A Branch-And-Price Algorithm for Scheduling Observations on a Telescope

Nicolas Catusse, Hadrien Cambazard, Nadia Brauner, Pierre Lemaire, and **Bernard Penz**
Univ. Grenoble Alpes, G-SCOP, F-38000 Grenoble, France
CNRS, G-SCOP, F-38000 Grenoble, France
Anne-Marie Lagrange and **Pascal Rubini**
Univ. Grenoble Alpes, IPAG, F-38000 Grenoble, France
CNRS, IPAG, F-38000 Grenoble, France

Abstract

We address a parallel machine scheduling problem for which the objective is to maximize the weighted number of scheduled tasks, and with the special constraint that each task has a mandatory processing instant. This problem arises, in our case, to schedule a set of astronomical observations on a telescope. We prove that the problem is NP-complete, and we propose a constraint-programming-based branch-and-price algorithm to solve it. Experiments on real and realistic datasets show that the method provides optimal solutions very efficiently.

1 Scheduling observations on a telescope

Large telescopes are few and observing time is a precious resource subject to a strong pressure from a competitive community. Besides, a growing number of astronomical projects require surveying a large number of targets during discontinuous runs spread over few months or years. An important example in today astronomy is the imaging surveys to discover and study extra-solar planets, which require to observe hundreds of targets to get a chance of imaging a few planets.

There are many targets to be selected as actual observations and the total demand of observing time far exceeds the supply. Furthermore, the optimization must occur on different scales: prior to each year or semester, the requested targets list must fit the projected calendar, leading to adjust both; then the schedule is optimized again prior to each run (roughly, every week) and re-optimized prior to each night, to take into account priority changes and to reschedule failed observations. Even during one night, unexpected conditions such as bad weather might indeed require quasi real time adjustments, calling for efficient algorithms.

Software solutions exist to cope with scheduling observations. SPIKE, originally designed for the Hubble Space Telescope, addresses specific cases and uses a constraint programming approach, together with a multistart stochastic repair procedure [Johnston and Miller, 1994]. SPOT is a recent integrated tool designed for automating the schedule computation in the case of a search for extra-solar planets: it first analyzes the various constraints and determines the observation windows of each target in each night, and then computes actual schedules using heuristic and meta-heuristic techniques. It is fast (5 minutes for 1000 targets on a modern laptop), provides a friendly user interface and is portable on all major operating systems [Lagrange *et al.*, 2015].

In this paper we investigate alternative, *exact* methods to cope with the typical kind of instances which are frequently addressed to SPOT and that can be described as a parallel machine scheduling problem with time-windows and the particular constraint that each task has a mandatory instant at which it must be in execution; the problem is formally defined in Section 2, together with notations, a mixed-integer programming formulation and the many links with interval scheduling problems. Then, we prove the NP-completeness of the problem (Section 3) and propose solution procedures and bounds (Section 4). Experimental results on real and realistic datasets are discussed in Section 5.

2 Problem description

We are given a set of allocated nights and a targets list. Each target has a scientific priority and many properties as its coordinates, its angular proximity to the moon, its elevation above the horizon, *etc.* Each night has also specific properties: its starting and ending times as well as particular conditions (e.g, due to weather) that prevent the observation of some specific directions or target faint objects. Furthermore, in the case of extra-solar planet surveys, a target must be observed when it crosses its meridian and the exposure must ensure a minimum rotation of the field of view. Altogether, this information defines, for every target: an interest, an observation duration, a mandatory instant of observation (the meridian) and an observation window; note that, apart from the interest, all these parameters are night-dependent.

As a consequence, the problem we address is a parallel scheduling problem with an apriori mandatory part known for each task (nights can be seen as machines and targets as tasks). It is precisely defined as follows.

We consider a set $\mathcal{M}$ of m nights and a set $\mathcal{T}$ of n targets. Each target i has an interval $[r_i^j, d_i^j[$ when it can be observed during night j, and a duration of observation p_i^j; furthermore each target must be observed at its meridian m_i^j. The telescope can observe only one target at once, and each target i has a weight w_i that represents its "interest". The objective is to maximize the total weight of the observed targets by decid-

ing the allocation of the targets to the nights and the schedule of the observations within the nights.

The existence of mandatory instants (meridians) implies a particular structure on both the instances and the solutions:

Property 1 *A target i has a mandatory instant during night j if and only if $2p_i^j \geq d_i^j - r_i^j$.*

Proof If m_i^j is a mandatory instant then $r_i^j \geq m_i^j - p_i^j$ and $d_i^j \leq m_i^j + p_i^j$ and thus $2p_i^j \geq d_i^j - r_i^j$. Conversely, if $2p_i^j \geq d_i^j - r_i^j$ then $t_i^j = (r_i^j + d_i^j)/2$ can be considered as a mandatory instant. □

Property 2 *The observations must be scheduled by non-decreasing mandatory instants (for a given night j).*

Proof Consider two targets 1 and 2 such that $m_1^j < m_2^j$. Target 2 ends after m_2^j (mandatory instant) and thus strictly after m_1^j: hence, target 1 cannot respect its mandatory instant m_1^j if it is schedule after target 2. □

Property 2 is a key feature for everything that follows as the order of the observations scheduled during a same night is known in advance. First, it allows for the following mixed-integer linear formulation:

$$\max \sum_{i \in \mathcal{T}} w_i z_i$$

$$\begin{array}{lccll}
(1) & \sum_{j \in \mathcal{M}} z_i^j & = & z_i & i \in \mathcal{T} \\
(2) & r_i^j z_i^j & \leq & t_i & i \in \mathcal{T}, j \in \mathcal{M} \\
(3) & t_i + p_i^j z_i^j & \leq & d_i^j z_i^j + M(1 - z_i^j) & i \in \mathcal{T}, j \in \mathcal{M} \\
(4) & z_{i_1}^j + z_{i_2}^j & \leq & y_{i_1,i_2} + 1 & i_1, i_2 \in \mathcal{T} \\
 & & & & j \in \mathcal{M} \\
(5) & t_{i_1} + p_{i_1}^j & \leq & t_{i_2} + M(1 - y_{i_1,i_2}) & i_1 \prec_j i_2 \in \mathcal{T} \\
 & & & & j \in \mathcal{M}
\end{array}$$

where the decision variables are: z_i (binary, 1 if and only if target i is scheduled); z_i^j (binary, 1 if and only if target i is scheduled during night j); y_{i_1,i_2} (binary, 1 if and only if targets i_1 and i_2 are scheduled during the same night); and t_i (non-negative, the starting time of the observation of target i).

The objective sums up the total interest of the scheduled targets. Constraints (1) ensure that each scheduled target is actually scheduled within some night. Constraints (2) and (3) impose that a target, if scheduled during night j, respects the corresponding observation interval (note: M is a large constant, e.g., the length of the longest night). Constraints (4) and (5) ensure that two observations scheduled during the same night do not overlap; in constraints (5), "$i_1 \prec_j i_2$" refers to the order induced by Property 2. This mixed-integer formulation allows for a clear statement of the problem, but is improper for a fast solution of large instances.

When interpreted in scheduling terms, the problem, due to Property 1, is described by the notation $R|r_i^j, 2p_i^j \geq d_i^j - r_i^j| \sum w_i U_i$ (see *e.g.,* [Pinedo, 2014] for classical notations and results from scheduling theory). If the observation intervals do not depend on the night ($r_i^j = r_i$; $d_i^j = d_i$; $p_i^j = p_i$), then nights are identical machines, and the problem becomes a special case of the parallel machine scheduling problem with release and due dates where one wants to minimize the weighted number of unscheduled tasks ($P|r_i| \sum w_i U_i$). This problem is NP-complete, even for a single night, no release dates and a unique due-date ($1|d_i = d| \sum w_i U_i$), or with no release dates, no due dates, and no weights ($P|| \sum U_i$), or for one night and no weights ($1|r_i| \sum U_i$). However, the special case $1|r_i, p_i = p| \sum w_i U_i$ is polynomially solvable [Baptiste, 1999], as is the case for $1|r_i, p_i = d_i - r_i| \sum w_i U_i$ [Chuzhoy *et al.*, 2006].

If the duration of the interval is exactly the processing time but the observation intervals depend on the night (*i.e.*, $p_i^j = d_i^j - r_i^j$), then the problem belongs to interval scheduling (see [Kolen *et al.*, 2007] for a review). If all targets have the same weight, then the problem can be solved in polynomial time by a greedy algorithm [Faigle and Nawijn, 1995; Carlisle and Lloyd, 1995]. If the duration does not depend on the night ($p_i = d_i - r_i$), the problem can be formulated as a minimum cost flow and is solvable in polynomial time [Arkin and Silverberg, 1987; Bouzina and Emmons, 1996]. However, when some targets are not visible some nights, the problem becomes NP-complete, except if the number of nights is fixed [Arkin and Silverberg, 1987].

When a discrete set of k intervals is associated to a target (e.g., one possible interval per night), then the problem of deciding whether all targets can be scheduled is NP-complete if $k \geq 3$ [Nakajima and Hakimi, 1982], and is polynomial if $k = 2$ [Keil, 1992]. However, scheduling a maximum number of targets with discrete intervals is NP-complete if $k \geq 2$ [Spieksma, 1999].

To the best of our knowledge, the assumption $2p_i^j \geq d_i^j - r_i^j$ has not been studied before. It implies a particular structure of a feasible solution but nevertheless, as we prove in the next section, the problem remains NP-complete.

3 Complexity analysis

The complexity of various related problems has been discussed above; Table 1 sums up the main results. In this section we prove that, with the assumption that $2p_i \geq d_i - r_i$ the problem is NP-complete but solvable in pseudo-polynomial time if there is one night (Section 3.1), and is unary NP-complete for several nights (Section 3.2). Note that NP-completeness is not straightforward since the problem is polynomial with either equal weights or duration equals to the observation window.

Case	Complexity
$1\|\|\sum U_i$	polynomial
$1\|d_i = d\| \sum w_i U_i$	NP-complete
$1\|r_i, p_i = p\| \sum w_i U_i$	polynomial
$1\|r_i\| \sum U_i$	NP-complete
$1\|r_i, 2p_i \geq d_i - r_i\| \sum U_i$	polynomial*
$1\|r_i, p_i = (d_i - r_i)\| \sum w_i U_i$	polynomial
$1\|r_i, 2p_i \geq d_i - r_i\| \sum w_i U_i$	NP-complete*
$P\|r_i, p_i = r_i - d_i\| \sum w_i U_i$	polynomial
$P\|\|\sum U_i$	NP-complete
$P\|r_i, 2p_i \geq d_i - r_i\| \sum w_i U_i$	NP-complete*

Table 1: Complexity results. Stared-results are proven below.

3.1 The single night problem

The problem reduced to one night is a special case of the one machine scheduling problem with release dates ($1|r_j|\sum w_jU_j$, NP-complete). It is indeed a constant relative window size problem, where each task verifies $k\,p_i \geq d_i - r_i$ (polynomial for $k = 1$): we consider the case $k = 2$ and we prove that the weighted problem is NP-complete (Theorem 1) and the unweighted case is polynomial (Proposition 1).

Theorem 1 *The single-night observation scheduling problem ($1|r_i, 2p_i \geq d_i - r_i|\sum w_iU_i$) is NP-complete.*

Proof The single-night observation scheduling problem is trivially in NP. We shall reduce a variant of the PARTITION problem to ours.

PARTITION: given n pairs of integers (a_{2i-1}, a_{2i}) such that $\sum_{i=1}^{2n} a_i = 2B$, is there a set S such that: S contains exactly one element of each pair (a_{2i-1}, a_{2i}), and $\sum_{i\in S} a_i = B$? This problem is NP-complete [Garey and Johnson, 1979].

Consider an instance of PARTITION. For each pair of integers, we create a pair of incompatible targets. More precisely, we consider $2n$ targets such that $w_i = p_i = 2B + a_i$ and:

- $r_{2i-1} = r_{2i} = 2(i-1)B + \sum_{k=1}^{i-1} \min(a_{2k-1}, a_{2k})$,
- $d_{2i-1} = d_{2i} = 2iB + \sum_{k=1}^{i} \max(a_{2k-1}, a_{2k})$, except for the last two targets: $d_{2n-1} = d_{2n} = 2(n+1)B$.

(Note that the assumption $2p_i \geq d_i - r_i$ holds, and that the transformation is polynomial.) The question is: is there a schedule of these targets of total interest at least $(2n+1)B$?

First, assume that PARTITION has a solution S. We build an observation schedule by picking the targets corresponding to the integers in S: they can be scheduled within their time-window and are done so without idle-time. The total interest of such a schedule is $2(n+1)B$. Hence if PARTITION has a solution, then, there is a schedule of value $2(n+1)B$.

Conversely, let assume that there is a feasible schedule of total interest $2(n+1)B$. Such a value implies that this schedule is non-idling, as the value of each observation is equal to its duration. Besides, for a given i, note that no target other than $2(i-1)$ or $2i$ can be scheduled during the interval $]2(i-1)B + \sum_{k=1}^{i-1} \max(a_{2k-1}; a_{2k}); 2iB + \sum_{k=1}^{i} \min(a_{2k-1}; a_{2k})[$, and that at most one of them can be scheduled, for they are overlapping. Since the schedule is non-idling, one target from each pair $(2(i-1), 2i)$ has been selected and the schedule corresponds to a solution of PARTITION. Hence, if there is a schedule of value $2(n+1)B$, then PARTITION has a solution.

As a consequence, solving the observation scheduling problem solves PARTITION. As the latter is NP-complete, so is the former. □

Proposition 1 *If all targets have the same interest ($w_i = 1$), then the single-night observation scheduling problem ($1|r_i, 2p_i \geq d_i - r_i|\sum U_i$) is polynomially solvable.*

Proof The selected targets must be scheduled in a predefined order (Property 2); we assume that they are numbered accordingly ($1 \prec 2 \prec \ldots \prec n$). Apply the following procedure:

1. Set initial solution $S = \emptyset$
2. For $i = 1$ to n do
   - if target i is compatible with every observation in S, schedule it as early as possible;
   - otherwise, target i is incompatible with target k, the last target added to S; replace k with i if i ends before k when scheduled as early as possible.

This algorithm runs in polynomial time and, besides, it is optimal. Indeed, let S^* be an optimal solution and let k^* and k be the first observations to differ respectively in S^* and S. By definition of the procedure, k ends before k^* and is thus compatible with all the remaining observations of S^*: the procedure must have selected them (or alternative observations that, in each case, ends before the corresponding observations in S^*). As a consequence S has as many observations as S^*, and is thus optimal. □

The known order of the observations within a night allows for a procedure based on standard dynamic programming:

Theorem 2 *The single-night observation scheduling problem ($1|r_i, 2p_i \geq d_i - r_i|\sum w_iU_i$) can be solved in pseudo-polynomial time.*

Proof Consider a set of n targets and a night length $T = \max_{i=1..n} d_i$. Let $f^*(i,t)$ be the optimal value of a schedule involving targets $1, \ldots, i$ and such that i finishes before t. f^* verifies:

$$f^*(i,t) = \begin{cases} 0 & i = 0, \forall t \in [0,T] \\ f^*(i-1,t) & \forall i \in [1,n], t \in [0, r_i + p_i[\\ \max(f^*(i-1,t), f^*(i-1, t-p_i) + w_i) & \\ & \forall i \in [1,n], t \in [r_i + p_i, d_i] \\ f^*(i,d_i) & \forall i \in [1,n], t \in]d_i, T] \end{cases}$$

Note that, for a target i the last three cases span over the whole night: $[0, r_i + p_i[\cup[r_i + p_i, d_i]\cup]d_i, T]$. We are looking for $f^*(n,T)$ which can be computed in space and time complexity $O(nT)$. □

Note that this procedure provides a very efficient algorithm when the time horizon is not too large.

3.2 Several nights problem

If the order of the observations does not depend on the night, then the algorithm of Theorem 2 can be generalized, yielding an $O(nT^m)$ algorithm, pseudo-polynomial for a fixed number of nights m. However, in the general case, the problem is unary NP-complete:

Theorem 3 *The observation scheduling problem is NP-complete in the strong sense when there are several nights, even for the unweighted case and if the observation intervals are the same for all the nights (that is: problem $P|2p_i \geq d_i - r_i|\sum U_i$ is unary NP-complete).*

Proof The observation scheduling problem is in NP. We shall prove that the NUMERICAL 3-DIMENSIONAL MATCHING problem reduces to ours.

NUMERICAL 3-DIMENSIONAL MATCHING (3-DM): given 3 sets of n non negative integers $X = \{x_1, \cdots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$ and $Z = \{z_1, \cdots, z_n\}$ such that $\sum_{i=1}^{n}(x_i + y_i + z_i) = nB$, is there a collection $\mathcal{S}$ of n subsets

$S_j = \{x_{j_1}, y_{j_2}, z_{j_3}\}$ containing exactly one element of each set X, Y and Z such that $\cup_{j=1}^{n} S_j = X \cup Y \cup Z$ and such that for each set S_j: $x_{j_1} + y_{j_2} + z_{j_3} = B$? This problem is NP-complete in the strong sense [Garey and Johnson, 1979].

Now, consider an instance of 3-DM. The reduction is as follows: consider 3 sets of n targets each $\mathcal{T}^X \cup \mathcal{T}^Y \cup \mathcal{T}^Z = \mathcal{T}$; the total number of targets is then $|\mathcal{T}| = 3n$. The duration of a target i is the same whatever the night and is defined by:

- $\forall i \in \mathcal{T}^X$, $p_i^X = 4B + x_i$ with $0 < x_i < B$;
- $\forall i \in \mathcal{T}^Y$, $p_i^Y = 2B + y_i$ with $0 < y_i < B$;
- $\forall i \in \mathcal{T}^Z$, $p_i^Z = B + z_i$ with $0 < z_i < B$.

The intervals $[r_i, d_i]$ associated with each target i are:

- $[0; 5B]$ for $i \in \mathcal{T}^X$;
- $[4B; 8B]$ for $i \in \mathcal{T}^Y$;
- $[6B; 8B]$ for $i \in \mathcal{T}^Z$.

Note that the assumption $2p_i \geq d_i - r_i$ holds. The weight of a target i is 1. The number of nights is $|\mathcal{M}| = n$. The question is: is there a schedule of the observations with a total gain of $3n$?

The complexity of this reduction is pseudo-polynomial, hence it is a pseudo-polynomial transformation, provided that 3-DM has a "yes" answer if and only if the observation scheduling problem has.

First, consider an instance of 3-DM for which a solution exists. Then, the corresponding instance of the observation scheduling problem admits a solution with a total gain of $3n$: each set S_j corresponds to night j, the first observation is scheduled from 0 to $4B + x_{j_1}$, the second one from $4B + x_{j_1}$ to $6B + x_{j_1} + y_{j_2}$ and the third one from $6B + x_{j_1} + y_{j_2}$ to $7B + x_{j_1} + y_{j_2} + z_{j_3} = 8B$, each respecting its interval, and the total gain is $3n$ as all the observations are scheduled.

Conversely, assume that the observation scheduling problem admits a schedule of total gain $3n$. Such a value is obtained if and only if all the observations are scheduled. As $p_i^X > 4B$, exactly one observation $i \in \mathcal{T}^X$ must be scheduled first each night. The time remaining each night for the other observations is less than $4B$ and, as $p_i^Y > 2B$, exactly one observation $i \in \mathcal{T}^Y$ must be scheduled each night. The time remaining each night is less than $2B$ and, as $p_i^Z > B$ exactly one observation $i \in \mathcal{T}^Z$ is scheduled last each night. The total duration of the observations is equal to the available duration of the n nights: $\sum_{i=1}^{n}(4B + x_i + 2B + y_i + B + z_i) = 7nB + \sum_{i=1}^{n}(x_i + y_i + z_i) = 8nB$. Consequently there is no idle time, and each night j is such that: $4B + x_{i_1} + 2B + y_{i_2} + B + z_{i_3} = 8B$, that is: $x_{i_1} + y_{i_2} + z_{i_3} = B$, leading to a solution for the instance of the 3-DM.

As a consequence, there exists a pseudo-polynomial transformation from 3-DM to the observation scheduling problem. As the former is NP-complete in the strong sense, so is the latter. □

4 Solution methods and upper bounds

Various direct formulations of the problem using mixed integer programming, constraint programming and even local search paradigms have been experimented before [Brauner *et al.*, 2015], but the results obtained are completely outperformed by approaches that take advantage of the fact that the single-night problem can be solved in pseudo-polynomial time. We therefore focus on exact and heuristic approaches that rely on the dynamic program proposed for the single-night case (Theorem 2).

4.1 A column generation procedure

The single-night case is not too hard (see Theorem 2), which allows for a column generation procedure: the set of all possible schedules for every night is considered and a MIP selects an optimal subset of them. The linear relaxation of this MIP is solved iteratively by adding the column (feasible schedule) with maximum reduced cost, which can be computed using the dynamic program for the single-night case.

Let Ω_j be the set of all possible schedules for night j. The k-th schedule for night j is described by a 0/1 vector $(s_{1,j}^k, \ldots, s_{n,j}^k)$, where $s_{i,j}^k$ is equal to 1 if and only if observation i belongs to the k-th schedule of night j, and the weight of this schedule is equal to $w_j^k = \sum_{i \in \mathcal{T}} w_i s_{i,j}^k$.

Let ρ_j^k be a binary variable indicating whether the k-th schedule is used for night j. An extended formulation of the observation scheduling problem can be stated as follows:

$$
\begin{array}{lrcll}
\max \sum_{j \in \mathcal{M}} \sum_{k \in \Omega_j} w_j^k \rho_j^k & & & & \\
(1) & \sum_{k \in \Omega_j} \rho_j^k & = & 1 & \forall j \in \mathcal{M} \\
(2) & \sum_{j \in \mathcal{M}} \sum_{k \in \Omega_j} s_{i,j}^k \rho_j^k & \leq & 1 & \forall i \in \mathcal{T} \\
& \rho_j^k \in \{0, 1\} & & & \forall j \in \mathcal{M}, \forall k \in \Omega_j
\end{array}
$$

The sets Ω_j are too large to be enumerated completely, thus the linear relaxation of this model is solved with a column generation procedure. The pricing problem is to identify a variable ρ_j^k to enter the basis. Let α_j and β_i denote the dual variables of respectively constraints (1) and (2). The reduced cost of variable ρ_j^k is defined as:

$$r_j^k = w_j^k - \alpha_j - \sum_{i \in \mathcal{T}} s_{i,j}^k \beta_i.$$

So the pricing problem is to identify a valid schedule $(s_{i,j}^k)$ that maximizes $\sum_{i \in \mathcal{T}} s_{i,j}^k (w_i - \beta_i) - \alpha_j$. Such a column of maximum reduced cost can be found with the dynamic program (Theorem 2) by replacing w_i with $(w_i - \beta_i)$. The sets Ω_j are thus iteratively increased as long as a positive reduced cost schedule can be found. When dropping the constraints that ρ_j^k are integers, the procedure yields an upper bound.

4.2 A branch-and-price algorithm

The problem can be formulated as a constraint programming model:

$$
\begin{array}{lll}
\max z = \sum_{i \in \mathcal{T}} w_i z_i & & \\
(1) \quad \sum_{j \in \mathcal{M}} z_i^j = z_i & & i \in \mathcal{T} \\
(2) \quad \textsc{NightDisjunctive}([z_1^j, \ldots, z_{|\mathcal{T}|}^j]) & & j \in \mathcal{M} \\
(3) \quad \sum_{i \in C_k^j} z_i^j \leq 1 & & j \in \mathcal{M} \\
 & & C_k^j \in C^j \\
(4) \quad \textsc{Objective}(z, [z_1^1, \ldots, z_{|\mathcal{T}|}^{|\mathcal{M}|}]) & &
\end{array}
$$

The binary decision variables z_i and z_i^j denote respectively whether target i is scheduled, and if it is scheduled during night j. The starting dates of the observations are not explicitly represented as the model ensures that a feasible schedule for each night can be obtained by scheduling targets as early as possible in the order of their meridians. This is ensured by constraints (2): NIGHTDISJUNCTIVE states that the targets chosen for a night j can be scheduled without overlapping. This disjunctive scheduling problem is easy to solve due to the known ordering of the targets: checking whether a set of targets can be scheduled can be done using a greedy algorithm starting each target as early as possible. Algorithm 1 is thus used as a necessary and sufficient condition for the constraint to be satisfied when applied to the mandatory targets of night j, $M_j = \{i | z_i^j = 1\}$. It is also used for each remaining ungrounded z_a^j by checking if $M_j \cup \{a\}$ can be scheduled and, if not, value 1 is filtered from the domain of z_a^j; this enforces generalized arc-consistency in $O(|\mathcal{T}|^2)$. This can be seen has a specific case of the unary resource or disjunctive global constraint with optional tasks when the order of the tasks is known.

Algorithm 1 returns whether the set $\mathcal{S}$ of targets can be scheduled in night j

Require: Targets $\mathcal{S}$ sorted by non-decreasing meridians
1: $ctime \leftarrow 0$
2: **for** each target $i \in S$ **do**
3: $\quad ctime \leftarrow \max(ctime, r_i^j) + p_i^j$
4: $\quad$ **if** $ctime > d_i^j$ **then**
5: $\quad\quad$ **return** false;
6: **return** true

Constraints (3) are a relaxation of the NIGHTDISJUNCTIVE constraints that aims at speeding up the filtering of targets that cannot be scheduled a given night. A simple example would be that two targets i_1 and i_2 cannot be scheduled in a same night j when their total durations exceeds the available time; that is when $p_{i_1}^j + p_{i_2}^j > \max(d_{i_1}^j, d_{i_2}^j) - \min(r_{i_1}^j, r_{i_2}^j)$. The two targets are then said to be incompatible. The incompatibility graph of night j is defined with a vertex for each target and an edge for each pair of incompatible targets. A clique C_k^j in this graph can provide a constraint stating that at most one of the targets of the clique can fit in night j. The C_k^j are obtained by computing maximal cliques in the incompatibility graph of night j; and C^j simply denotes a set of such maximal cliques.

Constraint (4) considers all variables of the problem and encapsulates the column generation procedure to propagate an upper bound of z. It can be seen as a redundant constraint stating the objective function and the key features for implementing it are:

- The dynamic program is modified to handle the mandatory/forbidden targets for a night. The domains of the variables are thus taken into account in the sub-problem of the column-generation and do not affect the master.
- All generated columns are kept and the ones compatible with the current domains are added to the master to initialize it whenever the filtering algorithm is called to update the upper bound.
- When the relaxation turns out to be integer (ρ_j^k end up being integers), the z_i^j are grounded to the related values.
- Finally the z_i^j are filtered using the optimal reduced costs. Reduced cost based filtering is a traditional filtering technique that can be applied from an optimal solution of the linear relaxation. It can possibly fix to 0 some binary (0/1) variables remaining out of the basis. This technique can also be applied with an extended formulation by taking advantage of the dynamic program.

Search strategy

The search is directed by the global relaxation provided by the column generation procedure. It is performed in two steps that can be seen as *best first* and *first fail* types of strategies.

First, a good feasible solution is identified. The branching is performed on a variable z_i^j that maximizes $\sum_{k \in \Omega_j^*} \rho_j^k s_{i,j}^k$, setting value 1 first. In other words, we set to 1 the targets that seem to be favored by the relaxation. This step is backtrack free and the search simply dives to a feasible integer solution.

The search is then restarted from scratch and the branching is performed on a variable z_i^j that minimizes $|0.5 - \sum_{k \in \Omega_j^*} \rho_j^k s_{i,j}^k|$ and setting value 1 first. In other words, we select the mostly undecided variables of the relaxation, hopping to decrease the upper bound as much as possible *to fail* as soon as possible.

4.3 A local search approach

We implemented a local search approach to serve as a baseline. The search space is represented by the possible assignments of the targets to the nights, with an additional fake night for unscheduled targets. The following neighborhood is used for any such assignment (a "night" might be the fake one):

- **shift**: move a target from one night to another.
- **swap**: swap two targets between two nights.
- **reschedule**: considering all targets of a given real night j and those of the fake night, reschedule night j with a set of maximum value (using the dynamic program of Theorem 2).

A move is valid if it leads to a feasible assignment (all nights can be scheduled) and does not degrade the objective function. The moves are explored in a random order and the first valid move found is performed. So the search is performed in the space of feasible assignments only, and can iterate over plateaus of configurations with the same objective value. A simple tabu mechanism is added. This approach can be seen as a large neighborhood search due to the **reschedule** move which is the core component.

5 Experimentations

Since we only have one real dataset, we generated realistic random datasets. We choose $|\mathcal{T}| \in \{400, 600, 800, 1000\}$ and $|\mathcal{M}| \in \{71, 107, 142\}$, respecting the proportion of the real dataset between the number of targets and the number

	SIZE		CG			B&P					MIP	LS		SPOT	
ins	$\|\mathcal{T}\|$	$\|\mathcal{M}\|$	RUB	GAP	CPU	NSOL	FCPU	LB	O	CPU	GAP-2h	GAP-2m	GAP-2h	GAP-2m	GAP-2h
1	400	71	9810	0%	12	1	46	9810	Y	49	11.31%	0.92%	0.41%	3.26%	2.24%
2	400	71	9420	0%	12	1	42	9420	Y	44	10.93%	0.74%	0.32%	2.97%	2.23%
3	400	71	9920	0%	9	1	36	9920	Y	38	7.96%	0.40%	0.10%	2.42%	2.22%
4	400	71	9300	0%	7	1	31	9300	Y	34	10.65%	0.75%	0.54%	3.23%	1.61%
5	400	71	9260	0%	15	1	44	9260	Y	47	9.50%	0.86%	0.65%	3.56%	2.16%
6	600	107	14340	0%	35	1	149	14340	Y	161	-	1.26%	0.56%	16.95%	3.14%
7	600	107	14120	0%	39	2	157	14120	Y	312	-	1.49%	0.85%	16.22%	3.12%
8	600	107	14740	0%	29	1	140	14740	Y	150	-	1.22%	0.54%	15.74%	2.51%
9	600	107	14020	0%	37	1	153	14020	Y	164	-	0.93%	0.57%	15.76%	2.43%
10	600	107	13600	0%	46	1	200	13600	Y	212	-	1.47%	1.10%	17.06%	2.87%
11	800	142	19450	0%	71	1	412	19450	Y	447	-	0.93%	0.51%	18.05%	2.57%
12	800	142	19630	0%	188	1	683	19630	Y	724	-	1.43%	0.92%	17.07%	2.85%
13	800	142	19940	0%	174	1	547	19940	Y	584	-	1.00%	0.45%	19.21%	2.21%
14	800	142	19110	0%	138	1	582	19110	Y	619	-	1.10%	0.58%	18.79%	2.15%
15	800	142	18770	0.11%	176	1	637	18750	N	7200	-	1.12%	0.59%	17.31%	2.66%
16	1000	142	23410	0%	205	2	786	23410	Y	2084	-	2.22%	0.94%	29.05%	4.70%
17	1000	142	23390	0%	342	1	1025	23390	Y	1092	-	1.84%	1.15%	29.24%	4.10%
18	1000	142	23710	0.04%	280	1	856	23700	N	7200	-	2.07%	1.22%	29.23%	4.85%
19	1000	142	23020	0.04%	228	1	881	23010	N	7200	-	2.13%	1.00%	29.06%	4.47%
20	1000	142	22710	0%	226	2	871	22710	Y	2338	-	2.03%	1.19%	28.40%	4.54%
real	800	142	18620	0%	85	1	364	18620	Y	413	-	1.02%	0.16%	22.40%	3.65%

Figure 1: Computational results

of nights (datasets 1–15) or increasing the number of targets with the same number of nights (datasets 16–20). For all datasets, the horizon is equal to 640 (time is discretized in minutes).

The dataset generator tries to reproduce the real dataset. We first generate random targets that can be observed some time during the year and we compute the visibility window of each target each day of a year. Then we generate a number of random nights of observation, batched into runs (consecutive days), as in the real dataset. As a consequence, the targets of two nights in the same run are almost the same, up to an offset of the visibility windows, due to the rotation of the earth around the sun.

The experiments were performed on an Intel Xeon E5-2440 v2 @ 1.9 GHz processor and 32 GB of RAM, and ran with a memory limit of 4 GB and a time limit of 2 hours.

Figure 1 provides, from left to right, the dataset name, the size of $\mathcal{T}$ and $\mathcal{M}$, and the computational result of each method described: the column generation (Section 4.1), the branch-and-price (Section 4.2), the MIP formulation (Section 2), the local search (Section 4.3) and the SPOT software (Section 1).

- Column generation: the upper bound rounded down to a multiple of 10 (RUB, w_i are multiples of 10), the gap between RUB and the best lower bound found (GAP) and the computation time in seconds (CPU).
- Branch-and-price: the number of solutions found (NSOL), the computation time to find the first solution (FCPU), the best lower bound (LB), yes (Y) or no (N) depending if optimality has been proved and the total computation time (CPU) in seconds.
- MIP: the gap between the best known lower bound and the solution obtained by the MIP in 2 hours, "-" if no feasible solution was found within the time limit (GAP-2h).
- Local search: the gap between the best known lower bound and the solution obtained with local search in 2 minutes (GAP-2m), and the same gap with the solution obtained after 2 hours (GAP-2h).
- SPOT: the gap between the best known lower bound and the solution obtained with SPOT in 2 minutes (GAP-2m), and the same gap with the solution obtained after 2 hours (GAP-2h).

The optimal solution is found and proved by the branch-and-price for 18 out of 21 datasets. For the 3 remaining, the gap with the upper bound computed by column generation is smaller than 0.11%. The best lower bound is always provided by the branch-and-price (the gap is thus not given for this approach). The proof of optimality is achieved when the bound is initially optimum. The heuristic guided by the linear relaxation dives directly to an optimal integer solution in all cases except three (instances 7, 16, 20).

The local search is also relevant because it finds good solutions (with a gap smaller than 2.22%) within only 2 minutes, whereas the first solution of the branch-and-price requires more computation time for the biggest datasets (with $|\mathcal{T}| \geq 800$). In comparison, on the real dataset, the MIP model (section 2) does not found any solution and SPOT provides a solution of objective 17940 (*i.e* a gap of 3.65%) using 2 hours of computation time.

6 Conclusion

In this paper, we have studied a parallel machine problem with mandatory instants. We proved the NP-completeness of the problem and proposed a column-generation, a branch-and-price and a local-search procedures to solve it. Experimental results show these methods strongly dominates an out-of-the-box MIP and a tailored dedicated heuristic. Indeed, optimal (or very near optimal) solutions can be obtained in a very short time (2 hours), with regard to the planning horizons (several months).

Acknowledgments

The authors are grateful to Benoit Cance and Alexandre Le Jean for preliminary studies and their work on the datasets.

We also thank all the students involved in preliminary study of this problem: Cyril Dutrieux, Ilyès Mezghani, Kimia Nadjahi and Lucie Pansart. This work received a grant from the LabEx PERSYVAL-Lab (ANR–11-LABX-0025).

References

[Arkin and Silverberg, 1987] Esther M. Arkin and Ellen B. Silverberg. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18(1):1–8, 1987.

[Baptiste, 1999] Philippe Baptiste. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. *Journal of Scheduling*, 2(6):245–252, 1999.

[Bouzina and Emmons, 1996] Khalid I. Bouzina and Hamilton Emmons. Interval scheduling on identical machines. *Journal of Global Optimization*, 9(3-4):379–393, 1996.

[Brauner *et al.*, 2015] Nadia Brauner, Hadrien Cambazard, Benoit Cance, Nicolas Catusse, Pierre Lemaire, Anne-Marie Lagrange, Bernard Penz, and Pascal Rubini. Star Scheduling. In *MAPSP 2015, 12th Workshop on Models and Algorithms for Planning and Scheduling Problems*, La Roche-en-Ardenne, Belgium, June 2015.

[Carlisle and Lloyd, 1995] Martin C. Carlisle and Errol L. Lloyd. On the k-coloring of intervals. *Discrete Applied Mathematics*, 59(3):225–235, 1995.

[Chuzhoy *et al.*, 2006] Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research*, 31(4):730–738, 2006.

[Faigle and Nawijn, 1995] Ulrich Faigle and Willem M. Nawijn. Note on scheduling intervals on-line. *Discrete Applied Mathematics*, 58(1):13–17, 1995.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of books in the mathematical sciences. W. H. Freeman, 1979.

[Johnston and Miller, 1994] Mark D. Johnston and Glenn E. Miller. Spike: Intelligent scheduling of hubble space telescope observations. In *Intelligent Scheduling*, pages 391–422. Morgan Kaufmann Publishers, 1994.

[Keil, 1992] J. Mark Keil. On the complexity of scheduling tasks with discrete starting times. *Operations Research Letters*, 12(5):293 – 295, 1992.

[Kolen *et al.*, 2007] Antoon W. J. Kolen, Jan Karel Lenstra, Christos H. Papadimitriou, and Frits C.R. Spieksma. Interval scheduling: A survey. *Naval Research Logistics (NRL)*, 54(5):530–543, 2007.

[Lagrange *et al.*, 2015] Anne-Marie Lagrange, Pascal Rubini, Nadia Brauner-Vettier, Hadrien Cambazard, Nicolas Catusse, Pierre Lemaire, and Laurence Baude. Spot: a scheduler for adaptive optics, high contrast imaging surveys. In *In the spirit of Bernard Lyot 2015, Direct Detection of Exoplanets and Circumstellar Disks, Montréal, Canada*, 2015.

[Nakajima and Hakimi, 1982] Kazuo Nakajima and S. Louis Hakimi. Complexity results for scheduling tasks with discrete starting times. *Journal of Algorithms*, 3(4):344 – 361, 1982.

[Pinedo, 2014] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer New York, 2014.

[Spieksma, 1999] Frits C. R. Spieksma. On the approximability of an interval scheduling problem. *Journal of Scheduling*, 2(5):215–227, 1999.