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† Ruslan Sadykov‡

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 MIP solver. The third formulation relies on a Branch-and-Price algorithm. Theoretical and experimental comparisons of the formulations are reported.

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

†Ecole Polytechnique, CNRS LIX UMR 7161, 91128 Palaiseau, France. E-mail: Phlippe.Baptiste@polytechnique.fr

‡INRIA Bordeaux — Sud-Ouest, 351 cours de la Liberation, 33405 Talence, France. E-mail: Ruslan.Sadykov@inria.fr

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 [11] and [14] 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

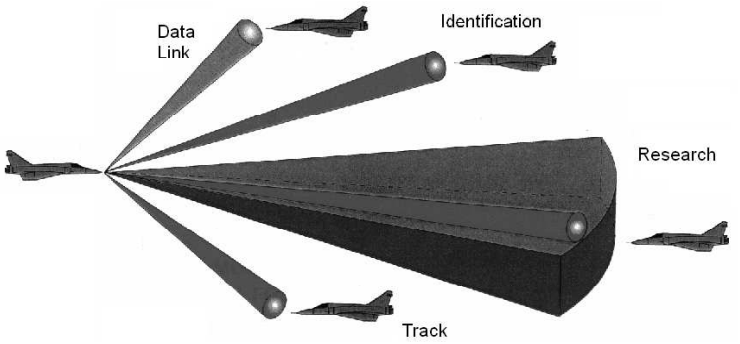

Figure 1: Functions of the ESA radars

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 loose 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. Thus, the problem is how to use the radar time resources to keep at best the whole situation awareness. 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 [2], Winter and Baptiste [16] 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 proportionally 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 [16] 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.

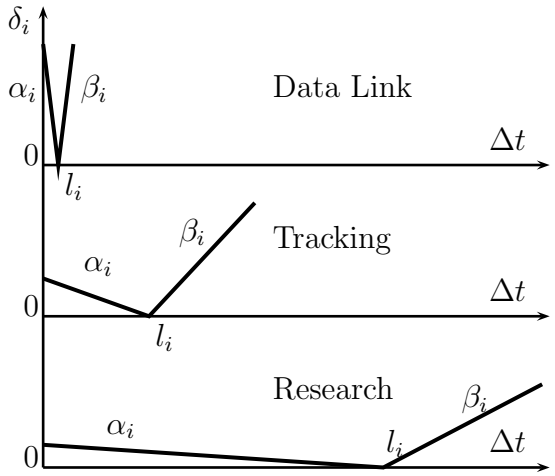

Figure 2: Penalty Functions

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 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 of an operation cannot be interrupted, and the machine can process only 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} 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 distance 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 distance 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 $\{S_{ij}\}_{i\in N, j\in N(i)}$ which minimizes the total penalty

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

We assume that all data except $\{\alpha_i, \beta_i\}_{i\in N}$ are integers. If this assumption does not hold, 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 a certain value (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 classic earliness-tardiness problem on a single machine [4] denoted as $1||\sum \alpha_j E_j + \beta_j T_j$ in the standard scheduling notation [7]. This problem is NP-hard in the strong sense [6].

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 [8] (DCTS stands for *Distance-Constraint Task System*) and the *Periodic Maintenance Problem* [15] (PMP).

If we use our notations, 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* [3].

Structure of the paper

In Section 2 we present a basic time-indexed formulation of the problem [13]. 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 much less variables. We prove that, under some reasonable assumption, the LP relaxation of this alternative formulation always provides tighter bounds than the initial LP relaxation. In the end of the Section 3, a variant of the alternative formulation is presented in which identical jobs can be aggregated into groups. This aggregation reduces further the size of the formulation. Section 4 is devoted to a non-compact formulation of the problem. We show how to solve the LP relaxation of this formulation by column generation procedure, and we describe how the latter can be combined with branching resulting in a branch-and-price algorithm for the problem. Experiments have been executed on real life and random instances (Section 5). Conclusions are drawn in Section 6.

2 Basic 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 [13]. 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 moment 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.

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

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

(TI)

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

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

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

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

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

$$X_{ijt'} \in \{0,1\}, \quad \forall i \in N,\ j \in N(i)\cup\{0\},\ t \in H. \tag{9}$$

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 only one operation is processed at a time). The constraints (5) link the variables X and S. The constraints (6) guarantee that operations of a job are processed in the given order. The constraints (7)-(8) are used to compute the values of the variables W.

In order to make the formulation (TI) more sparse, we introduce the binary variables Y. Y_t, $t \in H$, takes value 1 if and only if the machine is idle (does not process any operation) at time moment t. Then the constraints (4) can be rewritten as

$$\sum_{i\in N}\sum_{j\in N(i)\cup\{0\}} X_{ij0} + Y_0 = 1, \tag{10}$$

$$\sum_{i\in N}\sum_{j\in N(i)\cup\{0\}} X_{ijt} - \sum_{\substack{i\in N,\\ t-p_i\ge 0}}\ \sum_{j\in N(i)\cup\{0\}} X_{i,j,t-p_i} + Y_t - Y_{t-1} = 0,$$

$$\forall t \in H\setminus\{0\} \tag{11}$$

See [9] for details of such transformation. Note that, in each of the constraints (10)-(11), there are at most $\sum_{i\in N} 2n(i) + 2$ non-zero coefficients, whereas, in each of the constraints (4), there are up to $\sum_{i\in N} p_i \cdot n(i)$ non-zero coef-

ficients. We denote as (TIS) the formulation (TI) where the constraints (4) are replaces by the constraints (10)-(11).

3 New formulation

One of disadvantages of the (TI) formulation is that we use distinct variables for operations of the same job. As all operations of a job are identical, it would be more natural and more efficient to use the same set of variables for them. For example, we can introduce binary variables X_{it}, $i \in N$, $t \in H$, that state that some 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).

3.1 Reformulation of the problem

We now reformulate the problem. Consider a given schedule and let $n_i(t)$, $i \in N$, $t \in H$, be the number of operations of job i started within interval $[t - l_i + 1, \ldots, t]$ (see example in Figure 3). We define now 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 now introduce an alternative penalty function γ_i:

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

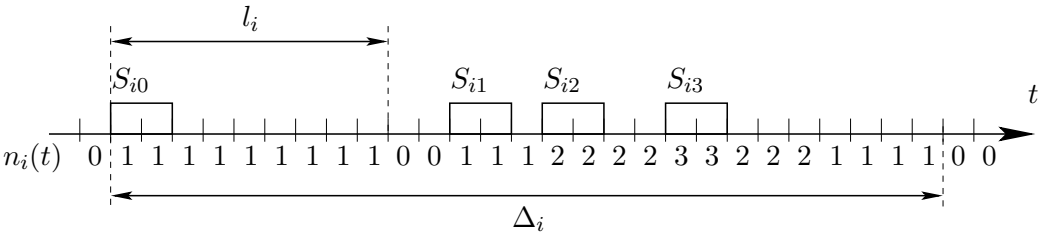

Figure 3: A schedule for job i ($n(i) = 3$, $p_i = 2$, $l_i = 9$)

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

Theorem 1 *For a given schedule* $(S_{i0}, S_{i1}, \ldots, S_{i,n(i)})$ *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 $B_i \cup A_i = \Delta_i$. We have now

$$\sum_{t \in \Delta_i} \gamma_i(t) = \alpha_i \cdot \sum_{t \in A_i} (n_i(t) - 1) + \beta_i \cdot |B_i|. \tag{12}$$

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

$$\delta_i(D_{ij}) = \alpha_i \cdot \underbrace{\max\{l_i - D_{ij}, 0\}}_{=E_{ij}} + \beta_i \cdot \underbrace{\max\{D_{ij} - l_i, 0\}}_{=T_{ij}}, \tag{13}$$

where E_{ij} is the relative earliness of operation O_{ij} and T_{ij} is its relative tardiness. From (12) and (13) 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 now 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 either is late ($T_{ij'} > 0$ and $E_{ij'} = 0$) or is 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 ($E_{ij} \geq 0$ and $T_{ij} = 0$).

Consider such a block of consecutive operations. We define $A' = \{S_{ij'}, \ldots, S_{ij''} + l_i - 1\}$ and, if $j' \neq 0$, $B' = \{S_{i,j'-1} + l_i, \ldots, S_{ij'} - 1\}$. Note that $A' \subseteq A_i$ and $B' \subseteq B_i$. Clearly, $|B'| = D_{ij'} - l_i = T_{ij'}$. 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 $\sum_{t \in A'} n_i(t)$. Therefore,

$$\begin{aligned} \sum_{t \in A'} n_i(t) - |A'| &= l_i(j'' - j' + 1) - (S_{ij''} + l_i - S_{ij'}) \\ &= \sum_{j=j'+1}^{j''} (l_i - S_{ij} + S_{i,j-1}) = \sum_{j=j'+1}^{j''} E_{ij}. \end{aligned} \tag{14}$$

Summing (14) over all blocks, we obtain $\sum_{t \in A_i} n_i(t) - |A_i| = \sum_{j \in N(i)} E_{ij}$. □

3.2 Alternative time-indexed formulation

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

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

Integer variable X_{it}, $i \in N$, $t \in H$, is now the number of operations of job i started before or at time moment t. Continuous variable W_{it}, $i \in N$, $t \in H_i$, represents 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 now present the alternative time-indexed ILP formulation.

(TIA)

$$\min \quad \sum_{i \in N} \sum_{t \in H_i} W_{it} \tag{15}$$

$$s.t. \quad X_{i,S_{i0}} = 1, \quad X_{i,S_{i0}-1} = 0, \quad \forall i \in N. \tag{16}$$

$$X_{i,t-1} \leq X_{it}, \quad \forall i \in N,\ t \in H \setminus \{0\}, \tag{17}$$

$$X_{i,h-p_i} = n(i) + 1, \quad \forall i \in N, \tag{18}$$

$$\sum_{i \in N} X_{it} - \sum_{\substack{i \in N, \\ t - p_i \geq 0}} X_{i,t-p_i} \leq 1, \quad \forall t \in H, \tag{19}$$

$$E_{it} \geq X_{i,t-l_i+1},\ \forall i \in N,\ t \in H_i,\ t \geq S_{i0} + l_i, \tag{20}$$

$$E_{it} \leq E_{i,t-1} \leq 1, \quad \forall i \in N,\ t \in H_i \setminus \{S_{i0}\}, \tag{21}$$

$$W_{it} \geq \alpha_i(X_{it} - X_{i,t-l_i} - E_{it}), \quad \forall i \in N,\ t \in H_i, \tag{22}$$

$$W_{it} \geq \beta_i(E_{it} - X_{it} + X_{i,t-l_i}), \quad \forall i \in N,\ t \in H_i, \tag{23}$$

$$X_{it} \in \mathbb{Z}_+, \quad \forall i \in N,\ t \in H. \tag{24}$$

The constraints (17) reflect the nature of the variables X. The constraints (18) state that exactly $n(i)+1$ operations of job i should be processed within interval $[S_{i0}, h]$. The constraints (19) prevent overlapping. The constraints (20)-(21) are used to compute the values for the variables E. Once variables X_{it} is equal to 1, variables $E_{it'}$, $S_{i0} \leq t' \leq t + l_i - 1$, 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 (22)-(23) are used to compute the values of the variables W.

3.3 Theoretical comparison

Here we show that, under some reasonable assumption, 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 now be rewritten:

$$\min \sum_{i \in N} \sum_{j \in N(i)} W_{ij} + \sum_{i \in N} \beta_i \cdot \big(S_{i,n(i)} - S_{i0} - l_i n(i)\big) \tag{25}$$

(SC) $s.t.$ (3) − (5),

$$W_{ij} \geq (\alpha_i + \beta_i) \cdot (l_i - S_{ij} + S_{i,j-1}), \quad \forall i \in N,\ j \in N(i), \tag{26}$$

$$W_{ij} \geq 0, \quad \forall i \in N,\ j \in N(i). \tag{27}$$

$$0 \leq X_{ijt} \leq 1, \quad \forall i \in N,\ j \in N(i), t \in H. \tag{28}$$

$$\min \sum_{i \in N} \sum_{t \in \Delta_i} W_{it} + \sum_{i \in N} \beta_i \cdot \big(S_{i,n(i)} - S_{i0} - l_i n(i)\big) \tag{29}$$

(SCA) $s.t.$ (16) − (19),

$$X_{i,S_{i,n(i)}} = n(i) + 1, \quad X_{i,S_{i,n(i)}-1} = n(i), \quad \forall i \in N. \tag{30}$$

$$W_{it} \geq (\alpha_i + \beta_i) \cdot (X_{it} - X_{i,t-l_i} - 1), \quad \forall i \in N,\ t \in \Delta_i, \tag{31}$$

$$W_{it} \geq 0, \quad \forall 0 \leq X_{it} \leq n(i) + 1, \quad i \in N,\ t \in H. \tag{32}$$

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

Theorem 2 *If the starting times of the last operations of jobs 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 all $i \in N$, given X_i^*, we build X_i' in the following way:

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

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 (19), X_i' satisfies the constraints (4). Then, for each $j \in N(i)$ and $t \in H$, we have

$$\sum_{t'=t}^{h-1} X'_{i,j-1,t'} \leq \sum_{t'=t+p_i}^{h-1} X'_{ijt'}, \tag{33}$$

otherwise we would have $X^*_{i,t+p_i} - X^*_{it} \geq \sum_{t'=t+1}^{t+p_i}(X'_{i,j-1,t} + X'_{ijt}) > 1$, and the constraints (19) would be violated. Given $j \in N(i)$, we sum up (33) for all $t \in [1,h]$ and obtain

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

and X_i' satisfies the 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 Theorem 1, we again partition the operations of job i into set M of block. Remember that the first operation in a block either is tardy or is the "zero" operation, and other operations in a block are early or on-time.

Consider block $k \in M$ containing operations $(O_{i,j'_k}, O_{i,j'_k+1}, \ldots, O_{i,j''_k})$, where number of operations is at least two $(j'_k < j''_k)$. Let t_k^1 be the last time moment such that $X'_{ij'_kt} > 0$ and t_k^2 be the first time moment such that $X'_{ij''_kt} > 0$. We denote $\Delta_{ik} = \{t_k^1, \ldots, t_k^2 + l_i - 1\}$. Note that $t_k^2 + l_i \leq t_{k+1}^1$,

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,

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

where η_k is introduced to simplify the presentation.

Note that the contribution of each variable X_{ijt} of the formulation (TI) to the sum $\sum_{t\in\Delta_{ik}} n_i(t) = \eta_k$ 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,

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

Finally,

$$\nu_{SCA}(X^*_i) \geq \sum_{k\in M} \left(\eta_k - t^2_k - l_i + t^1_k\right) \geq \sum_{k\in M} \sum_{j=j'_k+1}^{j''_k} \left(l_i - S'_{ij} + S'_{i,j-1}\right),$$

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

Unfortunately, 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 which data is shown in Table 1 ($h = 120$), then $\nu^*_{TI} \approx 36.1 > 33.2 \approx \nu^*_{TIA}$.

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 $\nu^*_{TI} \leq \nu^*_{TIA}$

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 5.

3.4 Aggregation of identical jobs

In the available practical test instances, usually, there are identical jobs. We call two jobs a and b identical if $p_a = p_b$, $l_a = l_b$, $\alpha_a = \alpha_b$ and $\beta_a = \beta_b$. In order to decrease the size of the formulation (TIA) we will aggregate jobs into groups. We partition the set N of jobs into the set $Q = \{1, \ldots, q\}$ of groups $\{G_k\}_{k \in Q}$ of identical jobs. It is easy to show that there exists an optimal schedule, in which the operations of jobs in the same group alternate, i.e. j-th operations of the jobs in the same group are processed in the same order as the "zero" operations of these jobs. Using this fact, in the next proposition, we show that the penalty functions γ_i for the jobs i in group G_k, $k \in Q$, can be "aggregated" into a single penalty function θ_k for the group. Let $G_{kt} = \{i \in G_k : \ t \in \Delta_i\}$. We renumber the jobs such that, for each $k \in Q$, $G_k = \{i'_k, i'_k + 1, \ldots, i''_k\}$, $S_{i'_k 0} < S_{i'_k+1,0} < \ldots < S_{i''_k}$. We set $\bar{H}_k = H_{i'_k}$, and

$$\theta_k(t) = \max \left\{ \alpha_i \left(\sum_{i \in G_k} n_i(t) - |G_{kt}| \right) \ , \ \beta_i \left(|G_{kt}| - \sum_{i \in G_k} n_i(t) \right) \right\}.$$

Theorem 3 *Consider a group G_k, $k \in Q$, of jobs and a schedule $\{S_{i0}, S_{i1}, \ldots, S_{i,n(i)}\}_{i \in G_k}$, in which the operations of jobs in G_k alternate. Then, for all $t \in \bar{H}_k$,*

$$\sum_{i \in G_{kt}} \gamma_i(t) = \theta_k(t).$$

Proof. To prove the proposition, we need to show that, given $t \in H_k$,

$$\sum_{i \in G_k} n_i(t) \geq |G_{kt}| \quad \Leftrightarrow \quad n_i(t) \geq 1, \ \forall i \in G_{kt}, \tag{34}$$

$$\sum_{i \in G_k} n_i(t) \leq |G_{kt}| \quad \Leftrightarrow \quad n_i(t) \leq 1, \ \forall i \in G_{kt}, \tag{35}$$

The implication "$\Leftarrow$" of (34) is obvious. We will now prove the implication "$\Rightarrow$" (34) by contradiction. Suppose that $\sum_{i \in G_k} n_i(t) \geq |G_{kt}|$ and there exists a job $i' \in G_{kt}$ such that $n_{i'}(t) = 0$. Then, there is a job $i'' \in G_{kt}$ such that $n_{i''}(t) \geq 2$. This means that, in the schedule, between two operations of job i'', there are no operations of job i' and an operation of job i' starts later than at t (as $i' \in G_{kt}$ and $n_{i'}(t) = 0$). This contradicts the condition of the proposition which states that the operations of jobs i' and i'' alternate.

The implications in (35) can be proved in the same way. □

So, to find an optimal solution of the problem, we can limit the search to the set of "alternating" schedules and use the penalty function $\sum_{k \in Q} \sum_{t \in \bar{H}_k} \theta_k(t)$ instead of $\sum_{i \in N} \sum_{t \in H_i} \gamma_i(t)$. We will now modify the formulation (TIA). Integer variable X_{kt}, $k \in Q$, $t \in H$, is now equal to the number of operations of jobs in G_k started before or at time moment t. Continuous variable W_{kt}, $k \in Q$, $t \in \bar{H}_k$, now represents the value $\theta_k(t)$. Here we again need to introduce additional variables E in order to compute $|G_{kt}|$, $k \in Q$, $t \in H$. To limit the number of such variables and to simplify the formulation, to the same group we include identical jobs a and b only if either $n(a) = n(b)$ or $n(a) = n(b)+1$ and $S_{a0} < S_{b0}$. Binary variable E_{kst}, $k \in Q$, $s \in \{1, \ldots, |Q_k|\}$, $t \in H_k$, takes value 1 if and only if $|G_{kt}| \geq s$. Again, we do not need to impose the integrality on these variables. We now present the aggregated alternative time-indexed ILP formulation. For each $k \in Q$, we set $\bar{p}_k = p_{i'_k}$, $\bar{l}_k = l_{i'_k}$, $\bar{\alpha}_k = \alpha_{i'_k}$, $\bar{\beta}_k = \beta_{i'_k}$.

$$\min \quad \sum_{k\in Q}\sum_{t\in \bar{H}_k} W_{kt} \tag{36}$$

$$s.t. \quad X_{k,S_{i0}} = i''_k - i + 1, \quad \forall k \in Q,\ i \in G_k, \tag{37}$$

$$X_{k,S_{i0}-1} = i''_k - i, \quad \forall k \in Q,\ i \in G_k, \tag{38}$$

$$X_{k,t-1} \le X_{kt}, \quad \forall k \in Q,\ t \in H \setminus \{0\}, \tag{39}$$

$$X_{k,h-\bar{p}_i} = \sum_{i\in G_k} (n(i)+1), \quad \forall k \in Q, \tag{40}$$

$$\sum_{k\in Q} X_{kt} - \sum_{\substack{k\in Q,\\ t-\bar{p}_k\ge 0}} X_{k,t-\bar{p}_k} \le 1, \quad \forall t \in H, \tag{41}$$

(TIAA)

$$E_{k1t} \ge X_{k,t-\bar{l}_k+1}, \quad \forall k \in Q,\ t \in \bar{H}_k,\ t \ge S_{i''_k,0} + \bar{l}_k, \tag{42}$$

$$E_{kst} \ge X_{k,t-\bar{l}_k+1} + E_{k,s-1,t+1}, \quad \forall k \in Q,\ s \in \{2,\dots,|Q_k|\},\ t \in \bar{H}_k,\ t \ge S_{i''_k,0} + \bar{l}_k, \tag{43}$$

$$E_{kst} \le E_{k,s,t-1} \le 1, \quad \forall k \in Q,\ s \in \{1,\dots,|Q_k|\},\ t \in \bar{H}_k,\ t > S_{i''_k,0}, \tag{44}$$

$$W_{kt} \ge \bar{\alpha}_k \left(X_{kt} - X_{k,t-\bar{l}_k} - \sum_{s=1}^{|Q_k|} E_{kst} \right), \quad \forall k \in Q,\ t \in \bar{H}_k, \tag{45}$$

$$W_{kt} \ge \bar{\beta}_k \left(\sum_{s=1}^{|Q_k|} E_{kst} - X_{kt} + X_{k,t-\bar{l}_k} \right), \quad \forall k \in Q,\ t \in \bar{H}_k, \tag{46}$$

$$X_{kt} \in \mathbb{Z}_+, \quad \forall k \in Q,\ t \in H, \tag{47}$$

$$E_{kst} = 0,\ \forall k \in Q,\ s \in \{2,\dots,|Q_k|\},\ t \in \bar{H}_k,\ t < S_{i'_k+s-1,0} \tag{48}$$

The constraints (37)-(38) fix the position of the "zero" operations. The constraints (39) reflect the nature of the variables X. The constraints (40) state that all operations of jobs in G_k should be processed within the time horizon. The constraints (41) prevent overlapping. The constraints (42)-(44) relate the variables X and E. The constraints (45)-(46) are used to compute the values of the variables W. Here $\sum_{s=1}^{|Q_k|} E_{kst} = |G_{kt}|$, $\forall k \in Q$, $t \in \bar{H}_k$.

The formulation (TIAA) has roughly $(2q+n)h$ variables, qh from which are integer, $(3q+2n+1)h$ constraints and $(10q+8n)h$ non-zero coefficients in the matrix. So, the size of this formulation depends first and foremost on the length of the time horizon h, then on the number of groups of identical

jobs q, and after all the others on the number of jobs n.

4 Non-compact formulation

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

4.1 The master problem

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 total penalty for it. Let also $\chi(i,k,t)$ be a characteristic function which indicates whether schedule S_i^k is "active" at time moment t:

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

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

$$\text{(MP)} \qquad \begin{aligned} \min \quad & \sum_{i \in N} \sum_{k \in K_i} w_i^k X_{ik} && (49) \\ s.t. \quad & \sum_{i \in N} \sum_{k \in K_i} \chi(i,k,t) X_{ik} \leq 1, \quad \forall t \in H. && (50) \\ & \sum_{k \in K_i} X_{ik} = 1, \quad \forall i \in N, && (51) \\ & X_{ik} \in \{0,1\}, \quad \forall i \in N,\ k \in K_i. && (52) \end{aligned}$$

The constraints (50) forbid overlapping, i.e. state that at each time moment only one schedule is active. The constraints (51) guarantee that each job is processed.

4.2 The pricing sub-problems

The formulation (MP) cannot be directly solved by a MIP solver, as the number of variables is huge. The standard approach for solving the master problem is column generation. It consists in iterating between a solution of the LP relaxation (RMP) of the master problem involving only a subset of the 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 optimal values of the dual variables which correspond to the constraints (50), and $\mu \in \mathbb{R}^n$ be the optimal values of the dual variables which correspond to the constraints (51). Then, the reduced cost for column column X_i^k is equal to

$$w_i^k + \sum_{t \in H} \chi(i,k,t)\pi_t - \mu_i. \tag{53}$$

Consequently, to verify if there is a column with a negative reduced cost, the pricing problem should be solved in which we search a schedule S_i^k minimizing (53). Given the structure of the reduced cost function (53), the pricing problem can be decomposed into sub-problems, one for each job $i \in N$. Each sub-problem can be formally written as

$$(\mathrm{PS}_i) \qquad \min \quad \sum_{j \in N(i)} \delta_i(S_{ij} - S_{i,j-1}) + \sum_{j=0}^{n(i)} \sum_{t=S_{ij}}^{S_{ij}+p_i-1} \pi_t - \mu_i \tag{54}$$

$$s.t. \quad S_{ij} \geq S_{i,j-1} + p_i, \quad \forall j \in N(i), \tag{55}$$

$$S_{i,n(i)} + p_i \leq h. \tag{56}$$

Basically, this sub-problem consists in finding a schedule of operations of job i which minimizes the sum of the penalty plus a dual cost (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 dynamic programming. Let $R_i(j,t)$ 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 $R_i(n(i),t) = \Pi_i(t)$, $t \in [0, h - p_i]$. Then, the recursion formula is

$$R_i(j,t) = \Pi_i(t) + \min_{t' \in [t+p_i, h-p_i]} \{\delta_i(t'-t) + R_i(j+1,t')\}, \tag{57}$$

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

The solution value of the problem (PS_i) is equal to $R_i(0, S_{j0}) - \mu_i$. Using the recursion above, the problem (PS_i) can be solved in $O(n(i) \cdot h^2)$ time, which is not practical. As shown in the next subsection, this can be improved when using the structure of the penalty function δ.

4.3 Fast algorithm for the pricing sub-problem

First, we define the following functions.

$$\begin{aligned} a_i(j,t) &= \operatorname*{argmin}_{t' \in [0,t]} \{\alpha_i(t-t') + R_i(j+1,t')\}, \\ b_i(j,t) &= \operatorname*{argmin}_{t' \in [t,h-p_i]} \{\beta_i(t'-t) + R_i(j+1,t')\}. \end{aligned}$$

Using these functions, we can rewrite the formula (57). If $a_i(j, t+l_i) \geq t+p_i$ then

$$R_i(j,t) = \Pi_i(t) + \min_{t' \in \{a_i(j,t+l_i), b_i(j,t+l_i)\}} \{\delta_i(t'-t) + R_i(j+1,t')\}. \tag{58}$$

If $a_i(j, t+l_i) < t+p_i$ then

$$R_i(j,t) = \Pi_i(t) + \min_{t' \in [t+p_i, t+l_i) \cup \{b_i(j,t+l_i)\}} \{\delta_i(t'-t) + R_i(j+1,t')\}. \tag{59}$$

We now estimate the complexity of computing values $R_i(j,t)$, $t \in [0, h-p_i]$, for a given j. Clearly, when $j = n(i)$, this can be done in $O(h)$ time. Suppose now that $j < n(i)$ and values $R_i(j+1,t)$, $t \in [0, h-p_i]$, are known. Then, values $a_i(j,t)$ and $b_i(j,t)$ can be also found in $O(h)$ time in the following way. We set $a_i(j,0) = 0$, $b_i(j, h-p_i) = h-p_i$, and then

$$a_i(j,t) = \begin{cases} t, & P_i(j+1,t) < P(j+1, a_i(j,t-1)) + \\ & \qquad \alpha_i \cdot (t - a_i(j,t-1)), \\ a_i(j,t-1), & \text{otherwise}, \end{cases}$$

$$b_i(j,t) = \begin{cases} t, & P_i(j+1,t) < P(j+1, b_i(j,t+1)) + \\ & \qquad \beta_i \cdot (b_i(j,t+1) - t), \\ b_i(j,t+1), & \text{otherwise}. \end{cases}$$

So, if one uses the recursion (58)-(59), 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 showed 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.

At some iteration of the column generation procedure, no column with a negative reduced cost will be found after resolution of 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. Such a combination is conventionally called a *branch-and-price* algorithm.

4.4 Branch-and-price algorithm

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 could not be applied beyond the top node of the search tree. For example, a usual way to branch on 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, application of 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 moment 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 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

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

Then, two child nodes are created. 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 (57) becomes

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

The recursion (58)-(59) can be also modified appropriately.

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* [10]. An $\alpha-$point of operation O_{ij} is defined to be the first time moment $t(\alpha_{ij})$, $0 < \alpha_{ij} \leq 1$, 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 in which the operations are processed according to the heuristic sequence. For this, it suffices to solve a linear program.

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.

5 Experimental research

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.

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:

- LB — lower bound given by an LP relaxation, or the best lower bound obtained while solving a formulation.
- UB — the best upper bound (solution) obtained while solving a formulation.
- Tm — the time needed to solve an LP relaxation/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.

First, we have tested the formulations of thee small instances, which data is given in Table 2. These test instances are of the same type as real life instances but of a reduced size. Results are presented in Table 3.

Instance	bib2	bib3	bib5
n	3	5	6
$(n(i); p_i, d_i; \alpha_i, \beta_i)$	(4; 9,35; 10,10) (4; 13,35; 10,10) (12; 5,5; 1,1)	(4; 9,50; 10,10) (4; 9,50; 10,10) (4; 13,50; 10,10) (4; 13,50; 10,10) (6; 5,5; 1,1)	(4; 6,50; 2,2) (4; 6,50; 2,2) (5; 6,40; 2,2) (4; 9,50; 1,1) (4; 9,50; 1,1) (5; 9,40; 1,1)
h	150	208	200

Table 2: Data for small instances

Test	(TIS)			(TIAA)			Branch-and-price		
	LB	UB	Tm	LB	UB	Tm	LB	UB	Tm
bib2(LP)	62.3		1.3s	136.5		0.3s	143.2		9.1s
bib2	101.8	1604	>10m	248.0	**248**	17s	248.0	**248**	6m
bib3(LP)	132.0		3.4s	188.5		1.2s	243.8		9.0s
bib3	158.4	—	>10m	302.0	**302**	32s	302.0	**302**	37s
bib5(LP)	10.0		0.3s	46.0		1.4s	46.0		1.4s
bib5	15.0	132	>30m	105.0	**105**	11m	105.0	**105**	23m

Table 3: Comparison of formulations

From these results it is clear that the formulation (TIS) is much less efficient than the other two formulations. So, it was excluded from the following numerical experiments.

The second 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 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 4 and 5. 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 5). 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. These instances are highly symmetric, i.e. there are an enormous number of schedules (and columns) with the same penalty value. Therefore the column generation algorithm needs to generates a lot of columns before converging to an optimum.

The formulation (TIAA) is much more successful on the "sc" instances. In some time, it manages 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 generated using the following procedure. The time horizon length h, the density d (the relation between the total processing time of operations and the horizon length), the number n of jobs and the total number of operations $o = \sum_{i=1}^{n} n(i)$ are fixed. Let $a = \lfloor \frac{o}{n} \rfloor$ and $b = \lfloor \frac{dh}{o} \rfloor$. Then, for each job $i \leq n$, $n(i)$ and p_i are uniformly generated over $[1, 2a]$ and $[1, 2b]$ accordingly in such a way that $\sum_{i=1}^{n} p_i n(i) = dh$. Next, for each job i, we set $l_i = \lfloor \frac{h}{n(i)} \rfloor$. At last, "zero" operations are scheduled before time moment 0 one just after another in the reverse order:

$$S_{10} = -p_1, \qquad S_{i0} = S_{i-1,0} - p_i, \ i \in [2, \ldots, n].$$

The instances were generated for the next values of the parameters: $h \in \{250, 500\}$, $d \in \{0.5, 0.75, 1.0\}$, $o \in \{30, 60, 100, 150\}$, $n \in \{3, 6, 15, 30\}$. For each combination of values $\{h, d, o, n\}$ satisfying the conditions $3o < dh$ and $3n < o$, 2 instances were created. Therefore, in total, 234 instances were tested.

The formulation (TIS) was not tested. Firstly, because of its poor results for the three small instances. Secondly, because of the quality of the lower bound given by the LP relaxation of the formulation (TIS). For the generated random instances, it was always less than the lower bound of the LP relaxation of the formulation (TIA), and on the average the relation between these two bounds was only 25.7%.

In the following, we present results for the following statistics:

- P — percentage of instances solved within 30 minutes.
- Tm — average solution time (if, for some instance, the time limit of 1800 seconds was reached, this time limit is included into statistics).
- Gap — average difference between the best upper bound known and the best lower bound obtained, in percents from the best solution known

(this statistics takes into account only instances for which an upper bound is known and the LP relaxation can be solved within the time limit).

- Nd — average number of nodes in the branch-and-bound or branch-and-price tree (only instances for which the LP relaxation can be solved within the time limit).

The results for the random instances are presented in Tables 6 and 7. The statistics for the whole set of instances show that the efficiency of the both approaches is roughly the same. The strength of the lower bound given by the solution of the LP relaxation of the formulation (TIA) is almost the same in comparison to the lower bound obtained by the column generation procedure (in average, the relation is 94.2%). This advantage of the column generation procedure results in slightly smaller gaps on average for the branch-and-price algorithm. Unfortunately, these gaps remain quite large even after 30 minutes of the solution time. So, the random instances happened to be harder to solve than the real life instances.

It can be also seen that instances with longer time horizon or with bigger operations density d are harder to solve. The impact of the number of jobs is not that simple. The results show that instances with small number of jobs are simpler for the formulation (TIA) and harder for the branch-and-price algorithm. This can be better seen in Table 8 where detailed results for the statistics T_{LP}, Gap_{LP} and Gap are presented. From Table 8 it is clear that such a dependence of the approach efficiency from the number of jobs comes from the LP relaxations. The solution of the LP relaxation of the formulation (TIA) takes much more time for instances with larger number of jobs. The contrary takes place when the column generation procedure is concerned.

The main conclusion of the experimental research is that the formulation (TIA) or (TIAA) should be used when the number of jobs (or the number of groups of identical jobs) is small (6 or less). When the number of jobs is more and each job contains a moderate number of operations, the branch-and-price algorithm should be preferred.

6 Conclusions and perspectives

In this paper we have introduced a generalization for the classic earliness-tardiness single machine scheduling problem. The motivation for this gen-

eralization 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. These two approaches are somewhat complementary to each other. The latter is better suited for solving instances with small number of jobs, and the former is more efficient when applied to instances for which the relation between the total number of operations and the number of jobs is small.

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, 9]. 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 straightforwardly 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 efficiency of the methods proposed, one may think of applying a Lagrangean relaxation. For example, in [12], 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 particularly for supplying real life instances.

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

[3] 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.

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

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

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

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

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

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

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

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

Test	h	q	LB_{LP}	Tm_{LP}	LB	UB	Tm	Nd
sc_2_01_4 (10,87)	600	4	59.1	185s	100.4	**100.4**	93m	448
sc_2_02_4 (10,75)	600	3	59.0	26s	92.7	**92.7**	37m	2038
sc_4_01_4 (11,77)	450	4	46.4	30s	74.3	**74.3**	15m	545
sc_4_02_4 (11,58)	450	4	43.9	14s	64.0	**64.0**	8m	568
sc_5_01_4 (11,91)	600	6	89.8	11m	528.7	**528.7**	212m	1087
sc_5_02_4 (11,80)	600	5	84.8	3m	143.1	**143.1**	54m	215
bib_9_57_i2	500	2	27000	1s	27000	**27000**	1s	0
bib_14_62_i2	500	2	3840	3s	3840	**3840**	3s	0
bib_14_83_i2	500	3	33500	21s	33500	**33500**	21s	0
bib_15_91_i2	500	5	7885.1	44s	8560	**8560**	79m	406
bib_15_91_i4	500	5	6723.5	24s	6728.2	**6728.2**	200s	18
bib_15_91_i8	500	5	80.2	88s	85.6	**85.6**	507s	265
bib_15_93_i2	500	5	8227.1	48s	8378.3	8880	>2h	1115
bib_15_93_i4	500	5	6728.6	47s	6730.2	**6730.2**	101s	5
bib_15_93_i8	500	5	85.6	102s	87.6	**87.6**	364s	65
bib_15_96_i2	500	5	8659.3	77s	8735.9	9810	>2h	338
bib_15_96_i4	500	5	6736.2	56s	6737.2	**6737.2**	64s	0
bib_15_96_i8	500	5	90.6	81s	94.6	**94.6**	857s	314
bib_18_72_i4	500	4	8598.8	7s	8653.8	**8653.8**	550s	169
bib_18_74_i8	500	4	87.3	89s	89.8	96.2	>2h	2811
bib_18_76_i2	500	4	7152.3	50s	7184.3	**7184.3**	158s	10
bib_15_151_i4	900	5	286.5	129s	292.7	**292.7**	321s	14
bib_15_151_i10	900	5	34.7	390s	44.9	**44.9**	6367s	571
bib_20_171_i1	1350	6	6977.2	5960s	6978.3	-	>2h	1

Table 4: Results for real life instances — the formulation (TIAA)

Test	h	LB_{LP}	Tm_{LP}	LB	UB	Tm	Nd
bib_9_57_i2	500	27000	2s	27000	**27000**	2s	0
bib_14_62_i2	500	3840	1s	3840	**3840**	1s	0
bib_14_83_i2	500	33500	52s	33500	**33500**	52s	0
bib_15_91_i2	500	8019.5	4s	8560	**8560**	24m	513
bib_15_91_i4	500	6728.2	2s	6728.2	**6728.2**	2s	0
bib_15_91_i8	500	84.8	2s	85.6	**85.6**	445s	355
bib_15_93_i2	500	8468.2	5s	8684.5	8760	>5h	4076
bib_15_93_i4	500	6730.2	5s	6730.2	**6730.2**	5s	0
bib_15_93_i8	500	86.7	4s	87.6	**87.6**	638s	131
bib_15_96_i2	500	8826.1	29s	9428.7	9700	>5h	338
bib_15_96_i4	500	6737.2	96s	6737.2	**6737.2**	96s	0
bib_15_96_i8	500	93.0	11s	94.6	**94.6**	6322s	17
bib_18_72_i4	500	8598.8	7s	8653.8	**8653.8**	3821s	2855
bib_18_74_i8	500	88.8	4s	95.9	**95.9**	4617s	1143
bib_18_76_i2	500	7153.7	12s	7184.3	**7184.3**	13082s	879
bib_15_151_i4	900	292.7	16s	292.7	**292.7**	16s	0
bib_15_151_i10	900	43.8	343s	44.0	44.9	>5h	4
bib_20_171_i1	1350	7524.8	10515s	7524.8	13210	>5h	2

Table 5: Results for real life instances — branch-and-price algorithm

Selection	P_{LP}	Tm_{LP}	Gap_{LP}	P	Tm	Gap	Nd
All instances	94.9%	222s	60.0%	26.1%	1458s	40.4%	2902
$h = 250$	100.0%	13s	57.2%	44.4%	1187s	26.4%	4660
$h = 500$	92.6%	315s	61.5%	17.9%	1578s	47.6%	1537
$d = 0.5$	100.0%	13s	47.6%	71.4%	772s	11.9%	1430
$d = 0.75$	100.0%	155s	59.3%	24.1%	1550s	37.9%	4090
$d = 1.0$	88.6%	363s	69.0%	13.3%	1597s	55.3%	4577
$n = 3$	100.0%	5s	68.2%	45.2%	1104s	37.4%	4291
$n = 6$	100.0%	36s	59.9%	22.6%	1568s	41.4%	1618
$n = 15$	100.0%	403s	50.4%	8.3%	1756s	45.5%	309
$n = 30$	33.3%	1614s	28.0%	0.0%	1800s	27.6%	1

Table 6: Results for random instances — the formulation (TIA)

Selection	P_{LP}	Tm_{LP}	Gap_{LP}	P	Tm	Gap	Nd
All instances	93.2%	212s	57.2%	26.1%	1385s	32.4%	802
$h = 250$	97.2%	72s	54.6%	37.5%	1202s	20.8%	1443
$h = 500$	91.4%	274s	58.4%	21.0%	1467s	37.9%	499
$d = 0.5$	100.0%	1s	46.5%	73.8%	512s	5.2%	1389
$d = 0.75$	100.0%	2s	57.0%	33.7%	1321s	23.3%	1322
$d = 1.0$	86.7%	422s	63.8%	12.4%	1611s	48.8%	713
$n = 3$	86.9%	294s	67.5%	33.3%	1272s	35.7%	965
$n = 6$	94.1%	231s	57.6%	30.6%	1313s	30.1%	849
$n = 15$	100.0%	96s	47.8%	14.6%	1554s	31.9%	677
$n = 30$	100.0%	50s	38.7%	0.0%	1800s	30.3%	268

Table 7: Results for random instances — the branch-and-price algorithm

	Branch-and-price								
	Tm_{LP}			Gap_{LP}			Gap		
$d =$	0.5	0.75	1.0	0.5	0.75	1.0	0.5	0.75	1.0
$n = 3$	.2s	3s	682s	56%	70%	77%	8%	29%	67%
$n = 6$	.3s	1s	539s	45%	58%	65%	5%	23%	51%
$n = 15$	.6s	2s	191s	21%	46%	56%	0%	21%	49%
$n = 30$	-	2s	74s	-	26%	45%	-	10%	40%
	Formulation (TIA)								
	Tm_{LP}			Gap_{LP}			Gap		
$d =$	0.5	0.75	1.0	0.5	0.75	1.0	0.5	0.75	1.0
$n = 3$	1s	3s	9s	58%	72%	74%	8%	37%	56%
$n = 6$	8s	23s	61s	46%	61%	69%	17%	40%	57%
$n = 15$	63s	256s	600s	22%	48%	59%	8%	41%	58%
$n = 30$	-	1243s	1800s	-	28%	-	-	28%	-

Table 8: Detailed results for random instances