimum delays is on average equal to 4 for (1.a) and (2.a) where as it is on average equal to 2 with regard to instances (1.b) and (2.b) when $\overline{F_j} \neq \underline{F_j}$ otherwise the gap is equal to 0. In addition, H is approximatively equal to 220 for the simulations (1) where as its value is on average 400 for problems (2). Finally, the size of problem $GSPS$ depends on the quantity of the tasks by type and on the size of the horizon time. Consequently the number of variables and contraints of $GSPS$ can be easy computed to give an idea of the difficulty of problems treated. For example, concerning instances (1.a) and (1.b) the problem size is of 12100 and 23320 variables whereas with regard to instances (2.a) and (2.b) the number of variables of $GSPS$ is equal to 12000 and up to 48000 variables. Thereby, we consider here very large scale problems which suggest that they would be difficult to be solved exactly using ILOG-Cplex10.0, especially for instances (2).

To assess the quality of the two lower bounds (LB) (namely, (TLB) and $\lceil Z[\overline{GSPS}] \rceil$) and the three upper bounds (UB) (cf. instances (1)) we used the optimum (when it can be found) to compute the relative gap (Gap = (UB - optimum)/(UB) or (optimum - LB)/(optimum)). Nevertheless, as the size of the problem $GSPS$ dramatically increases when $H \geq lcm$, the optimum is almost never reached by the use of the branch-and-bound provided by ILOG-Cplex10.0. Consequently, in the context of instances (2) we compared the lower bounds with the value of the upper bound we proposed i.e. Gap = (UB - LB)/(UB).

Our lower (TLB) and upper bounds were coded in C++ langage. The lower bound $\lceil Z[\overline{GSPS}] \rceil$ relative to the LP-relaxation of $GSPS$ as well as the optimum value of $GSPS$ were obtained using the commercial solver ILOG-Cplex10.0. Simulations were run on a bi-weon 3.4 GHz with 4 GB of main memory.

Table I displays the average deviation of each bound to the optimum over ten replications of each types of instances (1). It appears that our (TLB) as well as $\lceil Z[\overline{GSPS}] \rceil$ on average always reach the optimal value. It could be surprising because $H < lcm$ for these instances. Nevertheless, those two lower bounds behave very well even if when the gap between the maximum and minimum delays is small or equal to 0 (cf. (1.b)). Concerning the feasible solutions, the lowest gap is obtained by Heuristic III which suggests that both sorting the type according to the lowest gap between the maximum and minimum periods and solving exactly each sub-problem corresponding to this selected type provide a feasible solution of good quality. However, Heuristic II (which utilizes the same previous sorting criteria) behaves less well than Heuristic I. Consequently, the only use of criteria relative to Heuristic II would not be sufficient to imply an upper bound closed to the optimum. Finally, for this type of problems (1), the "strong formulation" allows the branch-and-bound of ILOG-Cplex10.0 to solve to optimality most of the instances. Indeed, over 40 replications, only two were not solved in a time limit equal to 10800 seconds.

TABLE I
COMPARISON OF THE QUALITY OF THE UPPER AND LOWER BOUNDS WHEN OPTIMUM IS OBTAINED: INSTANCES (1)

Inst.	# types	(TLB)	H.I	H.II	H.III	$\lceil Z[\overline{GSPS}] \rceil$
		Gap (%)				
(1.a)	5	0.0	9.49	22.86	5.05	0.0
(1.b)	5	0.0	10.2	25.1	7.3	0.0
(1.a)	10	0.0	12.68	28.69	5.85	0.0
(1.b)	10	0.0	13.2	29.62	6.02	0.0

Table II is concerned with the average deviation of each lower bound to the upper bounds over ten replications of each types of instances (2). Since, H is greater than the lower common multiplier of the $\overline{F_j}$, its value is very large which implies that the problem size of $GSPS$ model also corresponds to very large instances. Indeed, $GSPS$ is an integer linear program which is well known to be difficult to solve in practice. Moreover, even if Heuristic III still provides the lowest gap, we note that Heuristic I behaves also well. In addition, concering the lower bounds, (TLB) and $\lceil Z[\overline{GSPS}] \rceil$ provide the same lower bound of quite good quality which asserts the theoretically result presented in Corollary 2.

Table III displays the CPU time in seconds required to compute the two lower bounds, the three upper boundsand the optimum (using ILOG-Cplex10.0) concerning instances (1) and (2). The advantage of using greedy heuristics strikingly appears: the branch-and-bound algorithm requires on average at most 291 seconds to reach the optimum value. In any case, (TLB) and the two feasible solutions given by Heuristic I and II require less than one second to provide a value. The most time consuming bound is the LP-relaxation with a maximum

TABLE II
COMPARISON OF THE QUALITY OF THE UPPER AND LOWER BOUNDS WHEN OPTIMUM IS NOT REACHED: INSTANCES (2)

Inst.	# types	(TLB)			$\lceil Z[\overline{GSPS}] \rceil$		
		H.I	H.II	H.III	H.I	H.II	H.III
(2.a)	5	8.93	10.17	6.16	8.93	10.17	6.16
(2.b)	5	9.07	11.18	7.02	9.07	11.18	7.02
(2.a)	10	21.65	40.75	3.53	21.65	40.75	3.53
(2.b)	10	22.01	41.03	4.08	22.01	41.03	4.08

of about 987 seconds to solve one of the harder instances (2.b). The time to compute the feasible solution provided by Heuristic III deviates at most of 36.67 seconds for the largest problems (2.b). As a result, even if the quality of the upper bound obtained by Heuristic III is better than the two others, it takes a running time almost important which will not be advised to be used in a branch-and-bound algorithm. Finally, $\lceil Z[\overline{GSPS}] \rceil$ is very time consuming (up to 987 seconds). Since in this particular instances, (TLB) and $\lceil Z[\overline{GSPS}] \rceil$ provides the same lower bound, it would be an advantage to utilize (TLB) instead of $\lceil Z[\overline{GSPS}] \rceil$ because of its faster running time.

TABLE III
COMPARISON OF THE CPU TIMES EXECUTION OF UPPER AND LOWER BOUNDS METHODS AND BRANCH-AND-BOUND ALGORITHMS: INSTANCES (1) AND (2)

Inst.	# types	(TLB)	H.I	H.II	H.III	$\lceil Z[\overline{GSPS}] \rceil$	Opt.
		CPU times (s)					
(1.a)	5	0.0	0.0	0.0	3.95	152.95	291.77
(1.b)	5	0.0	0.0	0.0	4.02	194.51	321.69
(1.a)	10	0.0	0.0	0.0	4.62	153.08	852.23
(1.b)	10	0.0	0.0	0.0	5.07	201.32	978.36
(2.a)	5	0.0	0.0	0.0	19.08	263.11	-
(2.b)	5	0.0	0.0	0.0	21.23	278.34	-
(2.a)	10	0.0	0.0	0.0	33.02	949.52	-
(2.b)	10	0.0	0.0	0.0	36.67	987.12	-

VIII. CONCLUSION

In this paper we have studied $GSPS$ which is a generalization of the strictly periodic scheduling problem. After giving some properties, we have designed a simple technique to compute a lower bound (TLB) of good quality and obtained immediately. We also have proposed two mathematical formulations for $GSPS$. We have shown that "strong formulation" is more appropriated to solve $GSPS$. The lower bound derived from the resolution of the LP-relaxation of $GSPS$ "strong formulation" gives a value equal to (TLB) if $H \geq lcm$ and of good quality in any case. Finally, three greedy algorithms are suggested to get feasible solutions. Heuristic I and II provide upper bound of less quality than Heuristic III. Nevertheless, Heuristic III is very time consuming. A possible way to get further improvement of the computation time of the best feasible solution, would be to generate valid cuts so as to accelerate the resolution of each sub-problem.

REFERENCES

[1] M. H. Ammar and J. W. Wong, *On the optimality of cyclic transmission in teletext systems*, IEEE Transactions on Communication, COM-35 1, 68–73, 1987.

[2] M. H. Ammar and J. W. Wong, *The design of teletext broadcast cycles*, Performance Evaluation 5 (4), 235–242, 1985.

[3] J. Anderson and P. Holman and A. Srinivasan *Fair scheduling of real time tasks on multiprocessors*, Handbook of scheduling : Algorithms, Models and Performance analysis, 31.1–31.21, J. Leung, ed., 2004.

[4] S. Anily and C. A. Glass and R. Hassin, *Scheduling of maintenance services to three machines*,Annals of Operations Research 85, 375–391, 1999.

[5] A. Bar-Noy and V. Dreizin and B. Patt-Shamir, *Efficient algorithm for periodic scheduling*, Computer Networks 45, 155–173, 2004.

[6] A. Bar-Noy and B. Randeep and J. Naor and B. Schieber, *Minimizing service and operation cost of periodic scheduling*, 9th ACM Symposium on Discrete Algorithms, 11–20, 1998.

[7] S. Baruah and J. Gehrke and C. G. Plaxton and I. Stoica and H. Abdel-Wahab and K. Jeffay, *Fair on-line scheduling of a dynamic set of tasks on a single resource*, Information Processing Letters 64 (1), 43–51, 1997.

[8] S. Baruah and N. Cohen and C. G. Plaxton and D. Varvel, *Proportionate progress: a notion in resource allocation*, Algorithmica 15 (6), 600–625, 1996.

[9] S. Baruah and J. Gehrke and C. G. Plaxton, *Fast scheduling of periodic traks on multiple resources*, 9th International Parallel Processing Symposium, 280–288, 1995.

[10] A. M. Campbell and J. R. Hardin, *Vehicle minimization for periodic deliveries*, European Journal of Operational Research 165, 668–684, 2005.

[11] M. Gaudioso and G. Paoletta, *A heuristic for the periodic vehicle routing problem*, Transportation Sciences 26, 86–92, 1992.

[12] M. Gaudioso and G. Paoletta and S. Sanna, *Management of periodic demands in distribution systems*, European Journal of Operational Research 20, 234–238, 1985.

[13] M. B. Jones and D. Rosu and M-C. Rosu, *Cpu reservations and time constraints: efficient, predictable scheduling of independant activities*, 16th ACM Symposium on Operating Systems Principles, 198–211, 1992.

[14] V. Kats and E. Levner, *Minimizing the number of vehicles in periodic scheduling: the non-euclidean case*, European Journal of Operational Research 107, 371–377, 1998.

[15] V. Kats and E. Levner, *Minimizing the number of robots to meet a given cyclic schedule*, Annals of Operations Research 69, 209–226, 1997.

[16] C. Kenyon and N. Schabanel and N. Young, *Polynomial-time approximation scheme for data broadcast*, 32nd Annual ACM Symposium on Theory of Computing, 659–666, 2000.

[17] C. Kenyon and N. Schabanel, *The data broadcast problem with non-uniform transmission times*, 10th Annual ACM Symposium on Discrete Algorithms, 547–556, 1999.

[18] S. Khanna and S. Zhou, *On indexed data broadcast*, 30th Annual ACM Symposium on Theory of Computing, 463–472, 1998.

[19] J. Korst, *Periodic multiprocessor scheduling*, Ph. D. thesis, Technical report University of Eindhoven, 1992.

[20] C. J. Liao and W. J. Chen, *Single-machine scheduling with periodic maintenance and nonresumable jobs*, Computers and Operations Research 30, 1335–1347, 2003.

[21] C. L. Liu and J. W. Layland, *Scheduling algorithms for multiprocessors in a hard-real-time environment*, Journal of the ACM 20 (1), 46–61, 1973.

[22] M. J. Negreiros and A. E. Xavier and A. F. Xavier and P. Michelon, *A framework of computational systems and optimization models for the prevention and combat of dengue*, International Transactions in Operations Research, to appear.

[23] K. S. Park and D. K. Yun, *Optimal scheduling of periodic activities*, Operations Research 33, 690–695, 1985.

[24] W. F. J. Verhaegh and P. E. R. Lippens and E. H. L. Aarts and J. L. van Meerbergen and A. van der Werf, *The complexity of multidimensional periodic scheduling*, Discrete Applied Mathematics 89, 213–242, 1998.

[25] W. Wei and C. L. Liu, *On periodic maintenance problem*, Operations Research Letters 2, 90–93, 1983.