



HAL
open science

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

Philippe Baptiste, Ruslan Sadykov

► **To cite this version:**

Philippe Baptiste, Ruslan Sadykov. Time Indexed Formulations for Scheduling Chains on a Single Machine: An Application to Airborne Radars. *European Journal of Operational Research*, 2009, 10.1016/j.ejor.2009.07.037 . inria-00339639v1

HAL Id: inria-00339639

<https://inria.hal.science/inria-00339639v1>

Submitted on 18 Nov 2008 (v1), last revised 2 Sep 2009 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

3 Philippe Baptiste^{a,*}, Ruslan Sadykov^{b,c}

4 ^a*Ecole Polytechnique, CNRS LIX UMR 7161, 91128 Palaiseau, France*

5 ^b*INRIA Bordeaux — Sud-Ouest, France*

6 ^c*Institut de Mathématique de Bordeaux, 351 cours de la Liberation, 33405 Talence,*
7 *France*

8 **Abstract**

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

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

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

27 *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: Philippe.Baptiste@polytechnique.fr (Philippe Baptiste),
Ruslan.Sadykov@inria.fr (Ruslan Sadykov)

Preprint submitted to Elsevier

October 31, 2008

28 1. Airborne radars scheduling

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

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

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

48 The following tasks have to be performed by an airborne radar (see Fig-
49 ure 1).

- 50 • *Research.* The radar is sweeping across a domain to detect the potential
51 presence of targets inside.
- 52 • *Tracking.* The radar is closely monitoring the behavior of targets (ini-
53 tially detected at the Research stage).
- 54 • *Data Link.* The radar is used as a communication tool with other
55 platforms.
- 56 • *Calibration.* The radar is performing cyclic calibration to ensure a high
57 reliability level.

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

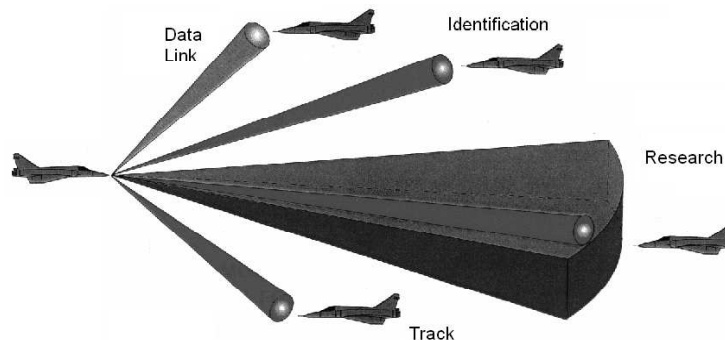


Figure 1: Functions of the ESA radars

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

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

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

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

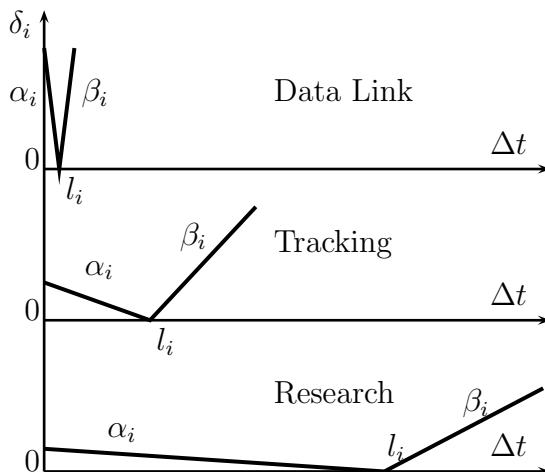


Figure 2: Penalty Functions

91 *Formal definition of the problem*

92 A set $N = \{1, \dots, n\}$ of jobs (which represent radar tasks) has to be pro-
93 cessed on a single machine. Each job $i \in N$ consists of a chain $(O_{i0}, O_{i1}, \dots, O_{i,n(i)})$
94 of $n(i) + 1$ identical operations (representing dwells) which should be pro-
95 cessed in the same order within a given time horizon $[0, h]$. We denote by
96 $N(i)$, $i \in N$, the set $\{1, \dots, n(i)\}$ and by H the set $\{0, 1, \dots, h - 1\}$. Pro-
97 cessing an operation cannot be interrupted, and the machine can process at
98 most one operation at a time. The processing time of each operation of job
99 i is equal to $p_i > 0$. The starting time of operation O_{ij} in schedule S will be
100 denoted as S_{ij} . For each job $i \in N$, the starting time S_{i0} of operation O_{i0} is
101 given and cannot be changed.

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

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

107 Examples of penalty functions used for different radar tasks are presented in
 108 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
 109 finding a feasible schedule S which minimizes the total penalty

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

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

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

120 *Cyclic variant of the problem*

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

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

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

136 *Structure of the paper*

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

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

148 2. Time indexed formulation

149 First, we formulate the problem as a Integer Linear Programming (ILP)
150 problem in a straightforward way. We use the standard time indexed for-
151 mulation for non-preemptive single-machine scheduling problems [15]. The
152 binary variable X_{ijt} , $i \in N$, $j \in N(i)$, $t \in H$, takes value 1 if and only
153 if operation O_{ij} is started at time t . The continuous variable S_{ij} , $i \in N$,
154 $j \in N(i)$, equals the starting time of operation O_{ij} . The values of variables
155 S_{i0} , $i \in N$, are fixed. Finally, the continuous variable W_{ij} , $i \in N$, $j \in N(i)$,
156 represents the value $\delta_i(S_{ij} - S_{i,j-1})$. We now present the time indexed ILP
157 formulation.

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

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

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

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

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

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

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

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

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

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

172 3. Alternative time-indexed formulation for scheduling chains on 173 a single machine

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

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

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

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

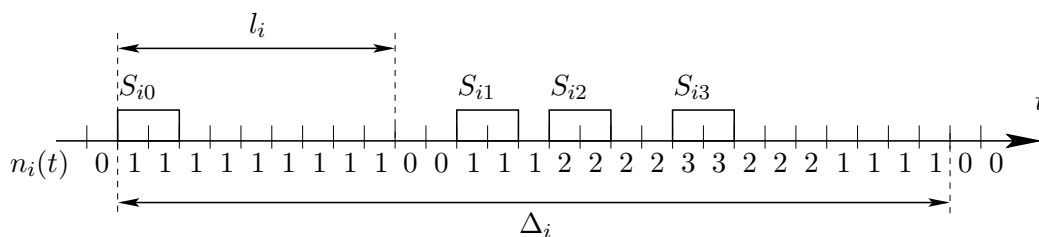


Figure 3: A schedule S for job i ($n(i) = 3$, $p_i = 2$, $l_i = 9$)

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

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

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

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

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

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

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

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

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

194 Consider such a block of consecutive operations. We define $A'_i(j', j'') =$
 195 $\{S_{ij'}, \dots, S_{ij''} + l_i - 1\}$ and, if $j' \neq 0$, $B'_i(j', j'') = \{S_{i,j'-1} + l_i, \dots, S_{ij'} - 1\}$.
 196 Note that $A'_i(j', j'') \subseteq A_i$ and $B'_i(j', j'') \subseteq B_i$. Clearly, $|B'_i(j', j'')| = D_{ij'} - l_i =$
 197 $T_{ij'}$. Summing this equality over all blocks except the first one, we obtain
 198 $|B_i| = \sum_{j \in N(i)} T_{ij}$.

199 Then, each operation in the block is counted l_i times in the sum
 200 $\sum_{t \in A'_i(j', j'')} n_i(t)$. Therefore,

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

201 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

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

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

204 The integer variables X_{it} , $i \in N$, $t \in H$, are now the number of the
 205 operations of job i started before or at time t . Continuous variables W_{it} ,
 206 $i \in N$, $t \in H_i$, represent the value $\gamma_i(t)$. The main issue with the alternative
 207 objective function is that the starting times $S_{i,n(i)}$, $i \in N$, are not known a
 208 priori. Therefore, we need to use *additional binary variables* E_{it} , $i \in N$, $t \in$
 209 H_i , *which indicate whether* $t \in \Delta_i$. We are ready to present the alternative
 210 time indexed ILP formulation.

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

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

$$X_{i,t-1} \leq X_{it}, \quad \forall i \in N, t \in H \setminus \{0\}, \quad (15)$$

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

$$\sum_{i \in N} X_{it} - \sum_{\substack{i \in N, \\ t-p_i \geq 0}} X_{i,t-p_i} \leq 1, \quad \forall t \in H, \quad (17)$$

$$E_{it} \geq X_{i,t-l_i+p_i} - X_{i,t-l_i}, \quad \forall i \in N, t \in H_i, t \geq S_{i0} + l_i, \quad (18)$$

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

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

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

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

211 The constraints (15) reflect the nature of the variables X . The constraints
 212 (16) state that exactly $n(i) + 1$ operations of job i should be processed within
 213 interval $[S_{i0}, h]$. The constraints (17) prevent overlapping. The constraints
 214 (18)-(19) are used to compute the values for the variables E . Once variables
 215 X_{it} is equal to 1, variables $E_{it'}$, $S_{i0} \leq t' \leq t + l_i - 1$, are forced to be equal to
 216 1 by these constraints. Note that we do not need to impose the integrality
 217 on the variables E . Whenever the variables X are integer, the variables E
 218 are also integer. The constraints (20)-(21) are used to compute the values of
 219 the variables W .

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

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

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

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

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

232 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
 233 Theorem 1, we have $E_i = \sum_{t \in \Delta_i} \max\{n_i(t) - 1, 0\}$. The LP relaxations of
 234 the formulations (TI) and (TIA) can be rewritten to respectively (SC) and
 235 (SCA) by putting the constant part $\beta_i (T_i - E_i)$ for every job i directly to the
 236 objective function:

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

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

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

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

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

237

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

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

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

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

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

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

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

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

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

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

246 Basically, every positive value $(X_{it}^* - X_{i,t-1}^*)$, $t \in H$, is “distributed” among
 247 variables X'_{ijst} , $j \in N(i)$, in such a way that $X'_{ijs} > 0$ and $X'_{i,j+1,t} > 0$ only
 248 if $s \leq t$. It is easy to see that $X'_{ijt} \in [0, 1]^{n(i) \times h}$, and the constraints (3) are
 249 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^*
 250 satisfies (17), X'_i satisfies the constraints (4). Then, for each $j \in N(i)$ and
 251 $t \in H$, we have

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

252 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
 253 constraints (17) would be violated. Given $j \in N(i)$, we sum up (31) for all
 254 $t \in [1, h]$ and obtain

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

255 X'_i satisfies then constraints (6).

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

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

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

261 Consider a block $k \in M$ containing operations $(O_{i,j'_k}, O_{i,j'_k+1}, \dots, O_{i,j''_k})$,
 262 where the number of operations is at least two ($j'_k < j''_k$). Let t_k^1 be the
 263 last time such that $X'_{ij'_k t} > 0$ and t_k^2 be the first time such that $X'_{ij''_k t} > 0$.
 264 We denote $\Delta_{ik} = \{t_k^1, \dots, t_k^2 + l_i - 1\}$. Note that $t_k^2 + l_i \leq t_{k+1}^1$, otherwise
 265 $S'_{ij''_k} \geq t_k^2 > t_{k+1}^1 - l_i \geq S'_{ij'_{k+1}} - l_i$ which would contradict the fact that
 266 operation $O_{ij'_{k+1}}$ is tardy. Therefore,

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

267 We denote

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

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

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

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

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

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 $\nu_{TI}^* \leq \nu_{TIA}^*$

272 Finally,

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

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

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

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

282 5. Dantzing-Wolfe reformulation

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

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

291 In the following, S_i^k , $k \in K_i$, is the k -th possible schedule $(S_{i0}^k, S_{i0}^k, \dots, S_{i,n(i)}^k)$
 292 for job $i \in N$, and w_i^k be the corresponding penalty function. Let also
 293 $\chi(i, k, t)$ be a characteristic function which indicates whether schedule S_i^k is
 294 “active” at time t :

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

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

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

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

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

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

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

301 6. Branch-and-Price algorithm

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

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

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

317 *6.1. Resolution of the pricing sub-problem*

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

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

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

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

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

326 As the sequence of the operations is fixed, the problem (PS_i) can be solved
 327 by a dynamic programming approach. Let $R_{ij}(t)$ denote the minimal reduced
 328 cost of a schedule of operations $O_{ij}, \dots, O_{i,n(i)}$ assuming that $S_{ij} = t$. We
 329 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
 330 recursion, we use the fact that $R_{i,n(i)}(t) = \Pi_i(t)$, $t \in \{0, \dots, h - p_i\}$. Then,
 331 the recursion formula is

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

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

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

$$a_{ij}(t) = \operatorname{argmin}_{t' \in \{0, \dots, t\}} \{R_{i,j+1}(t') - \alpha_i t'\},$$

$$b_{ij}(t) = \operatorname{argmin}_{t' \in \{t, \dots, h-p_i\}} \{R_{i,j+1}(t') + \beta_i t'\}.$$

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

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

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

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

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

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

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

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

352 6.2. Branching procedure

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

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

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

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

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

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

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

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

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

377 The branching procedure will consist in creating two nodes. At the first one,
 378 we add the constraint $S_{i'j'} \leq \lfloor S'_{i'j'} \rfloor$, and at the second one, the constraint
 379 $S_{i'j'} \geq \lfloor S'_{i'j'} \rfloor + 1$ is added.

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

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

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

386 6.3. Computation of an upper bound

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

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

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

393 The heuristic sequence of operations is obtained by ordering them accord-
 394 ing to their α -points. Then, the best schedule is found by solving a linear
 395 program in which the operations are processed according to the α -point se-
 396 quence.

397 In our implementation, at each node of the search tree, 30 heuristic
 398 schedules are built, and the best one is chosen. The first 10 schedules S^k ,
 399 $k = \{1, \dots, 10\}$, are obtained using $\alpha_{ij} = 1/k$. To build a schedule from the
 400 last twenty, the values α_{ij} , are chosen randomly. The penalty of the best
 401 found schedule gives us an upper bound.

402 7. Computational tests

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

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

- 411 • Gap_{LP} , Gap_{LB} , Gap_{UB} — deviation of respectively the lower bound
 412 given by the LP relaxation, the best lower bound found and the best
 413 upper bound found from the best solution known (in per cent from the
 414 latter).
- 415 • Tm_{LP} , Tm — the time needed to solve, respectively, the LP relax-
 416 ation and the formulation (if the resolution was interrupted before the
 417 termination, the time limit is given, preceded by “>”).
- 418 • Nd — the number of nodes in the branch-and-bound or branch-and-
 419 price tree.

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

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

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

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

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

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

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

465 8. Conclusions and perspectives

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

473 Among three methods considered, the theoretical and experimental re-
474 search showed the superiority of the branch-and-price algorithm and the for-
475 mulation (TIA) which can be solved by a MIP solver.

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

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

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

494 **Acknowledgments**

495 The authors would like to thank Emilie Winter from Thales for the col-
496 laboration and supplying us with real life instances. The authors are also
497 grateful to anonymous referees who helped to improve the quality of the
498 paper.

499 **References**

- 500 [1] J.M. van den Akker, C.A.J. Hurkens, M.W.P. Savelsbergh (1999), A
501 polyhedral approach to single-machine scheduling problems, *Mathemat-*
502 *ical Programming* 85:541–572.
- 503 [2] J.M. van den Akker, C.A.J. Hurkens, M.W.P. Savelsbergh (2000), Time-
504 indexed formulations for machine scheduling problems: column genera-
505 tion, *INFORMS Journal on Computing* 12:111–124.
- 506 [3] F. Barbaresco (2003), Approche Cognitive de la Gestion Radar, *Pro-*
507 *ceedings of the conference Cogis 2003*.
- 508 [4] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and
509 P.H. Vance (1998), Branch-and-price: Column generation for solving
510 huge integer programs. *Operations Research* 44(6):852–863.
- 511 [5] D. Chen, A.K. Mok (2004), The pinwheel: A real-time scheduling prob-
512 lem, in *Handbook of Scheduling: Algorithms, Models, and Performance*
513 *Analysis*, J.Y.-T. Leung (ed.), Chapman Hall, CRC Press.
- 514 [6] J.S. Davis, J.J. Kanet (1993), Single-machine scheduling with early and
515 tardy completion costs, *Naval Research Logistics* 40: 85–101.
- 516 [7] M. Elshafei, H. D. Sherali, J. C. Smith (2003), Radar pulse interleaving
517 for multi-target tracking, *Naval Research Logistics*, 51(1):72–94.
- 518 [8] M.R. Garey, R.E. Tarjan, G.T. Wilfong (1988), One-processor schedul-
519 ing with symmetric earliness and tardiness penalties, *Mathematics of*
520 *Operations Research* 13:330–348.
- 521 [9] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G Rinnooy Kan (1979),
522 Optimization and approximation in deterministic sequencing and
523 scheduling: a survey, *Ann. Discrete Math.* 5:287–326.

- 524 [10] C.C. Han, K.J. Lin (1992), Scheduling Distance-Constrained Real-time
525 Tasks, *IEEE Real-Time Systems Symposium*, December 1992, 814–826.
- 526 [11] Y. Pan, L. Shi (2006), On the equivalence of the max-min transporta-
527 tion lower bound and the time-indexed lower bound for single-machine
528 scheduling problems, *Mathematical Programming* 110(3):543–559.
- 529 [12] M.W.P. Savelsbergh, R.N. Uma, J. Wein (2005), An experimental study
530 of LP-based approximation algorithms for scheduling problems, *IN-
531 FORMS Journal on Computing* 17:123–136.
- 532 [13] M. I. Skolnik (1979), *Introduction to Radar Systems*, 2nd edition,
533 *McGraw-Hill Kogakusha*.
- 534 [14] F. Sourd, S. Kedad-Sidhoum (2008), A faster branch-and-bound algo-
535 rithm for the earliness-tardiness scheduling problem, *Journal of Schedul-
536 ing* 11:49–58.
- 537 [15] J.P. Sousa, L.A. Wolsey (1992), A time-indexed formulation of non-
538 preemptive single-machine scheduling problems, *Mathematical Program-
539 ming* 54:353–367.
- 540 [16] G. W. Stimson (1998), *Introduction to Airborne Radar*, 2nd Edition.
541 SciTech Publishing, Inc.
- 542 [17] W. Wei, C. Liu (1983), On a Periodic Maintenance Problem. *Operations
543 Research Letters* 2(2):90–93.
- 544 [18] E. Winter, Ph. Baptiste (2007), On Scheduling a Multifunction Radar,
545 *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