Time Indexed Formulations for Scheduling Chains on a Single Machine: An Application to Airborne Radars

Philippe Baptiste, Ruslan Sadykov

Time Indexed Formulations for Scheduling Chains on a Single Machine: An Application to Airborne Radars[☆]

Philippe Baptiste[a,*], Ruslan Sadykov[b,c]

[a] *Ecole Polytechnique, CNRS LIX UMR 7161, 91128 Palaiseau, France*
[b] *INRIA Bordeaux — Sud-Ouest, France*
[c] *Institut de Mathématique de Bordeaux, 351 cours de la Liberation, 33405 Talence, France*

Abstract

Airborne radars are widely used to perform a large variety of tasks in an aircraft (searching, tracking, identifying targets, etc.) Such tasks play a crucial role for the aircraft and they are repeated in a "more or less" cyclic fashion. This defines a scheduling problem that impacts a lot on the quality of the radar output and on the overall safety of the aircraft.

In our model, jointly defined with "Thales Airborne Systems", the radar executes the schedule of the current time frame while the schedule of the next frame is computed. The radar is a single machine and a radar task is a job to be scheduled on the machine. A job consists of a chain of operations with identical processing times. The operations of the same job should be ideally scheduled with a given frequency, and any deviation from this frequency is penalized using a V-shape function. Our objective is then to minimize the total penalty.

In this paper, we present three time indexed formulations for the problem. Two of the formulations are compact and can be solved directly by a Mixed-Integer Programming solver. The third formulation relies on a Branch-and-Price algorithm. Theoretical and experimental comparisons of the formulations are reported.

Key words: Scheduling, Integer Programming, OR in military

[☆]This work has been supported by a DGA–Ecole Polytechnique research grant, and by ANR Alpage

[*]Corresponding author. Tel: +33 1 69 33 40 56. Fax: +33 1 69 33 41 58.
Email addresses: Phlippe.Baptiste@polytechnique.fr (Philippe Baptiste), Ruslan.Sadykov@inria.fr (Ruslan Sadykov)

1. Airborne radars scheduling

A radar is a system using radio-waves to detect the presence of objects in a given volume of space. It can also compute the range (distance) as well as the relative radial velocity of these objects. Airborne radars consist of a transmitter, a single antenna and a receiver. The transmitter generates radio-waves which are sent out in a narrow beam by the antenna in a specific direction. Objects located in the beam intercept this signal and scatter the energy in all directions. A portion of this energy is scattered back to the receiver of the radar listening to all potential echoes. See [13] and [16] for a detailed description of airborne radars.

Recent radars have an Electronically Steered Antenna (ESA). An ESA is a planar array antenna made of many individual radiating elements. Unlike a mechanically steered antenna, an ESA lies in a fix position on the aircraft. The phase of the radio-waves is controlled electronically so that the radar beam lights up the desired direction.

One of the key advantages of ESA is that the beam is extremely agile. As it is not subjected to the mechanical inertia of the antenna, it can be moved instantaneously from one part of the space to another, even outside the search domain. Moreover, the radar can switch almost instantaneously to an appropriate waveform.

The following tasks have to be performed by an airborne radar (see Figure 1).

- *Research.* The radar is sweeping across a domain to detect the potential presence of targets inside.
- *Tracking.* The radar is closely monitoring the behavior of targets (initially detected at the Research stage).
- *Data Link.* The radar is used as a communication tool with other platforms.
- *Calibration.* The radar is performing cyclic calibration to ensure a high reliability level.

Research, tracking, data link and calibration require incompatible waveforms. So all corresponding tasks have to be *scheduled to ensure they do not overlap in time*. Moreover, research, tracking, data link and calibration tasks have to be repeated in a more or less regular fashion. The execution

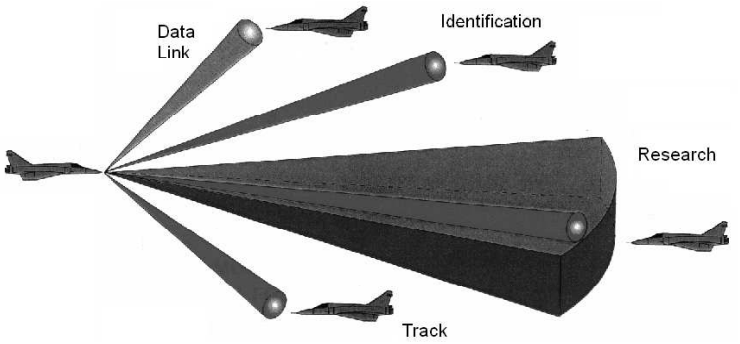

Figure 1: Functions of the ESA radars

of any single radar operation is called a *dwell*. Depending on the task, the periodicity constraint can be extremely important or not. For instance, as it is absolutely forbidden to lose a tracked target, tracking dwells have to be repeated with high regularity. Likewise, data link dwells are played at regular intervals, but there, the regularity constraint is much higher. Finally, to ensure a surveillance of great quality, in any circumstances, the radar has to play at least a minimum amount of research dwells.

The use of only one device for all radar functions results in certain limitations. For instance, as different dwells cannot be played at the same time, while the radar is used in tracking mode, the pilot has no knowledge of the evolution of the tactical situation, that is new targets coming and so on. Also, the more targets to track, the less time for the search. We are interested with the scheduling of the dwells to make the radar more efficient while meeting the constraints informally described above.

Following the framework of Barbaresco [3], Winter and Baptiste [18] have proposed a formal model based on cost functions that describes the problem. In this model, dwells are scheduled on a frame duration and are executed while the next schedule is computed. For each task, we know the duration of its dwells and the desired periodicity of their repetition. We need to schedule a certain number of dwells of each type within the next time frame. As dwells cannot overlap in time, usually it is not possible to respect the desired dwell periodicity of each task. Therefore a deviation from ideal periodicity is permitted but penalized in proportion to the deviation. For different tasks, penalties differ according to the periodicity importance. The objective is to find a feasible schedule which minimizes the total penalty.

This model has been used in [18] to compare several fast scheduling heuristics. In this paper, we propose methods to find optimal solutions or good lower bounds. Of course, these methods cannot be used in real-time however, such lower bounds are very useful to assess the intrinsic quality of the heuristics.

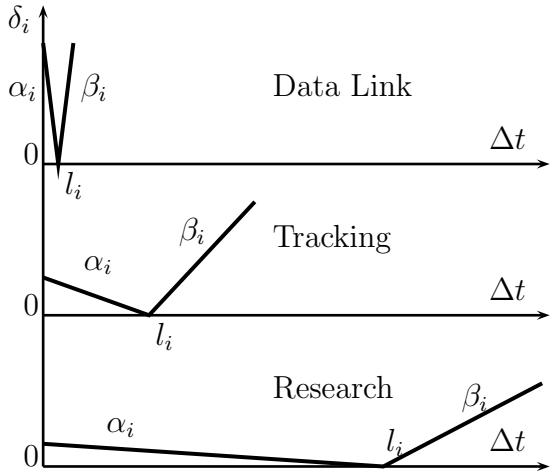

Figure 2: Penalty Functions

Formal definition of the problem

A set $N = \{1, \ldots, n\}$ of jobs (which represent radar tasks) has to be processed on a single machine. Each job $i \in N$ consists of a chain $(O_{i0}, O_{i1}, \ldots, O_{i,n(i)})$ of $n(i) + 1$ identical operations (representing dwells) which should be processed in the same order within a given time horizon $[0, h]$. We denote by $N(i)$, $i \in N$, the set $\{1, \ldots, n(i)\}$ and by H the set $\{0, 1, \ldots, h-1\}$. Processing an operation cannot be interrupted, and the machine can process at most one operation at a time. The processing time of each operation of job i is equal to $p_i > 0$. The starting time of operation O_{ij} in schedule S will be denoted as S_{ij}. For each job $i \in N$, the starting time S_{i0} of operation O_{i0} is given and cannot be changed.

The operations of each job $i \in N$ should be ideally scheduled with a given frequency l_i (the difference between starting times of two consecutive operations of a job should be ideally equal to l_i). For each job $i \in N$, a penalty function $\delta_i(x)$ for the difference between starting times of two consecutive operations of i is introduced. We set

$$\delta_i(x) = \max\{\alpha_i(l_i - x), \beta_i(x - l_i)\}$$

Examples of penalty functions used for different radar tasks are presented in Figure 2. So, given $\{n(i), p_i, l_i, \alpha_i, \beta_i, S_{i0}\}_{i\in N}$ and h, the problem consists in finding a feasible schedule S which minimizes the total penalty

$$F(S) = \sum_{i\in N} \sum_{j\in N(i)} \delta_i(S_{ij} - S_{i,j-1}) \tag{1}$$

We assume that all data except $\{\alpha_i, \beta_i\}_{i\in N}$ are integers. If something from these data is rational, to obtain an equivalent instance in which all necessary data are integers, we can multiply $\{p_i, l_i, S_{i0}\}_{i\in N}$ and h by the least common multiple of their denominators (and divide $\{\alpha_i, \beta_i\}_{i\in N}$ by the same value). However, this transformation increases the length of the time horizon, i.e. the size of the instance.

Complexity: If each job i consists of two operations O_{i0} and O_{i1} and if h is large enough, then the problem reduces to the classical earliness-tardiness problem on a single machine [6] denoted as $1||\sum \alpha_j E_j + \beta_j T_j$ in the standard scheduling notation [9]. This problem is NP-hard in the strong sense [8].

Cyclic variant of the problem

In this variant, we want to find a cyclic schedule, i.e. a schedule which is repeated every certain period of time. The penalty of a cyclic schedule depends also on the difference between the first and the last operations of each job (the "zero" operations are absent).

This variant of the problem is not considered in the paper. We introduce it to show that it has relations with some real-time scheduling problems. For example, the cyclic variant is an optimization version of the DCTS non-preemptive scheduling problem [10] (DCTS stands for *Distance-Constraint Task System*) and the *Periodic Maintenance Problem* [17] (PMP).

If we use our notation, the PMP consists in finding a cyclic schedule with zero penalty. In the DCTS scheduling problem, we need to find a cyclic schedule in which, for each job $i \in N$, the distance between every two consecutive operations does not exceed l_i (a schedule with zero penalty given that $\alpha_i = 0, \forall i \in N$). A special case of the latter problem, in which all processing times are equal to 1, is called the *Pinwheel Scheduling Problem* [5].

Structure of the paper

In Section 2 we present the time indexed formulation of the problem [15]. In Section 3 we introduce a new objective function and we show that it leads

to the same set of optimal schedules as for the initial objective function. This leads to an alternative time indexed formulation that contains far fewer variables. In Section 4, we prove that, under some reasonable assumption, the Linear Programming (LP) relaxation of this alternative formulation always provides tighter bounds than the initial LP relaxation. Section 5 concerns a Dantzing-Wolfe reformulation of the problem. A branch-and-price algorithm for solving this reformulation is presented in Section 6. Experiments have been executed on real life and random instances and are reported in Section 7. Conclusions are drawn in Section 8.

2. Time indexed formulation

First, we formulate the problem as a Integer Linear Programming (ILP) problem in a straightforward way. We use the standard time indexed formulation for non-preemptive single-machine scheduling problems [15]. The binary variable X_{ijt}, $i \in N$, $j \in N(i)$, $t \in H$, takes value 1 if and only if operation O_{ij} is started at time t. The continuous variable S_{ij}, $i \in N$, $j \in N(i)$, equals the starting time of operation O_{ij}. The values of variables S_{i0}, $i \in N$, are fixed. Finally, the continuous variable W_{ij}, $i \in N$, $j \in N(i)$, represents the value $\delta_i(S_{ij} - S_{i,j-1})$. We now present the time indexed ILP formulation.

$$(\mathbf{TI}): \quad \min \quad \sum_{i \in N} \sum_{j \in N(i)} W_{ij} \tag{2}$$

$$s.t. \quad \sum_{t=0}^{h-p_i} X_{ijt} = 1, \quad \forall i \in N, j \in N(i) \cup \{0\}, \tag{3}$$

$$\sum_{i \in N} \sum_{j \in N(i) \cup \{0\}} \sum_{t'=\max\{t-p_i+1,0\}}^{t} X_{ijt'} \leq 1, \quad \forall t \in H, \tag{4}$$

$$S_{ij} = \sum_{t \in H} t \cdot X_{ijt}, \quad \forall i \in N,\ j \in N(i) \cup \{0\}, \tag{5}$$

$$S_{i,j-1} + p_i \leq S_{ij}, \quad \forall i \in N,\ j \in N(i), \tag{6}$$

$$W_{ij} \geq \alpha_i(l_i - S_{ij} + S_{i,j-1}), \quad \forall i \in N,\ j \in N(i) \tag{7}$$

$$W_{ij} \geq \beta_i(S_{ij} - S_{i,j-1} - l_i), \quad \forall i \in N,\ j \in N(i), \tag{8}$$

$$X_{ijt} \in \{0,1\}, \quad \forall i \in N,\ j \in N(i) \cup \{0\},\ t \in H. \tag{9}$$

The objective function (2) follows from (1). The constraints (3) state that each operation should start and finish within interval $[0, h]$. The constraints (4) prevent overlapping (ensure that at most one operation is processed at a time). The constraints (5) link the variables X and S. The constraints (6) guarantee that the operations of a job are processed in the appropriate order. The constraints (7)-(8) are used to compute the values of the variables W.

One of the disadvantages of the (TI) formulation is that we use distinct variables for the operations of the same job. As all the operations of a job are identical, it would be more natural and more efficient to use the same set of variables. For example, we can introduce binary variables X_{it}, $i \in N$, $t \in H$, that state that one operation of job i is started at time t. However, using these variables, it is not *a priori* possible to express the objective function (1). To overcome this difficulty, in the following subsection, we introduce another objective function which is equivalent to (1).

3. Alternative time-indexed formulation for scheduling chains on a single machine

Consider a schedule S and let $n_i(t)$, $i \in N$, $t \in H$, be the number of the operations of job i started within interval $[t - l_i + 1, \ldots, t]$, as shown in Figure 3. We define now the sets

$$\Delta_i = \{S_{i0}, S_{i0} + 1, \ldots, S_{i,n(i)} + l_i - 1\}, \quad \forall i \in N.$$

Note that, if the operations of job i are scheduled with the ideal frequency l_i then $n_i(t) = 1$ for all $t \in \Delta_i$. We are ready to introduce an alternative penalty function γ_i:

$$\gamma_i(t) = \max\left\{\alpha_i\left(n_i(t) - 1\right), \beta_i\left(1 - n_i(t)\right)\right\}$$

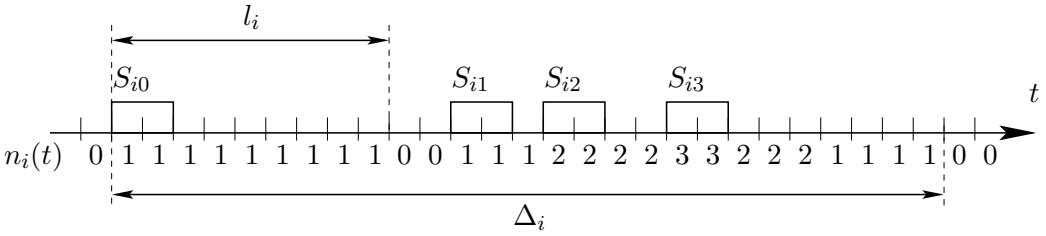

Figure 3: A schedule S for job i ($n(i) = 3$, $p_i = 2$, $l_i = 9$)

As shown in Theorem 1, the penalty functions δ and γ give the same total penalty for any schedule.

Theorem 1. *For any schedule S_i of job i,*

$$\sum_{j\in N(i)} \delta_i(S_{ij} - S_{i,j-1}) = \sum_{t\in\Delta_i} \gamma_i(t).$$

Proof. We denote $B_i = \{t \in \Delta_i :\ n_i(t) = 0\}$ and $A_i = \{t \in \Delta_i :\ n_i(t) \geq 1\}$. Note that $B_i \cup A_i = \Delta_i$. Therefore, we have

$$\sum_{t\in\Delta_i} \gamma_i(t) = \alpha_i \cdot \sum_{t\in A_i} (n_i(t) - 1) + \beta_i \cdot |B_i|. \tag{10}$$

We also denote $D_{ij} = S_{ij} - S_{i,j-1}$, $j \in N(i)$. Remember that, for all $j \in N(i)$,

$$\delta_i(D_{ij}) = \alpha_i \cdot \underbrace{\max\{l_i - D_{ij}, 0\}}_{=E_{ij}} + \beta_i \cdot \underbrace{\max\{D_{ij} - l_i, 0\}}_{=T_{ij}}, \tag{11}$$

where E_{ij} is the relative earliness of operation O_{ij} and T_{ij} is its relative tardiness. From (10) and (11) it follows that, to prove the proposition, it suffices to show that $\sum_{j\in N(i)} T_{ij} = |B_i|$ and $\sum_{j\in N(i)} E_{ij} = \sum_{t\in A_i} n_i(t) - |A_i|$.

We partition the operations of job i into blocks of consecutive operations in the following way. A block contains operations $(O_{ij'}, O_{i,j'+1}, \ldots, O_{ij''})$. The first operation $O_{ij'}$ in a block is either late ($T_{ij'} > 0$ and $E_{ij'} = 0$) or the "zero" operation ($j' = 0$). All other (if any) operations $(O_{i,j'+1}, \ldots, O_{ij''})$ in a block are early or on time, i.e. $E_{ij} \geq 0$ and $T_{ij} = 0$.

Consider such a block of consecutive operations. We define $A'_i(j', j'') = \{S_{ij'}, \ldots, S_{ij''} + l_i - 1\}$ and, if $j' \neq 0$, $B'_i(j', j'') = \{S_{i,j'-1} + l_i, \ldots, S_{ij'} - 1\}$. Note that $A'_i(j', j'') \subseteq A_i$ and $B'_i(j', j'') \subseteq B_i$. Clearly, $|B'_i(j', j'')| = D_{ij'} - l_i = T_{ij'}$. Summing this equality over all blocks except the first one, we obtain $|B_i| = \sum_{j\in N(i)} T_{ij}$.

Then, each operation in the block is counted l_i times in the sum $\sum_{t\in A'_i(j',j'')} n_i(t)$. Therefore,

$$\begin{aligned} \sum_{t\in A'_i(j',j'')} n_i(t) - |A'| &= l_i(j'' - j' + 1) - (S_{ij''} + l_i - S_{ij'}) \\ &= \sum_{j=j'+1}^{j''} (l_i - S_{ij} + S_{i,j-1}) = \sum_{j=j'+1}^{j''} E_{ij}. \end{aligned} \tag{12}$$

Summing (12) over all blocks, we obtain $\sum_{t\in A_i} n_i(t) - |A_i| = \sum_{j\in N(i)} E_{ij}$. $\square$

Using penalty function γ, we will alter the formulation (TI). We consider an extended time horizon H_i for each job $i \in N$:

$$H_i = \{S_{i0}, \cdots, h - p_i + l_i\}.$$

The integer variables X_{it}, $i \in N$, $t \in H$, are now the number of the operations of job i started before or at time t. Continuous variables W_{it}, $i \in N$, $t \in H_i$, represent the value $\gamma_i(t)$. The main issue with the alternative objective function is that the starting times $S_{i,n(i)}$, $i \in N$, are not known a priori. Therefore, we need to use *additional binary variables* E_{it}, $i \in N$, $t \in H_i$, *which indicate whether* $t \in \Delta_i$. We are ready to present the alternative time indexed ILP formulation.

$$(\mathbf{TIA}): \quad \min \quad \sum_{i\in N}\sum_{t\in H_i} W_{it} \tag{13}$$

$$s.t. \quad X_{i,S_{i0}} = 1, \quad X_{i,S_{i0}-1} = 0, \quad \forall i \in N. \tag{14}$$

$$X_{i,t-1} \le X_{it}, \quad \forall i \in N,\ t \in H \setminus \{0\}, \tag{15}$$

$$X_{i,h-p_i} = n(i) + 1, \quad \forall i \in N, \tag{16}$$

$$\sum_{i\in N} X_{it} - \sum_{\substack{i\in N,\\ t-p_i\ge 0}} X_{i,t-p_i} \le 1, \quad \forall t \in H, \tag{17}$$

$$E_{it} \ge X_{i,t-l_i+p_i} - X_{i,t-l_i},\ \forall i \in N,\ t \in H_i,\ t \ge S_{i0} + l_i, \tag{18}$$

$$E_{it} \le E_{i,t-1} \le 1, \quad \forall i \in N,\ t \in H_i \setminus \{S_{i0}\}, \tag{19}$$

$$W_{it} \ge \alpha_i(X_{it} - X_{i,t-l_i} - E_{it}), \quad \forall i \in N,\ t \in H_i, \tag{20}$$

$$W_{it} \ge \beta_i(E_{it} - X_{it} + X_{i,t-l_i}), \quad \forall i \in N,\ t \in H_i, \tag{21}$$

$$X_{it} \in \mathbb{Z}_+, \quad \forall i \in N,\ t \in H. \tag{22}$$

The constraints (15) reflect the nature of the variables X. The constraints (16) state that exactly $n(i)+1$ operations of job i should be processed within interval $[S_{i0}, h]$. The constraints (17) prevent overlapping. The constraints (18)-(19) are used to compute the values for the variables E. Once variables X_{it} is equal to 1, variables $E_{it'}$, $S_{i0} \le t' \le t + l_i - 1$, are forced to be equal to 1 by these constraints. Note that we do not need to impose the integrality on the variables E. Whenever the variables X are integer, the variables E are also integer. The constraints (20)-(21) are used to compute the values of the variables W.

The continuous relaxation of (TIA) is a relaxation and a strengthening of the continuous relaxation of (TI). The first fact comes from the introduction of variables E. The strengthening is due to the modified objective function which implicitly imposes the constraints $\sum_{t'=0}^{t} X_{i,j-1,t} < 1 \Rightarrow \sum_{t'=0}^{t} X_{ijt} = 0$, where variables X are from the formulation (TI). In the next section, we show formally that (TIA) is stronger than (TI) if we can do without variables E.

4. Theoretical comparison of the formulations (TI) and (TIA)

We show that, under some reasonable assumptions, the LP relaxation of (TIA) is at least as strong as the LP relaxation of (TI).

Note that the difference between the total relative tardiness T_i and the total relative earliness E_i of job $i \in N$ depends only on l_i, $n(i)$, S_{i0}, $S_{i,n(i)}$ which are fixed:

$$\begin{aligned} T_i - E_i &= \sum_{j \in N(i)} \max\{S_{ij} - S_{i,j-1} - l_i, 0\} - \sum_{j \in N(i)} \max\{l_i - S_{ij} + S_{i,j-1}, 0\} \\ &= \sum_{j \in N(i)} (S_{ij} - S_{i,j-1} - l_i) = S_{i,n(i)} - S_{i0} - l_i n(i). \end{aligned}$$

Then, $\alpha_i E_i + \beta_i T_i = \alpha_i E_i + \beta_i E_i - \beta_i E_i + \beta_i T_i = \beta_i (T_i - E_i) + (\alpha_i + \beta_i) E_i$. By Theorem 1, we have $E_i = \sum_{t \in \Delta_i} \max\{n_i(t) - 1, 0\}$. The LP relaxations of the formulations (TI) and (TIA) can be rewritten to respectively (SC) and (SCA) by putting the constant part $\beta_i(T_i - E_i)$ for every job i directly to the objective function:

$$(\mathbf{SC}): \quad \min \quad \sum_{i \in N} \sum_{j \in N(i)} W_{ij} + \sum_{i \in N} \beta_i \cdot \big(S_{i,n(i)} - S_{i0} - l_i n(i)\big) \tag{23}$$

$$s.t. \quad (3) - (6),$$

$$W_{ij} \geq (\alpha_i + \beta_i) \cdot (l_i - S_{ij} + S_{i,j-1}), \ \forall i \in N, \ j \in N(i), \tag{24}$$

$$W_{ij} \geq 0, \quad \forall i \in N, \ j \in N(i). \tag{25}$$

$$0 \leq X_{ijt} \leq 1, \quad \forall i \in N, \ j \in N(i), t \in H. \tag{26}$$

$$(\mathbf{SCA}): \quad \min \quad \sum_{i \in N} \sum_{t \in \Delta_i} W_{it} + \sum_{i \in N} \beta_i \cdot \big(S_{i,n(i)} - S_{i0} - l_i n(i)\big) \tag{27}$$

$$s.t. \quad (14) - (17),$$

$$X_{i,S_{i,n(i)}} = n(i) + 1, \quad X_{i,S_{i,n(i)}-1} = n(i), \quad \forall i \in N. \tag{28}$$

$$W_{it} \geq (\alpha_i + \beta_i) \cdot (X_{it} - X_{i,t-l_i} - 1), \ \forall i \in N, \ t \in \Delta_i, \tag{29}$$

$$W_{it} \geq 0, \quad \forall 0 \leq X_{it} \leq n(i) + 1, \quad i \in N, \ t \in H. \tag{30}$$

We denote by ν_F^* and $\nu_F(\bar{X})$ the values of the objective function of an optimal solution and solution $\bar{X}$ of formulation (F).

Theorem 2. *If the starting times of the last operation of each job are fixed, then* $\nu_{SC}^* \leq \nu_{SCA}^*$.

Proof. To prove the theorem, we will show that, given a feasible solution X^* of the formulation (SCA), a feasible solution X' of the the formulation (SC) can be found such that $\nu_{SC}(X') \leq \nu_{SCA}(X^*)$.

For a given solution X_i^* of (SCA), X_i' is built in the following way:

$$X'_{ijt} = \min\left\{1 - \sum_{t'=0}^{t-1} X'_{ijt'}, X^*_{it} - X^*_{i,t-1} - \sum_{k=1}^{j-1} X'_{ikt}\right\},\ \forall i \in N, j \in N(i), t \in H.$$

Basically, every positive value $(X_{it}^* - X_{i,t-1}^*)$, $t \in H$, is "distributed" among variables X'_{ijt}, $j \in N(i)$, in such a way that $X'_{ijs} > 0$ and $X'_{i,j+1,t} > 0$ only if $s \leq t$. It is easy to see that $X'_{ijt} \in [0,1]^{n(i)\times h}$, and the constraints (3) are satisfied for X_i'. As $\sum_{j\in N(i)} \sum_{t'=t-p_i+1}^{t} X'_{ijt'} = X^*_{it} - X^*_{i,t-p_i}$, $t \in H$, and X_i^* satisfies (17), X_i' satisfies the constraints (4). Then, for each $j \in N(i)$ and $t \in H$, we have

$$\sum_{t'=t}^{h-1} X'_{i,j-1,t'} \leq \sum_{t'=t+p_i}^{h-1} X'_{ijt'}, \tag{31}$$

otherwise we would have $X^*_{i,t+p_i} - X^*_{it} \geq \sum_{t'=t+1}^{t+p_i}(X'_{i,j-1,t} + X'_{ijt}) > 1$, so that constraints (17) would be violated. Given $j \in N(i)$, we sum up (31) for all $t \in [1,h]$ and obtain

$$\begin{aligned} S'_{i,j-1} &= \sum_{t=0}^{h} tX'_{i,j-1,t} \leq \sum_{t=p_i}^{h}(t-p_i)X'_{ijt} \\ &= \sum_{t=p_i}^{h} tX'_{ijt} - p_i \sum_{t=p_i}^{h} X'_{ijt} = S'_{ij} - p_i. \end{aligned}$$

X_i' satisfies then constraints (6).

It remains to show that, for each $i \in N$, $\nu_{SC}(X_i') \leq \nu_{SCA}(X_i^*)$, i.e.

$$\sum_{j\in N(i)} \max\left\{l_i - S'_{ij} + S'_{i,j-1}, 0\right\} \leq \sum_{t\in\Delta_i} \max\left\{X^*_{it} - X^*_{i,t-l_i} - 1, 0\right\}.$$

Consider job $i \in N$. Similarly to the arguments of the proof of Theorem 1, we again partition the operations of job i into a set M of blocks. Remember that the first operation in a block either is tardy or is the "zero" operation, and the other operations in a block are early or on-time.

Consider a block $k \in M$ containing operations $(O_{i,j'_k}, O_{i,j'_k+1}, \ldots, O_{i,j''_k})$, where the number of operations is at least two ($j'_k < j''_k$). Let t^1_k be the last time such that $X'_{ij'_k t} > 0$ and t^2_k be the first time such that $X'_{ij''_k t} > 0$. We denote $\Delta_{ik} = \{t^1_k, \ldots, t^2_k + l_i - 1\}$. Note that $t^2_k + l_i \leq t^1_{k+1}$, otherwise $S'_{ij''_k} \geq t^2_k > t^1_{k+1} - l_i \geq S'_{ij'_{k+1}} - l_i$ which would contradict the fact that operation $O_{ij'_{k+1}}$ is tardy. Therefore,

$$\begin{aligned}
\nu_{SCA}(X^*_i) &= \sum_{t\in\Delta_i} \max\{X^*_{it} - X^*_{i,t-l_i} - 1, 0\} \\
&\geq \sum_{k\in M} \sum_{t\in\Delta_{ik}} \max\{X^*_{it} - X^*_{i,t-l_i} - 1, 0\} \\
&\geq \sum_{k\in M} \left(\sum_{t\in\Delta_{ik}} (X^*_{it} - X^*_{i,t-l_i}) - (t^2_k + l_i - t^1_k) \right)
\end{aligned}$$

We denote

$$\eta_k = \sum_{t\in\Delta_{ik}} n_i(t) = \sum_{t\in\Delta_{ik}} \left(X^*_{it} - X^*_{i,t-l_i}\right)$$

Note that the contribution of each variable X_{ijt} of the formulation (TI) to η_k is equal to

$$\sum_{t\in H} X'_{ijt} \cdot \max\left\{l_i - \max\{t^1_k - t, t - t^2_k, 0\}, 0\right\}.$$

Then, by definition of t^1_k and t^2_k and by construction of X', $\sum_{t=t^1_k}^{t^2_k} X'_{ijt} = 1$ for all $j'_k < j < j''_k$. Therefore,

$$\begin{aligned}
\eta_k &\geq \sum_{j=j'_k+1}^{j''_k-1} l_i + \sum_{t\in H} X'_{ij'_k t} \max\{l_i - t^1_k + t, 0\} + \sum_{t\in H} X'_{ij''_k t} \max\{l_i - t + t^2_k, 0\} \\
&\geq l_i(j''_k - j'_k - 1) + l_i - t^1_k + S'_{ij'_k} + l_i - S'_{ij''_k} + t^2_k \\
&= l_i(j''_k - j'_k + 1) + S'_{ij'_k} + S'_{ij''_k} - t^1_k + t^2_k.
\end{aligned}$$

i	p_i	l_i	$n(i)$	S_{i0}	α_i	β_i
1	9	12	8	5	1	1
1	5	5	6	0	1	1

Table 1: Data for a counterexample instance for $\nu^*_{TI} \leq \nu^*_{TIA}$

Finally,

$$\nu_{SCA}(X_i^*) \geq \sum_{k \in M} \left(\eta_k - t_k^2 - l_i + t_k^1\right) \geq \sum_{k \in M} \sum_{j=j'_k+1}^{j''_k} \left(l_i - S'_{ij} + S'_{i,j-1}\right),$$

which is exactly equal to $\nu_{SC}(X'_i)$. □

Unfortunately, the inequality $\nu^*_{SC} \leq \nu^*_{SCA}$ does not always hold when values $S_{i,n(i)}$, $i \in N$, are not known in advance. For example, if we take the instance with the data shown in Table 1 ($h = 120$), then $\nu^*_{TI} \approx 36.1 > 33.2 \approx \nu^*_{TIA}$ (these values a computed using an LP solver).

Nevertheless, Theorem 2 suggests that the LP relaxation of the alternative time indexed formulation is usually stronger than the one of the standard time indexed formulation. This is confirmed by the numerical experiments reported in Section 7.

5. Dantzing-Wolfe reformulation

The objective function (1) is the sum of the penalty functions for all the jobs. So, for each job and each possible schedule of this job, we are able to associate a binary variable stating whether this schedule is used or not. Using these variables, the formulation (TI) can be reformulated and decomposed into a master problem and n pricing sub-problems (one for each job). We now present details of this Dantzing-Wolfe formulation.

Note that another variant of Dantzig-Wolfe reformulation have been used to solve (TI) by Akker et al. [2].

In the following, S_i^k, $k \in K_i$, is the k-th possible schedule $(S_{i0}^k, S_{i0}^k, \ldots, S_{i,n(i)}^k)$ for job $i \in N$, and w_i^k be the corresponding penalty function. Let also $\chi(i,k,t)$ be a characteristic function which indicates whether schedule S_i^k is "active" at time t:

$$\chi(i,k,t) = \begin{cases} 1, & \text{if} \quad t \in \bigcup_{j=0}^{n(i)} [S_{ij}^k, S_{ij}^k + p_i - 1], \\ 0, & \text{otherwise} \end{cases}$$

Binary variable X_{ik}, $i \in N$, $k \in K_i$, represents whether schedule S_i^k is included in the solution. We are ready to write down the reformulation (MP).

$$(\mathbf{MP}) \quad \min \quad \sum_{i \in N} \sum_{k \in K_i} w_i^k X_{ik} \tag{32}$$

$$s.t. \quad \sum_{i \in N} \sum_{k \in K_i} \chi(i,k,t) X_{ik} \leq 1, \quad \forall t \in H. \tag{33}$$

$$\sum_{k \in K_i} X_{ik} = 1, \quad \forall i \in N, \tag{34}$$

$$X_{ik} \in \{0,1\}, \quad \forall i \in N,\ k \in K_i. \tag{35}$$

The constraints (33) forbid overlapping, i.e. state that at each time at most one schedule is active. The constraints (34) guarantee that each job is processed.

6. Branch-and-Price algorithm

The formulation (MP) cannot be directly solved by a Mixed Integer Programming (MIP) solver, as the number of variables is huge. The standard approach for solving such problems is called Branch-and-Price [4] in which the LP relaxation of (MP) is solved by column generation. It consists in iterating between a solution of the LP relaxation (RMP) of the master problem involving only a subset of columns, and the solution of the pricing problem. The pricing sub-problem for job $i \in N$ is used to generate an additional column corresponding to a schedule in K_i with a negative reduced cost, or to prove that no such column exists.

Given a solution of (RMP), let $\pi \in \mathbb{R}^h$ be the vector of optimal values of the dual variables which correspond to the constraints (33), and $\mu \in \mathbb{R}^n$ be the vector of the optimal values of the dual variables which correspond to the constraints (34). Note that the elements of vector π are non-negative, as the constraints (33) are inequalities. Then, the reduced cost for column X_{ik} is equal to

$$w_i^k + \sum_{t \in H} \chi(i,k,t) \pi_t - \mu_i. \tag{36}$$

6.1. Resolution of the pricing sub-problem

To verify if there is a column with a negative reduced cost, we solve the pricing problem and search for a schedule S_i^k minimizing (36). Given the structure of the reduced cost function (36), the pricing problem can be decomposed into sub-problems, one for each job $i \in N$. Each sub-problem (PS_i) can be formally written as

$$(\mathbf{PS}_i) \quad \min \quad \sum_{j \in N(i)} \delta_i(S_{ij} - S_{i,j-1}) + \sum_{j=0}^{n(i)} \sum_{t=S_{ij}}^{S_{ij}+p_i-1} \pi_t - \mu_i \tag{37}$$

$$s.t. \quad S_{ij} \geq S_{i,j-1} + p_i, \quad \forall j \in N(i), \tag{38}$$

$$S_{i,n(i)} + p_i \leq h. \tag{39}$$

Basically, this sub-problem consists in finding a schedule of the operations of job i which minimizes the sum of the penalty plus a dual cost involving an arbitrary function π_t of the starting times of the operations.

As the sequence of the operations is fixed, the problem (PS_i) can be solved by a dynamic programming approach. Let $R_{ij}(t)$ denote the minimal reduced cost of a schedule of operations $O_{ij}, \ldots, O_{i,n(i)}$ assuming that $S_{ij} = t$. We also define function $\Pi_i(t) = \sum_{s=t}^{t+p_i-1} \pi_s$, $t \in [0, h - p_i]$. As the basis of the recursion, we use the fact that $R_{i,n(i)}(t) = \Pi_i(t)$, $t \in \{0, \ldots, h - p_i\}$. Then, the recursion formula is

$$R_{ij}(t) = \Pi_i(t) + \min_{t' \in \{t+p_i, \ldots, h-p_i\}} \{\delta_i(t' - t) + R_{i,j+1}(t')\}, \tag{40}$$

where $j \in \{0, \ldots, n(i) - 1\}$ and $t \in \{0, \ldots, h - p_i\}$.

The solution value of the problem (PS_i) is equal to $R_{i0}(S_{j0}) - \mu_i$. Using the recursion above, the problem (PS_i) can be solved in $O(n(i) \cdot h^2)$ time, which is not computationally tractable. This can be improved when using the structure of the penalty function δ. To do it, we define the following functions.

$$\begin{aligned} a_{ij}(t) &= \operatorname*{argmin}_{t' \in \{0, \ldots, t\}} \{R_{i,j+1}(t') - \alpha_i t'\}, \\ b_{ij}(t) &= \operatorname*{argmin}_{t' \in \{t, \ldots, h-p_i\}} \{R_{i,j+1}(t') + \beta_i t'\}. \end{aligned}$$

Using these functions, we can rewrite the formula (40). If $a_{ij}(t+l_i) \geq t+p_i$ then

$$R_{ij}(t) = \Pi_i(t) + \min_{t' \in \{a_{ij}(t+l_i), b_{ij}(t+l_i)\}} \{\delta_i(t' - t) + R_{i,j+1}(t')\}. \tag{41}$$

If $a_{ij}(t+l_i) < t+p_i$ then

$$R_{ij}(t) = \Pi_i(t) + \min_{t' \in \{t+p_i,\ldots,t+l_i-1\} \cup \{b_{ij}(t+l_i)\}} \{\delta_i(t'-t) + R_{i,j+1}(t')\}. \qquad (42)$$

Let us estimate the complexity of computing values $R_{ij}(t)$, $t \in [0, h-p_i]$, for a given j. Clearly, when $j = n(i)$, the computation of $R_{ij}(t)$ can be done in $O(h)$ time. Suppose that $j < n(i)$ and values $R_{i,j+1}(t)$, $t \in [0, h-p_i]$, are known. Then, values $a_{ij}(t)$ and $b_{ij}(t)$ can be also found in $O(h)$ time in the following way. We set $a_{ij}(0) = 0$, $b_{ij}(h-p_i) = h-p_i$, and then

$$a_{ij}(t) = \begin{cases} t, & \text{if } R_{i,j+1}(t) < R_{i,j+1}(a_{ij}(t-1)) + \\ & \qquad \alpha_i \cdot (t - a_{ij}(t-1)), \\ a_{ij}(t-1), & \text{otherwise,} \end{cases}$$

$$b_{ij}(t) = \begin{cases} t, & \text{if } R_{i,j+1}(t) < R_{i,j+1}(b_{ij}(t+1)) + \\ & \qquad \beta_i \cdot (b_{ij}(t+1) - t), \\ b_{ij}(t+1), & \text{otherwise.} \end{cases}$$

So, if one uses the recursion (41)-(42), the complexity of the dynamic programming algorithm becomes $O(n(i) \cdot h \cdot \max\{1, l_i - p_i\})$. In practice, the complexity of solving the problem (PS_i) is close to $O(n(i) \cdot h)$. Experiments have shown that the time needed to solve the pricing sub-problems for all jobs is negligible in comparison with the overall running time of the column generation algorithm.

6.2. Branching procedure

At a given iteration of the column generation procedure, no column with a negative reduced cost will be found after solving all the pricing sub-problems. In this case, the procedure stops and gives an optimal solution of the LP relaxation of the master problem (MP). Of course, this solution can be fractional. Therefore, the column generation approach should be complemented by a branching procedure.

One of the most important properties of a branch-and-price algorithm is that its branching procedure should not destroy the structure of the pricing problem. Otherwise, the algorithm for solving the pricing problem can not be applied beyond the root node of the search tree. For example, a usual way to branch on the variables of the master problem is not appropriate in our case. At a node of the search tree, where some variables are fixed to zero, applying

the dynamic programming algorithm presented above is not possible. The latter cannot take into account the additional constraints stating that some particular schedules should not be considered. Moreover, the branching on variables X would result in a highly unbalanced search tree.

Instead, we propose to branch on the domains of possible starting times of operations. Given a feasible solution X' of the LP relaxation of the formulation (MP), let Z'_{ijt} be the part of operation O_{ij} started at time t in X':

$$Z'_{ijt} = \sum_{\substack{k \in K_i, \\ S^k_{ij} = t}} X'_{ik}, \quad i \in N,\ j \in N(i),\ t \in H.$$

It is easy to see that Z' is integer if and only if X' is integer. If the solution X' is fractional (and therefore Z' is), we compute mean starting times for all the operations

$$S'_{ij} = \sum_{t \in H} t Z'_{ijt}, \quad i \in N,\ j \in N(i),$$

and choose an operation $O_{i'j'}$ with the maximum deviance $D'_{i'j'}$, where

$$D'_{ij} = \sum_{t \in H} Z'_{ijt}(t - S'_{ij})^2, \quad i \in N,\ j \in N(i).$$

The branching procedure will consist in creating two nodes. At the first one, we add the constraint $S_{i'j'} \leq \lfloor S'_{i'j'} \rfloor$, and at the second one, the constraint $S_{i'j'} \geq \lfloor S'_{i'j'} \rfloor + 1$ is added.

Note that the dynamic programming algorithm for the pricing sub-problems can easily take into account these additional constraints. If, at the current node of the search tree, the domain of possible starting times of operation O_{ij}, $i \in N$, $j \in N(i)$, is equal to $[\underline{S}_{ij}, \overline{S}_{ij}]$, then the recursion formula (40) becomes

$$R_{ij}(t) = \Pi_i(t) + \min_{t' \in \left\{\max\left(t+p_i, \underline{S}_{i,j+1}\right), \ldots, \overline{S}_{i,j+1}\right\}} \left\{\delta_i(t'-t) + R_{i,j+1}(t')\right\}.$$

The recursion (41)-(42) can be also modified appropriately.

6.3. Computation of an upper bound

To accelerate the search in the branch-and-price tree, at each node, a heuristic is executed to obtain an upper bound. The heuristic transforms

a fractional solution Z' of the column generation procedure to a feasible solution. For this, we use the notion of an $\alpha-$*point* [12]. An $\alpha-$point of operation O_{ij} is defined to be the first time $t(\alpha_{ij})$, $0 < \alpha_{ij} \leq 1$, in which the α_{ij} fraction of operation O_{ij} has been started, according to Z':

$$t(\alpha_{ij}) = \min \left\{ t \in H : \sum_{t'=0}^{t} Z'_{ijt} \geq \alpha_{ij} \right\}.$$

The heuristic sequence of operations is obtained by ordering them according to their $\alpha-$points. Then, the best schedule is found by solving a linear program in which the operations are processed according to the α-point sequence.

In our implementation, at each node of the search tree, 30 heuristic schedules are built, and the best one is chosen. The first 10 schedules S^k, $k = \{1, \ldots, 10\}$, are obtained using $\alpha_{ij} = 1/k$. To build a schedule from the last twenty, the values α_{ij}, are chosen randomly. The penalty of the best found schedule gives us an upper bound.

7. Computational tests

We have performed an experimental research to compare the practical efficiency of the formulations (TI), (TIA) and (MP). All tests have been performed on 1.8 GHz PC running Linux. We use Cplex 10.1 as an LP and MIP solver. Note that we used the sparse reformulation of (TI) (see [11] for details) and the aggregated version of (TIA) (see Appendix A).

In the experiments, we were interested in the efficiency of both the LP relaxations of the formulations and the formulations themselves. In the following, we will present results for the following statistics:

- Gap_{LP}, Gap_{LB}, Gap_{UB} — deviation of respectively the lower bound given be the LP relaxation, the best lower bound found and the best upper bound found from the best solution known (in per cent from the latter).
- Tm_{LP}, Tm — the time needed to solve, respectively, the LP relaxation and the formulation (if the resolution was interrupted before the termination, the time limit is given, preceded by ">").
- Nd — the number of nodes in the branch-and-bound or branch-and-price tree.

From the experiments on small instances (see Appendix B), it is clear that the formulation (TI) is much less efficient than the other two formulations. So, it was excluded from the following numerical experiments.

The first set of test instances was generated together with the engineering department of Thales (`www.thalesonline.com`). These instances represent real life situations arising while using an airborne radar. Note that, in reality, the length of a time unit is usually set to 1 millisecond. In our experiments, we increased this length to 10 milliseconds to decrease the size of the instances. However, in most cases, this increase does not have an impact on the set of optimal solutions (as the durations of dwells and frequencies are usually multiples of 10). The results for the second set of test instances are presented in Tables 2 and 3. Note that the first number in the names of the "bib" instances denotes jobs number, and the second one — total number of operations excluding the "zero" operations. For the "sc" instances, these numbers are shown in brackets.

In general, the efficiency of the algorithms when solving real life instances is satisfactory. The instances of the type "bib" seem to be easier to solve than instances of the type "sc". This can be explained by the fact that, for the "bib" instances, the LP relaxation lower bound is quite strong. For the "bib" instances which were not solved optimally, the gap between lower and upper bounds found is small. An exception is the instance "bib_20_171_i1". The reason seems to be its size, mainly the length of the time horizon.

The instances of the type "sc" are very hard for the branch-and-price algorithm. The column generation procedure on the top node could not even terminate in a reasonable time (therefore the results for these instances are not presented in Table 3). The reason is that such instances contain jobs representing research dwells (for these jobs we have $p_i = l_i$) with large number of operations. There are an enormous number of partial schedules (and columns) for these jobs with the same penalty value. Because of that, the column generation algorithm generates a lot of such columns before converging to an optimum.

The formulation (TIA) is much more successful on the "sc" instances. For all instances in this group, it managed to close the LP relaxation gap which is quite significant.

Finally, we tested the formulation (TIA) and the branch-and-price algorithm on random test instances (see Appendix C). Note that these instances are purely random and do not resemble the real life ones. The main purpose was to test algorithms on instances different from real ones.

The random instances happened to be harder to solve than the real life instances. Another conclusion of this test is that the two approaches proposed are somewhat complementary to each other. The formulation (TIA) is better suited for solving instances with small number of jobs, whereas the branch-and-price algorithm is more efficient when applied to instances for which the relation between the total number of operations and the number of jobs is small.

8. Conclusions and perspectives

In this paper we have introduced a generalization for the classic earliness-tardiness single machine scheduling problem. The motivation for this generalization came from the airborne radars scheduling. Given the theoretical and practical complexity of the problem and a shortage of time we possess to find a solution, heuristics should be used in practice. In this work, for the first time, we have proposed methods to find good lower bounds and optimal solutions. Such methods are needed to estimate the quality of heuristics.

Among three methods considered, the theoretical and experimental research showed the superiority of the branch-and-price algorithm and the formulation (TIA) which can be solved by a MIP solver.

The approaches we proposed are based on time indexed MIP formulations which are known to provide tight lower bounds for some scheduling problems when integrality constraints are relaxed [1, 11]. Unfortunately, this is not the case for the problem we consider. The results of the experimental research on a set of random instances showed that usually a significant gap between lower and upper bound remains even after 30 minutes of the solution time. Nevertheless, we could solve to optimality many real life instances of a moderate size. Real life instances seem to be easier to solve than randomly generated instances, mainly because the integrality gap is usually not large for them.

The solution methods presented seem to be directly modifiable for the multi-machine variant (when machines or radars are identical) and for the cyclic variant of the problem (with a fixed cycle length h). Note that Theorem 2 holds for the cyclic variant. Different objective functions which make sense in practice can be also considered.

In order to increase the efficiency of the methods proposed, one may think of applying a Lagrangean relaxation. For example, in [14], such an approach was quite efficient in the context of a time indexed formulation.

Acknowledgments

The authors would like to thank Emilie Winter from Thales for the collaboration and supplying us with real life instances. The authors are also grateful to anonymous referees who helped to improve the quality of the paper.

References

[1] J.M. van den Akker, C.A.J. Hurkens, M.W.P. Savelsbergh (1999), A polyhedral approach to single-machine scheduling problems, *Mathematical Programming* 85:541–572.

[2] J.M. van den Akker, C.A.J. Hurkens, M.W.P. Savelsbergh (2000), Time-indexed formulations for machine scheduling problems: column generation, *INFORMS Journal on Computing* 12:111–124.

[3] F. Barbaresco (2003), Approche Cognitive de la Gestion Radar, *Proceedings of the conference Cogis 2003.*

[4] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance (1998), Branch-and-price: Column generation for solving huge integer programs. *Operations Research* 44(6):852–863.

[5] D. Chen, A.K. Mok (2004), The pinwheel: A real-time scheduling problem, in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, J.Y.-T. Leung (ed.), Chapman Hall, CRC Press.

[6] J.S. Davis, J.J. Kanet (1993), Single-machine scheduling with early and tardy completion costs, *Naval Research Logistics* 40: 85–101.

[7] M. Elshafei, H. D. Sherali, J. C. Smith (2003), Radar pulse interleaving for multi-target tracking, *Naval Research Logistics*, 51(1):72–94.

[8] M.R. Garey, R.E. Tarjan, G.T. Wilfong (1988), One-processor scheduling with symmetric earliness and tardiness penalties, *Mathematics of Operations Research* 13:330–348.

[9] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G Rinnooy Kan (1979), Optimization and approximation in deterministic sequencing and scheduling: a survey, *Ann. Discrete Math.* 5:287–326.

[10] C.C. Han, K.J. Lin (1992), Scheduling Distance-Constrained Real-time Tasks, *IEEE Real-Time Systems Symposium*, December 1992, 814–826.

[11] Y. Pan, L. Shi (2006), On the equivalence of the max-min transportation lower bound and the time-indexed lower bound for single-machine scheduling problems, *Mathematical Programming* 110(3):543–559.

[12] M.W.P. Savelsbergh, R.N. Uma, J. Wein (2005), An experimental study of LP-based approximation algorithms for scheduling problems, *INFORMS Journal on Computing* 17:123–136.

[13] M. I. Skolnik (1979), *Introduction to Radar Systems*, 2^{nd} edition, *McGraw-Hill Kogakusha*.

[14] F. Sourd, S. Kedad-Sidhoum (2008), A faster branch-and-bound algorithm for the earliness-tardiness scheduling problem, *Journal of Scheduling* 11:49–58.

[15] J.P. Sousa, L.A. Wolsey (1992), A time-indexed formulation of non-preemptive single-machine scheduling problems, *Mathematical Programming* 54:353-367.

[16] G. W. Stimson (1998), *Introduction to Airborne Radar*, 2^{nd} Edition. SciTech Publishing, Inc.

[17] W. Wei, C. Liu (1983), On a Periodic Maintenance Problem. *Operations Research Letters* 2(2):90–93.

[18] E. Winter, Ph. Baptiste (2007), On Scheduling a Multifunction Radar, *Aerospace Science and Technologies* 11(4):289–294.

Test	h	q	Gap_{LP}	Tm_{LP}	Gap_{LB}	Gap_{UB}	Tm	Nd
sc_2_01_4 (10,87)	600	4	41.1%	185s	0%	**0%**	93m	448
sc_2_02_4 (10,75)	600	3	36.4%	26s	0%	**0%**	37m	2038
sc_4_01_4 (11,77)	450	4	37.6%	30s	0%	**0%**	15m	545
sc_4_02_4 (11,58)	450	4	31.4%	14s	0%	**0%**	8m	568
sc_5_01_4 (11,91)	600	6	83.0%	11m	0%	**0%**	212m	1087
sc_5_02_4 (11,80)	600	5	59.3%	3m	0%	**0%**	54m	215
bib_9_57_i2	500	2	0%	1s	0%	**0%**	1s	0
bib_14_62_i2	500	2	0%	3s	0%	**0%**	3s	0
bib_14_83_i2	500	3	0%	21s	0%	**0%**	21s	0
bib_15_91_i2	500	5	7.9%	44s	0%	**0%**	79m	406
bib_15_91_i4	500	5	0.1%	24s	0%	**0%**	200s	18
bib_15_91_i8	500	5	6.3%	88s	0%	**0%**	507s	265
bib_15_93_i2	500	5	6.1%	48s	4.4%	1.4%	>2h	1115
bib_15_93_i4	500	5	<0.1%	47s	0%	**0%**	101s	5
bib_15_93_i8	500	5	2.3%	102s	0%	**0%**	364s	65
bib_15_96_i2	500	5	10.7%	77s	9.9%	1.1%	>2h	338
bib_15_96_i4	500	5	<0.1%	56s	0%	**0%**	64s	0
bib_15_96_i8	500	5	4.2%	81s	0%	**0%**	857s	314
bib_18_72_i4	500	4	0.6%	7s	0%	**0%**	550s	169
bib_18_74_i8	500	4	9.0%	89s	6.4%	0.3%	>2h	2811
bib_18_76_i2	500	4	0.4%	50s	0%	**0%**	158s	10
bib_15_151_i4	900	5	3.8%	129s	0%	**0%**	321s	14
bib_15_151_i10	900	5	22.7%	390s	0%	**0%**	6367s	571
bib_20_171_i1	1350	6	47.2%	5960s	47.2%	-	>2h	1

Table 2: Results for real life instances — the formulation (TIA)

Test	h	Gap_{LP}	Tm_{LP}	Gap_{LB}	Gap_{UB}	Tm	Nd
bib_9_57_i2	500	0%	2s	0%	**0%**	2s	0
bib_14_62_i2	500	0%	1s	0%	**0%**	1s	0
bib_14_83_i2	500	0%	52s	0%	**0%**	52s	0
bib_15_91_i2	500	0%	4s	0%	**0%**	24m	513
bib_15_91_i4	500	0%	2s	0%	**0%**	2s	0
bib_15_91_i8	500	0.9%	2s	0%	**0%**	445s	355
bib_15_93_i2	500	3.3%	5s	0.9%	0%	>5h	4076
bib_15_93_i4	500	0%	5s	0%	**0%**	5s	0
bib_15_93_i8	500	1.0%	4s	0%	**0%**	638s	131
bib_15_96_i2	500	9.0%	29s	2.8%	0%	>5h	338
bib_15_96_i4	500	0%	96s	0%	**0%**	96s	0
bib_15_96_i8	500	1.7%	11s	0%	**0%**	6322s	17
bib_18_72_i4	500	0.6%	7s	0%	**0%**	3821s	2855
bib_18_74_i8	500	7.4%	4s	0%	**0%**	4617s	1143
bib_18_76_i2	500	0.4%	12s	0%	**0%**	13082s	879
bib_15_151_i4	900	0%	16s	0%	**0%**	16s	0
bib_15_151_i10	900	2.4%	343s	2.0%	0%	>5h	4
bib_20_171_i1	1350	43.0%	10515s	43.0%	0%	>5h	2

Table 3: Results for real life instances — branch-and-price algorithm