Scheduling Running Modes of Satellite Instruments Using Constraint-Based Local Search

Cédric Pralet[1], Solange Lemai-Chenevier[2], and Jean Jaubert[2]

[1] ONERA – The French Aerospace Lab, F-31055, Toulouse, France
{firstname.lastname}@onera.fr
[2] CNES Toulouse, France
{firstname.lastname}@cnes.fr

Abstract. In this paper, we consider a problem involved in the management of Earth observation satellites. The input of the problem is a time-stamped observation plan and the output is a schedule of transitions between running modes of instruments available on board. These transitions must be scheduled so as to effectively realize the acquisitions requested, while satisfying several constraints, including duration constraints, non-overlapping constraints, and resource constraints over the thermal consumption of instruments. Criteria such as the minimization of the number of times instruments are switched off are also considered, for long-term reliability issues. The scheduling problem obtained needs to be solved several times per day, and the requirement is to produce good quality solutions in a few seconds. To produce such solutions, we propose a specific constraint-based local search procedure. Experiments are performed on realistic scenarios involving hundreds of observations, and the approach is compared with other techniques.

1 Problem Description

The study described in this paper is performed in relation with a future space program involving an observation satellite orbiting around the Earth at a low-altitude orbit (several hundreds of kilometers). The task of this satellite is to realize acquisitions over specific Earth areas using three optical instruments $I1$, $I2$, $I3$ available on board, and to download collected data towards ground reception stations. For such a satellite, each orbit takes about one hour and a half, and different areas are overflied during successive orbits due to the rotation of the Earth on itself. The set of areas selected for observation is defined by a ground mission center, which produces so-called *acquisition plans* for the satellite. Each acquisition in these plans is defined by its start and end times, by a set of instruments to be used, and by *instrument configurations*, which correspond to specific settings regarding instrument signal processing.

In addition to acquisition plans, the ground mission center must also provide the satellite with low-level plans specifying in more detail the operations to execute in order to make instruments ready for acquisition when requested. These low-level plans must define, for each instrument, the successive transitions to

make between several possible running modes: the (O)ff mode, the (S)tandby mode, the (H)eat mode, and the (A)cquisition mode. The possible transitions between these modes are described in Fig. 1a. To effectively realize an acquisition a over time interval $[t_1, t_2]$, each instrument required for a must be in the acquisition mode over $[t_1, t_2]$. For this, low-level plans must specify when instruments must perform a transition from the off mode to the standby mode, when the right instrument configuration must be loaded, when a transition to the heat mode must be performed to warm up the instrument, and when a transition to the acquisition mode must be triggered to have the instrument ready at t_1. Between two acquisitions, the instrument may come back to the heat, standby, or off mode depending on the time and resources available. Each transition will then result in a sequence of even more basic commands executed on board.

In this paper, we consider this problem of scheduling on the ground the transitions between instrument modes, from a time-stamped acquisition plan given as an input. In this problem, several kinds of constraints must be taken into account, including (1) *duration constraints*, enforcing a minimum duration spent in each mode and a fixed duration for each mode transition; (2) *non-overlapping constraints* over transitions; more precisely, transitions over instruments I1 and I2 cannot overlap except when they correspond to the same transition; the reason for this is that instruments I1 and I2 are managed on board by the same monitor, which is able to send a common sequence of basic commands to both I1 and I2, but not two distinct command sequences in parallel; similarly, for instrument monitoring reasons, it is not possible to perform a transition be-

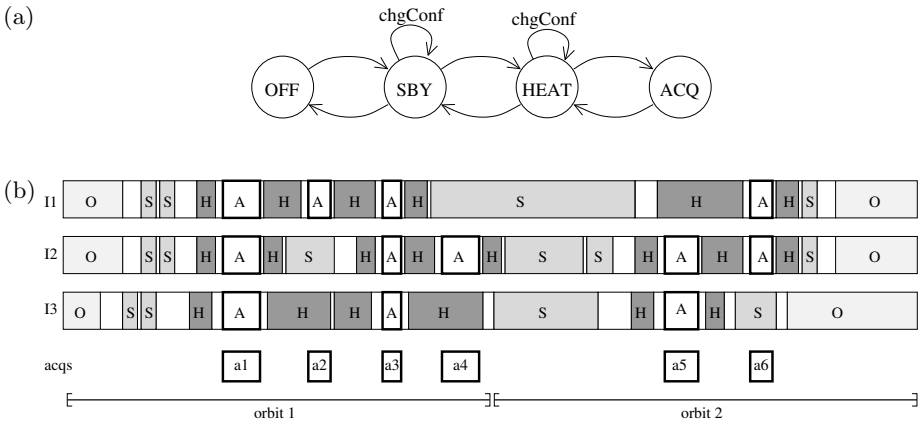

Fig. 1. (a) Mode automaton associated with each instrument; chgConf stands for a change of the configuration of the instrument; (b) example of a schedule defining mode transitions for instruments I1, I2, I3, given an acquisition plan involving 6 acquisitions a_1 to a_6 over 2 orbits; transitions correspond to empty rectangles between modes; transitions between the same two modes correspond to changes in instrument configuration; an acquisition such as a_1 uses all instruments, while a_4 uses only I2.

tween off and standby over I3 in parallel to a transition between off and standby over I1 or I2; moreover, no transition must be performed during acquisitions to avoid disturbances; (3) *resource constraints*, imposing that the total thermal consumption induced by the activities performed on each instrument over each orbit cannot exceed a maximum limit. In terms of optimization criterion, the main objective is to limit the number of times instruments are switched off, for long-term reliability issues. For a given number of times instruments are switched off, a secondary objective is to minimize the total thermal consumption of the instruments (better resource usage). In the end, the goal is to produce schedules such as the one provided in Fig. 1b.

This mode transition scheduling activity is the last step of the complete mission programming process which may be done several times a day, depending on the station opportunities to upload a new plan to the satellite. It is required to be solved in less than a few seconds. Constraint Programming (CP) comes into play because the realization of consistent mode transition schedules becomes more and more challenging for space engineers, due to the increasing complexity of satellites. One expected benefit in using CP is a better exploration of the search space, at least better than with hard-coded decision rules. Another advantage is that in first design phases as the one we are in, the declarative nature of CP is more flexible in case of changes in the satellite specifications, and it allows to easily assess the impact of new design choices.

The paper is organized as follows. We first give a formal model of the problem (Sect. 2). We then present a specific constraint-based local search encoding (Sect. 3) and a dedicated local search procedure (Sect. 4). Finally, we describe experiments on realistic scenarios involving hundreds of acquisitions (Sect. 5).

2 Problem Modeling

Data The set of possible running modes of the instruments is $\mathcal{M} = \{O, S, H, A\}$: O for Off, S for Standby, H for Heat, A for Acquisition. The set of possible transitions between these modes is $\mathcal{T} = \{OS, SO, SH, HS, HA, AH, SS, HH\}$. In this set, mm' stands for a transition from mode m to mode m', transitions mm being associated with instrument configuration loading. We also define as $\mathcal{S} = \mathcal{M} \cup \mathcal{T}$ the set of possible states of each instrument at each time. We then introduce several input data:

- a set of contiguous orbits $\mathcal{O}$; each orbit $o \in \mathcal{O}$ has a start time $\mathbf{TsOrb}_o$ and an end time $\mathbf{TeOrb}_o$; the planning horizon considered is $[\mathbf{Ts}, \mathbf{Te}]$ with $\mathbf{Ts}$ the start time of the first orbit and $\mathbf{Te}$ the end time of the last orbit;
- a set of instruments $\mathcal{I}$; for each instrument $i \in \mathcal{I}$ and each mode $m \in \mathcal{M}$, $\mathbf{DuMin}_{i,m}$ denotes a minimum duration for i in mode m, and for each transition of type $t \in \mathcal{T}$, $\mathbf{Du}_{i,t}$ denotes the fixed duration of transition t for instrument i; a thermal consumption rate $\mathbf{ThRate}_{i,s}$ is associated with each state $s \in \mathcal{S}$ of instrument i; last, $\mathbf{ThMax}_i$ denotes a maximal thermal consumption allowed for instrument i over each orbit;

- a set of acquisitions $\mathcal{A}$; each acquisition $a \in \mathcal{A}$ has a fixed start time $\mathbf{TsAcq}_a$ and a fixed duration $\mathbf{DuAcq}_a$; for each instrument $i \in \mathcal{I}$, $\mathbf{Conf}_{a,i}$ denotes the configuration required by acquisition a on instrument i (value *nil* when instrument i is not used in the realization of a); without loss of generality, we assume that each instrument is used at least once in the acquisition plan;
- a set of incompatibilities **Inc** between mode transitions; each element in **Inc** is a 4-tuple $(i, i', t, t') \in \mathcal{I}^2 \times \mathcal{T}^2$ specifying that transition t on instrument i cannot be performed in parallel to transition t' on instrument i'; for instance, $(I1, I2, SH, OS) \in \mathbf{Inc}$ because transition SH over $I1$ cannot be performed in parallel to transition OS over $I2$.

For each instrument $i \in \mathcal{I}$, several additional data can be derived, such as the number of acquisitions $\mathbf{Nacqs}_i$ for which i is used. For every $u \in [1..\mathbf{Nacqs}_i]$, $\mathbf{Acq}_{i,u} \in \mathcal{A}$ denotes the acquisition associated with the uth use of instrument i.

Allowed state sequences In order to produce plans such as the one given in Fig. 1b, it is necessary to determine, for each instrument, the sequence of its contiguous states over planning horizon $[\mathbf{Ts}, \mathbf{Te}]$, as well as the exact start and end times associated with these states. As the desired state sequences must contain the acquisitions, the main question is actually how to handle each instrument outside these acquisitions, that is (1) from the beginning of the planning horizon, where instruments are switched off, to the first acquisition, (2) between two successive acquisitions which use the same instrument configuration, (3) between two successive acquisitions which use distinct configurations, and (4) from the last acquisition to the end of the horizon, where instruments must all be switched off for safety reasons. These four types of global transitions between successive uses of instruments must be composed of state sequences that are consistent with the automaton given in Fig. 1a.

To limit the search space, one key point is to avoid considering state sequences such as [**A**,*AH*,*H*,*HS*,*S*,**SH,H,HS**,*S*,*SH*,*H*,*HA*,**A**], in which the instrument is warmed up and cooled down (part **SH**,**H**,**HS**) in the middle of two acquisitions. To do this, we define beforehand a set of allowed state sequences for each global transition between instrument uses. The decompositions allowed are provided in Fig. 2. It can be seen that from the beginning of the horizon to the first acquisition, there are only two relevant state sequences, performing the configuration loading operation in the standby mode and heat mode respectively. For global transitions between acquisitions which use the same instrument configuration, there are four relevant decompositions: one which comes back to the heat mode between the acquisitions, one which comes back to the standby mode, and two which come back to the off mode and differ in the place where the configuration is loaded (the loaded configuration is lost when using the off mode). Similarly, there are six possible decompositions for global transitions between acquisitions which use distinct configurations. For the last global transition, there is a unique possible decomposition.

Decision variables In the following, we define a *state interval* as a temporal interval over which a unique state is active. Formally, a state interval *itv* is

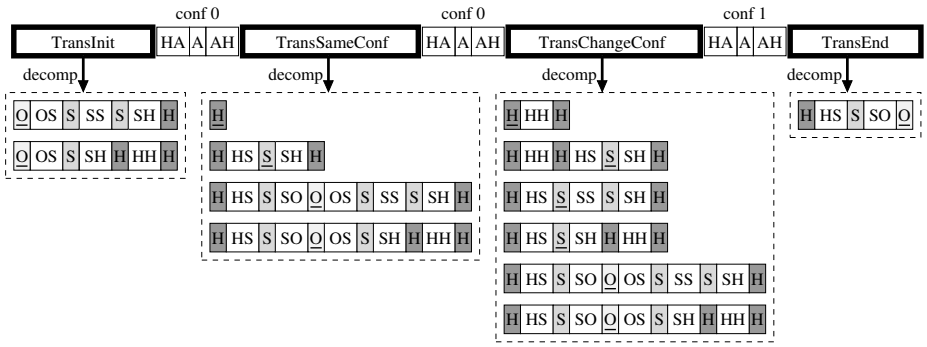

Fig. 2. Global transitions between instrument uses and their possible decompositions

defined by a boolean presence variable $pres(itv)$, by an integer start time variable $start(itv)$, by an integer end time variable $end(itv)$, by a fixed state $\mathbf{S}(itv) \in \mathcal{S}$, and by a fixed instrument $\mathbf{I}(itv)$ over which it is placed. In case of acquisition intervals, $\mathbf{I}(itv)$ contains all instruments required for the acquisition. We consider integer dates because the equipments which send mode transition commands to the instruments work with a discrete time clock. We also consider *global transition intervals* $gitv$, which are defined by an integer start time variable $start(gitv)$, by an integer end time variable $end(gitv)$, and by a fixed type among $\{TransInit, TransSameConf, TransChangeConf, TransEnd\}$.

Then, to describe schedules at a global level (upper part in Fig. 2), we introduce the following elements:

- for each acquisition $a \in \mathcal{A}$, one state interval $\mathbf{itvAcq}_a$ of state A is introduced for representing the temporal interval during which a is realized;
- for each instrument i and each use $u \in [1..\mathbf{Nacqs}_i]$ of this instrument, two mandatory state intervals $\mathbf{itvHA}_{i,u}$ and $\mathbf{itvAH}_{i,u}$ of respective states HA and AH are introduced for representing the transitions between the heat mode and the acquisition mode just before and just after acquisition $\mathbf{Acq}_{i,u}$;
- for each instrument i and each use $u \in [1..\mathbf{Nacqs}_i+1]$ of this instrument, one global transition interval $\mathbf{itvTrans}_{i,u}$ is defined for representing the global transition performed before the uth use of the instrument (use $\mathbf{Nacqs}_i + 1$ corresponds to the last global transition before the end of the horizon);
- for each instrument i, if $[a_1, \ldots, a_n]$ denotes the sequence of acquisitions performed by i, the sequence of intervals associated with i is defined by:

$$\mathbf{Seq}_i : [\mathbf{itvTrans}_{i,1}, \mathbf{itvHA}_{i,1}, \mathbf{itvAcq}_{a_1}, \mathbf{itvAH}_{i,1}, \ldots \mathbf{itvTrans}_{i,n}, \mathbf{itvHA}_{i,n}, \mathbf{itvAcq}_{a_n}, \mathbf{itvAH}_{i,n}, \mathbf{itvTrans}_{i,n+1}].$$

To describe schedules at a finer level (lower part in Fig. 2), we introduce the following elements:

- for each instrument i and each use $u \in [1..\mathbf{Nacqs}_i + 1]$ of i, one integer decision variable $\mathbf{decomp}_{i,u} \in [1..\mathbf{Ndec}_{i,u}]$ is introduced for representing the

index of the decomposition chosen for realizing global transition $\mathbf{itvTrans}_{i,u}$, among those given in Fig. 2; parameter $\mathbf{Ndec}_{i,u}$ stands for the number of possible decompositions of $\mathbf{itvTrans}_{i,u}$;
- for each instrument i, for each use $u \in [1..\mathbf{Nacqs}_i + 1]$ of i, and for each decomposition $d \in [1..\mathbf{Ndec}_{i,u}]$ of $\mathbf{itvTrans}_{i,u}$, one sequence of state intervals $\mathbf{Dec}_{i,u,d} = [itv_1, \ldots, itv_k]$ is introduced; the latter is determined following the decomposition schemes provided in Fig. 2; state intervals in sequence $\mathbf{Dec}_{i,u,d}$ are optional (they will be present only in case $\mathbf{decomp}_{i,u} = d$).

The set of state intervals defined in the model is then $Itv = \{\mathbf{itvAcq}_a \,|\, a \in \mathcal{A}\} \cup \{\mathbf{itvHA}_{i,u} \,|\, i \in \mathcal{I}, u \in [1..\mathbf{Nacqs}_i]\} \cup \{\mathbf{itvAH}_{i,u} \,|\, i \in \mathcal{I}, u \in [1..\mathbf{Nacqs}_i]\} \cup \{itv \in \mathbf{Dec}_{i,u,d} \,|\, i \in \mathcal{I}, u \in [1..\mathbf{Nacqs}_i+1], d \in [1..\mathbf{Ndec}_{i,u}]\}$. Each interval itv in this set has a minimum start time $Min(itv)$ and a maximum end time $Max(itv)$, both obtained from the known dates of acquisitions and from the minimum durations of state intervals. The set of intervals in Itv which may overlap orbit o and which concern instrument i is denoted by $ItvOrb_{i,o}$.

Constraints We now express the set of constraints holding over the integer variables, intervals, and sequences of intervals previously introduced. These constraints are given in Eq. 1 to 15. Each constraint introduced is active only when the intervals over which it holds are present. Constraint 1 expresses that all intervals involved in sequence $\mathbf{Seq}_i$ must be present. Constraints 2-3 express that the start and end times of this sequence must coincide with the start and end times of the planning horizon. Constraint 4 imposes that successive intervals belonging to sequence $\mathbf{Seq}_i$ must be contiguous (no period during which the state of the instrument is undefined). Constraint 5 expresses that for a given global transition, only state intervals involved in the decomposition chosen for this transition are present. Constraints 6-7 impose that the start and end times of the sequence of intervals associated with a decomposition must coincide with the start and end times of the global transition from which it is derived. Constraint 8 specifies that successive intervals belonging to sequence $\mathbf{Dec}_{i,u,d}$ must be contiguous (no period during which the state of the instrument is undefined). Constraints 9-10 define a minimum duration for state intervals labeled by a mode and a fixed duration for state intervals labeled by a transition. Constraints 11-12 express that acquisition intervals must start at the required start time and have the required duration. Constraint 13 imposes that no transition must occur during acquisitions. Constraint 14 expresses that incompatible transitions must not overlap. Constraint 15 imposes a maximum thermal consumption over each orbit of the satellite for each instrument. In this constraint, the left term of the inequality takes into account only the part of intervals which overlaps the orbit considered.

$\forall i \in \mathcal{I},$

$$\forall itv \in \mathbf{Seq}_i,\ pres(itv) = 1 \tag{1}$$

$$start(first(\mathbf{Seq}_i)) = \mathbf{Ts} \tag{2}$$

$$end(last(\mathbf{Seq}_i)) = \mathbf{Te} \tag{3}$$

$$\forall itv \in \mathbf{Seq}_i \,|\, itv \neq last(\mathbf{Seq}_i),\ end(itv) = start(next(itv, \mathbf{Seq}_i)) \tag{4}$$

$\forall i \in \mathcal{I}, \forall u \in [1..\mathbf{Nacqs}_i + 1], \forall d \in [1..\mathbf{Ndec}_{i,u}]$,

$$\forall itv \in \mathbf{Dec}_{i,u,d},\ pres(itv) = (\mathbf{decomp}_{i,u} = d) \quad (5)$$

$$start(first(\mathbf{Dec}_{i,u,d})) = start(\mathbf{itvTrans}_{i,u}) \quad (6)$$

$$end(last(\mathbf{Dec}_{i,u,d})) = end(\mathbf{itvTrans}_{i,u}) \quad (7)$$

$$\forall itv \in \mathbf{Dec}_{i,u,d} \,|\, itv \neq last(\mathbf{Dec}_{i,u,d}),\ end(itv) = start(next(itv, \mathbf{Dec}_{i,u,d})) \quad (8)$$

$$\forall itv \in Itv\ s.t.\ \mathbf{S}(itv) \in \mathcal{M},\ duration(itv) \geq \mathbf{DuMin}_{\mathbf{I}(itv),\mathbf{S}(itv)} \quad (9)$$

$$\forall itv \in Itv\ s.t.\ \mathbf{S}(itv) \in \mathcal{T},\ duration(itv) = \mathbf{Du}_{\mathbf{I}(itv),\mathbf{S}(itv)} \quad (10)$$

$\forall a \in \mathcal{A}$,

$$start(\mathbf{itvAcq}_a) = \mathbf{TsAcq}_a \quad (11)$$

$$duration(\mathbf{itvAcq}_a) = \mathbf{DuAcq}_a \quad (12)$$

$$\forall itv \in Itv \,|\, \mathbf{S}(itv) \in \mathcal{T},\ noOverlap(itv, \mathbf{itvAcq}_a) \quad (13)$$

$$\forall itv, itv' \in Itv \text{ s.t. } (\mathbf{I}(itv), \mathbf{I}(itv'), \mathbf{S}(itv), \mathbf{S}(itv')) \in \mathbf{Inc},\ noOverlap(itv, itv') \quad (14)$$

$$\forall i \in \mathcal{I}, \forall o \in \mathcal{O},\ \sum_{itv \in ItvOrb_{i,o}} pres(itv) \cdot d(itv) \cdot \mathbf{ThRate}_{i,\mathbf{S}(itv)} \leq \mathbf{ThMax}_i \quad (15)$$

$$\text{with } d(itv) = \begin{array}{l} 0 \text{ if } (end(itv) \leq \mathbf{TsOrb}_o) \vee (start(itv) \geq \mathbf{TeOrb}_o) \\ \min(end(itv), \mathbf{TeOrb}_o) - \max(start(itv), \mathbf{TsOrb}_o)) \text{ otherwise} \end{array}$$

Optimization criteria The first objective is to minimize the number of times instruments are switched off, for long-term reliability issues:

$$\text{minimize } \text{card}\{itv \in Itv \,|\, (\mathbf{S}(itv) = O) \wedge (pres(itv) = 1)\} \quad (16)$$

The second (and less important) criterion is to minimize the total thermal consumption over all instruments:

$$\text{minimize } \sum_{itv \in Itv} pres(itv) \cdot (end(itv) - start(itv)) \cdot \mathbf{ThRate}_{\mathbf{I}(itv),\mathbf{S}(itv)} \quad (17)$$

The two criteria defined are antagonistic because for instance the addition of off periods into the plan reduces the total thermal consumption. Criteria imposing a fair sharing between instruments could also be considered, as well as a minimization of the number of off intervals for a particular instrument.

Problem analysis The decision problem obtained is a constraint-based scheduling problem involving temporal constraints, resource constraints, and optional tasks [1]. It can be related with standard Job Shop Scheduling Problems (JSSP [2]). In our case, there would be one job per instrument, and the (optional) operations associated with each job would be the set of state intervals associated with each instrument. However, there are several differences with JSSP. For instance, some operations are shared between jobs, namely the acquisition tasks that require using several instruments simultaneously. Also, successive operations must be contiguous. Contiguity is important for ensuring that the state of the instrument is defined at every time and for getting the right thermal consumption.

This consumption depends on the duration of state intervals, which is not fixed for state intervals associated with modes.

Another significant feature of the problem is the presence of global transition tasks which can be decomposed in several possible ways. Such a hierarchical aspect is also found in the framework of Hierarchical Task Networks (HTN [3]) developed in the planning community. In this framework, some abstract tasks are specified and the goal is to decompose these tasks into basic sequences of actions, through a set of available decomposition methods. In CP, the possible decompositions of global transitions could be expressed using an *alternative constraint* [4]. In another direction, the set of allowed sequences of state intervals could be represented using a *regular constraint* [5].

Last, the criteria considered are not standard criteria such as the makespan or the tardiness. On this point, it is worth noting that some transitions should be scheduled as early as possible to reduce the resource consumption, whereas others should be scheduled as late as possible for the same reason (more details on these points later in the paper).

3 Specific Problem Encoding

To obtain an efficient approach, several difficulties must be overcome. We present these difficulties and we show how they are handled.

3.1 Problem Complexity Versus Allowed Running Times

The problem obtained could be modeled and solved using CP solvers such as IBM ILOG CpOptimizer, which contains efficient scheduling techniques. It could also be represented using Mixed Integer Programming [6], which would be particularly adapted to deal with linear expressions related to thermal aspects. However, we must be able to solve in a few seconds large instances, which can involve several hundreds of acquisitions generating several thousands or tens of thousands candidate state intervals.

To overcome this difficulty, we use Constraint-Based Local Search (CBLS [7]), which has the capacity to quickly produce good quality solutions for large-size problems. As in standard constraint programming, CBLS models are defined by decision variables, constraints, and criteria. One distinctive feature is that in CBLS, all decision variables are assigned when searching for a solution, *i.e.* the approach manipulates complete variable assignments. The search space is then explored by performing local moves which reassign some decision variables, and it is explored more freely than in tree search with backtracking. One specificity of CBLS models is that they manipulate so-called *invariants*. The latter are one-way constraints $x \leftarrow exp$ where x is a variable (or a set of variables) and exp is a functional expression of other variables of the problem, such as $x \leftarrow sum(i \in [1..N])\, y_i$. During local moves, these invariants are efficiently maintained thanks to specific procedures that incrementally reevaluate the output of invariants (left part) in case of changes in their inputs (right part).

For tackling the application, we use the InCELL CBLS engine [8]. The latter offers several generic building blocks, including a large catalog of CBLS invariants together with incremental reevaluation techniques, and generic elements for defining specific local search procedures. In InCELL, a temporal interval itv is represented by a presence variable $pres(itv)$, a start time-point $start(itv)$, and an end time-point $end(itv)$. For incrementally managing temporal constraints over intervals, InCELL uses the *Simple Temporal Network* invariant [9]. The latter takes as input an STN [10], which is a set of simple temporal constraints over time-points, and it maintains as an output the temporal consistency of this STN as well as, for each time-point t, two variables $earliestTime(t)$ and $latestTime(t)$ giving its earliest and latest temporal position in a consistent schedule respectively. In the following, given an interval itv, variables $earliestTime(start(itv))$, $latestTime(start(itv))$, $earliestTime(end(itv))$, and $latestTime(end(itv))$ are denoted more succinctly as $est(itv)$, $lst(itv)$, $eet(itv)$, and $let(itv)$.

3.2 Specific Manipulation of Topological Orders

To implement a specific local search procedure, we need to control the main decisions involved in the problem. These decisions first include the decompositions chosen for global transitions, which are directly represented by variables $\mathbf{decomp}_{i,u}$. They also include the ordering between state intervals which must not overlap. To represent such decisions, we explicitly manipulate in the CBLS model a *topological ordering* **topoOrder** over intervals involved in non-overlapping constraints (Constraints 13-14). Technically speaking, this topological ordering is a sequence which expresses that if two intervals itv, itv' must not overlap and if itv is placed before itv' in the sequence, denoted by $before(itv, itv', \mathbf{topoOrder}) = 1$, then $end(itv) \leq start(itv')$ must hold. Constraints $noOverlap(itv, itv')$ appearing in Eq. 13-14 are then reformulated as:

$$(before(itv, itv', \mathbf{topoOrder}) = 1) \rightarrow (end(itv) \leq start(itv')) \quad (18)$$

$$(before(itv', itv, \mathbf{topoOrder}) = 1) \rightarrow (end(itv') \leq start(itv)) \quad (19)$$

In the local search algorithm defined, some of the local moves are performed by updating directly the topological ordering of conflicting intervals. To obtain an efficient approach, it is crucial to avoid considering local moves which create precedence cycles between intervals, where for instance an interval itv_1 is requested to be placed both before and after another interval itv_2. For avoiding this, as in Job Shop Scheduling, we exploit a *mandatory precedence graph*. The latter captures all mandatory precedences over transitions and acquisitions, including: (1) precedences between successive transition intervals involved in the same sequence $\mathbf{Dec}_{i,u,d}$ (sequence associated with the dth possible decomposition of the global transition realized before the uth use of instrument i), and (2) precedences between the first (resp. last) transition intervals in $\mathbf{Dec}_{i,u,d}$ and the acquisition that precedes (resp. follows) $\mathbf{Dec}_{i,u,d}$. From this precedence graph, when a transition interval itv must be moved just after another interval

itv' in topological order **topoOrder**, all mandatory successors of itv placed before itv' in the order are also moved just after itv, to avoid the creation of a precedence cycle.

Also, for each instrument $i \in \mathcal{I}$ and each use of this instrument $u \in [1..\mathbf{Nacqs}_i]$, we merge intervals $\mathbf{itvHA}_{i,u}$ and $\mathbf{itvAcq}_{\mathbf{Acq}_{i,u}}$ in the topological order. Indeed, as transitions and acquisitions cannot overlap, a transition interval itv which is in conflict with $\mathbf{itvHA}_{i,u}$ either ends before $\mathbf{itvHA}_{i,u}$, or starts after $\mathbf{itvAcq}_{\mathbf{Acq}_{i,u}}$, hence there is no need to dissociate $\mathbf{itvHA}_{i,u}$ and $\mathbf{itvAcq}_{\mathbf{Acq}_{i,u}}$ in the order. For the same reason, intervals $\mathbf{itvAH}_{i,u}$ and $\mathbf{itvAcq}_{\mathbf{Acq}_{i,u}}$ are also merged.

3.3 Specific Encoding of the Thermal Resource Constraint

Once a decomposition is selected for each global transition ($\mathbf{decomp}_{i,u}$ variables) and once an ordering between conflicting intervals is chosen (**topoOrder** variable), the temporal consistency of the plan can be determined as well as the value of the first criterion (number of times instruments are switched off). The only remaining freedom concerns the choice of precise start and end times of all intervals in the model. There is usually a huge freedom in the choice of these precise times, because we must deal with a large possible set of time steps. To avoid loosing time exploring all precise time values and to detect thermal inconsistency earlier, there is a need to build an approximation of the impact of decisions made so far on the thermal resource constraint given in Eq. 15 and on the thermal consumption criterion given in Eq. 17.

To do this, we again use a specificity of the problem which is that in each decomposition $\mathbf{Dec}_{i,u,d}$, it is possible to identify one particular interval whose thermal consumption is the lowest, and therefore one particular interval which should be enlarged as much as possible. In Fig. 2, for each possible decomposition, this particular interval to enlarge is represented with an underlined label. From this observation, we build an approximation of the thermal constraint (Constraint 15) as follows: in Eq. 15, (1) if itv is an interval to enlarge, $start(itv)$ and $end(itv)$ are replaced by $est(itv)$ and $let(itv)$ respectively (start as early as possible, end as late as possible); (2) if itv is placed on the left of the interval to enlarge in a decomposition, then $start(itv)$ and $end(itv)$ are replaced by $est(itv)$ and $eet(itv)$ respectively (start and end as early as possible); (3) if itv is placed on the right of the interval to enlarge in a decomposition, then $start(itv)$ and $end(itv)$ are replaced by $lst(itv)$ and $let(itv)$ respectively (start and end as late as possible). The schedule obtained with these dates may not be feasible since it mixes earliest dates and latest dates, but the thermal consumption obtained over each orbit for each instrument can be shown to be a lower bound on the real consumption.[3] The same rewriting is used for building a lower approximation of the thermal criterion (Eq. 17).

[3] Without going into further details, this result holds under the assumption that the thermal consumption rate of the SS (resp. HH) state is equal to the thermal consumption of the S (resp. H) state, which is true in practice.

3.4 Specific Management of Some Hard Constraints

As almost always in local search, there is a choice between handling constraints as hard constraints, to be always satisfied by the current assignment, or as constraints returning a constraint violation degree. Here, we choose to make soft the constraints enforcing due dates (Constraints 3 and 11) and the thermal resource constraint (Constraint 15), and we keep the other constraints as hard. Such a choice is made because finding a first plan satisfying Constraints 3, 11, 15 is not straightforward, therefore we need to temporarily explore plans which violate these constraints to progressively get a first consistent solution. On the other side, for all other constraints, e.g. for all constraints enforcing contiguity between successive activities, it is easy to produce a first consistent schedule.

More precisely, we transform Constraint 11 $start(\mathbf{itvAcq}_a) = \mathbf{TsAcq}_a$ into one permanent constraint (Eq. 20), one constraint whose presence can be controlled based on an additional variable named $\mathbf{lock}_a$ (Eq. 21), and one constraint violation degree called $\mathbf{delay}_a$, defined as a CBLS invariant (Eq. 22). The latter represents the gap between the earliest start time of a and its required start time. Variable $\mathbf{lock}_a$ serves us to explore either consistent schedules or temporarily inconsistent schedules when needed. All **lock** variables will be set to 1 at the end of the search, meaning that all acquisitions will be realized on time.

$$start(\mathbf{itvAcq}_a) \geq \mathbf{TsAcq}_a \tag{20}$$

$$(\mathbf{lock}_a = 1) \rightarrow (start(\mathbf{itvAcq}_a) \leq \mathbf{TsAcq}_a) \tag{21}$$

$$\mathbf{delay}_a \leftarrow est(\mathbf{itvAcq}_a) - \mathbf{TsAcq}_a \tag{22}$$

The same transformation is used for Constraint 3, with a $\mathbf{lock}_{|\mathcal{A}|+1}$ variable and with a violation degree $\mathbf{delay}_{|\mathcal{A}|+1} \leftarrow \max_{i \in \mathcal{I}}(eet(last(\mathbf{Seq}_i)) - \mathbf{Te})$. When all **lock** variables are set to 0, provided that there is no precedence cycle, there always exists a schedule which satisfies all active temporal constraints, essentially because there is no maximum duration for off, standby, and heat intervals.

Last, given an instrument i and an orbit o, the violation degree $\mathbf{thv}_{i,o}$ associated with Constraint 15 is the maximum between 0 and the difference between the thermal consumption obtained and the maximum thermal consumption allowed:

$$\mathbf{thv}_{i,o} \leftarrow \max(0, \sum_{itv \in ItvOrb_{i,o}} pres(itv) \cdot d(itv) \cdot \mathbf{ThRate}_{i,\mathbf{S}(itv)} - \mathbf{ThMax}_i) \tag{23}$$

Fig. 3 gives a global view of the structure of the resulting CBLS model. The figure does not represent all individual invariants of the model.

4 A Multi-Phase Local Search Algorithm

We now describe how we actually solve the problem, using a specific local search procedure. In the latter, the first step is to perform some preprocessing to restrict the set of feasible decompositions of global transitions. This step is able

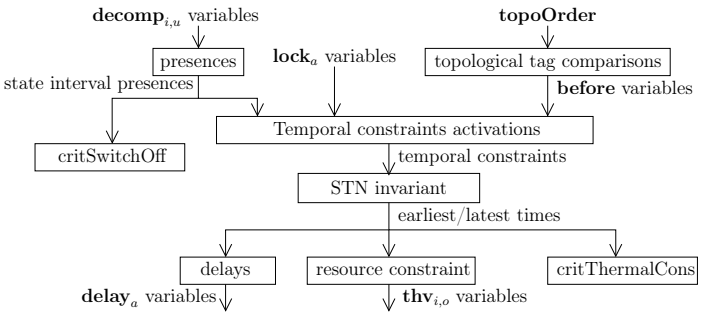

Fig. 3. Global view of the CBLS model

to automatically detect that for temporal reasons, the decomposition using the heat mode is the only feasible one between successive acquisitions placed very close to each other. It is also able to detect that for thermal reasons, only decompositions which come back to the off mode are feasible between successive acquisitions placed very far from each other.

After this preprocessing step, the main difficulty is that the problem involves antagonistic aspects, because for instance the best option regarding the on/off criterion is to never come back to the off mode between two successive acquisitions, whereas it is the worst option from a thermal point of view. This is why the resolution process is divided into several search phases (see Fig. 4).

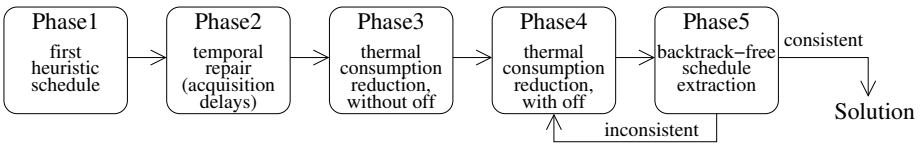

Fig. 4. Multi-phase local search algorithm

Phase 1: first heuristic schedule In the first phase, we build an initial schedule from simple decision rules. For decomposing global transitions, we systematically choose the option whose total minimum duration is the lowest. For choosing a topological order between intervals, we use a dispatching rule called the *best date heuristic*. In this heuristic, we compute, for each global transition and for each of its possible decompositions *dec*, the optimal start time of each transition interval *itv* involved in *dec* when non-overlapping constraints are discarded. For intervals *itv* placed after the interval to enlarge in *dec*, this optimal start time corresponds to the latest start time, denoted by $lstAlone(itv)$, and for the others it corresponds to the earliest start time, denoted by $estAlone(itv)$. To get a first schedule, transitions with the smallest best dates are inserted first into the topological ordering. During this phase, all **lock** variables are set to 0.

Phase 2: temporal repair The schedule obtained after Phase 1 may violate temporal constraints on the realization dates of acquisitions. To repair such constraint violations, we consider acquisitions chronologically. When considering an acquisition a, we try to enforce the realization date of a as long as the constraint violation degree associated with it is not equal to 0 (Eq. 22). To do this, we build the critical path explaining the late date obtained for a, we randomly select on this path a critical arc $itv \rightarrow itv'$ relating two intervals associated with distinct instruments, and we try to place itv just after itv' in the topological order. This local move is accepted iff it strictly reduces the delay of acquisition a. When all arcs in the critical path have been considered without improving the delay of a, the algorithm returns an inconsistency, meaning that the acquisition plan is too tight and should be revised. Otherwise, once the delay of acquisition a is equal to 0, variable $\mathbf{lock}_a$ is set to 1 before considering the next acquisition, meaning that future moves are not allowed to delay the realization of a. After the traversal of the acquisition plan, all acquisitions meet their due date. The same principle is used when the last state interval ends after the end of the planning horizon.

Phase 3: resource optimization During the third phase, we consider the satisfaction of the thermal consumption constraint, by working on possible decompositions of global transitions. More specifically, we traverse the schedule chronologically, from the first acquisition to the last. At each step, we consider an acquisition a and we lock the realization dates of all acquisitions except for a ($\mathbf{lock}_a = 0$). For each instrument involved in the realization of a, we try to change the decomposition of the global transition leading to a. Only decompositions which do not use additional switch off operations are considered, and decompositions which induce lower thermal consumption are considered first. For assessing the impact of changing the current decomposition used dec for another decomposition dec', we remove from the topological order all transition intervals belonging to dec, and we insert into it all transition intervals belonging to dec'. To do this, we use an insertion heuristic which works as follows: the transition intervals in dec' which must be scheduled as early as possible (resp. as late as possible) are inserted from the first to the last (resp. from the last to the first); the insertion of an interval itv is made just before the first (resp. after the last) interval itv' in the topological order such that $estAlone(itv) < start(itv')$ (resp. $lstAlone(itv) > start(itv')$) and such that the insertion does not lead to an inconsistency. Possible delays over the realization time of the acquisition considered are also repaired following the mechanism described in Phase 2. If delays cannot be repaired, decomposition dec' is rejected and we go on with the next possible decomposition. Otherwise, dec' is accepted as the new decomposition.

Phase 4: resource repair After the third phase, the schedule may still be inconsistent from a thermal point of view. In this case, it is repaired by testing decompositions which add new off intervals during orbits where the thermal resource constraint is violated. To do this, the plan is traversed chronologically, orbit by orbit, and we work on an orbit until the thermal constraint of the CBLS model is satisfied. For introducing off intervals, we consider first global transi-

tions which have the longest duration, because inserting off intervals during these global transitions is more likely to better reduce the total thermal consumption. For testing each decomposition, the same mechanism as in Phase 3 is used.

Phase 5: solution extraction In the fifth phase, we use a backtrack-free procedure for determining good start and end times for intervals. This procedure traverses the topological ordering and considers at each step the next interval itv in the order. It sets the start time of itv to its latest consistent value when itv appears on the right of the interval to enlarge in a decomposition, and to its earliest consistent value otherwise. After setting the start time of itv, the procedure propagates the effect of this decision in the temporal network before going on to the next step. The schedule obtained at the end of the traversal of the topological ordering is temporally consistent, essentially because of the *decomposability* property of Simple Temporal Networks [10]. If the schedule obtained satisfies thermal constraints over each orbit, it is returned as a solution and the algorithm stops. Otherwise, we come back to fourth phase to insert additional off intervals. If no more off interval can be added, an inconsistency is returned.

5 Experiments

Three approaches are compared: (1) the local search algorithm described previously, built on top of the InCELL CBLS library [8], (2) a MIP approach based on IBM ILOG CPLEX 12.5; in the MIP model, boolean variables $before(itv, itv')$ are introduced for every pair of intervals itv, itv' which must not overlap; (3) a CP approach based on IBM ILOG CpOptimizer 12.5; in the CP model, CpOptimizer constraints such as the *alternative constraint* or the *noOverlap* constraint are used. From the definition of the formal model (Eq. 1-17) and from the definition of data structures for representing the instances, obtaining a first implementation with CpOptimizer took us less than one day, obtaining a first implementation with CPLEX took us approximately one day, and obtaining the local search approach took us approximately one day for implementing the CBLS model and one week for specifying and testing specific local search procedures.

Experiments were performed over realistic instances provided by the French space agency, ranging from small instances involving 6 acquisitions to large instances involving more than 200 acquisitions (see Table 1). To give an idea of the problem sizes, the largest instance (sat8) leads to 61515 variables and 176842 constraints in the CpOptimizer model, 103428 rows and 62434 columns in the CPLEX model, and 831074 variables and 258224 invariants in the InCELL CBLS model. Globally, it can first be observed that the attempt made with CpOptimizer is not efficient. We believe that the behavior observed is due to the fact that the sum constraint involved in the computation of the total thermal consumption does not allow to prune earliest/latest dates of activities that much because of its duration-dependent nature. Another point is that the domain of temporal variables may contain millions of values. As for the MIP approach, it performs very well on small and medium instances, for which it is able to find

			CpOptimizer			CPLEX			Specific CBLS		
	#acqs	#orbits	time (s)	#offs	conso	time (s)	#offs	conso	time (s)	#offs	conso
sat1	6	1	120	6	1048	0.11	6*	1046.0*	0.43	6	1074
sat2	11	2	-	-	-	14.01	7*	774.38*	0.61	7	785.87
sat3	37	1	-	-	-	1.41	8*	2922.30*	0.95	8	2933.80
sat4	42	1	-	-	-	55.35	13*	3395.49*	0.95	13	3416.27
sat5	47	1	-	-	-	3.29	8*	2895.54*	1.10	8	2903.03
sat6	82	2	-	-	-	120	20	6413.07	1.76	19	6569.49
sat7	129	3	-	-	-	-	-	-	2.25	17	8768.01
sat8	233	10	-	-	-	-	-	-	3.86	39	17270.62

Table 1. Results obtained on an Intel i5-520 1.2GHz, 4GBRAM for 8 realistic instances, with a max. CPU time set to 2 minutes; *#orbits* gives the number of orbits which contain acquisitions; *#offs* stands for the number of times instruments are switched off and *conso* for the total thermal consumption; '*' indicates that the solution is optimal

optimal solutions.[4] However, for the largest instances, the MIP approach does not find any solution in less than two minutes. Last, the CBLS approach scales well and delivers good quality solutions in a few seconds, which is compatible with the application needs. On the largest instance (sat8), the distribution of computation times among local search phases is as follows: CBLS model creation: 2.5s, preprocessing step: 0.69s, Phase 1: 0.4s, Phase 2: 0.02s, Phase 3: 0.20s, Phase 4: 0.03s, Phase 5: 0.02s. Moreover, there is no come back to Phase 4 after Phase 5.

6 Conclusion

This paper presents a specific CBLS approach for tackling a scheduling problem from the space domain. This problem raises several questions for CP. Indeed, whereas scheduling is known to be one of the most successful application area of CP, one lesson learned is that it can be difficult for standard CP solvers to deal with scheduling problems where the duration of some tasks is not fixed initially and where the duration of these tasks has an impact on resource consumption. Also, we believe that alternative choices between task decompositions make the propagation of the resource constraint even harder. On the opposite, it is easy to enlarge the right state intervals for the specific local search procedure or for the MIP approach. Due to the good performances of MIP on small and medium instances, it would probably be useful either to run CBLS and MIP in parallel, or to use the MIP model for realizing Large Neighborhood Search (LNS [11]). In another direction, the treatment of this application will also make us add new elements in our CBLS solver, for representing abstract tasks decomposable into basic activities following a set of available decomposition methods.

[4] Without further details, we relax the requirement of having integer dates, essentially because an integer solution can almost often be directly extracted from the solution found by the MIP.

References

1. Baptiste, P., Pape, C.L., Nuijten, W.: Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems. Kluwer Academic Publishers (2001)
2. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Springer (2012)
3. Erol, K., Hendler, J., Nau, D.S.: HTN Planning: Complexity and Expressivity. In: Proc. of the 12th National Conference on Artificial Intelligence (AAAI-94). (1994) 1123–1128
4. Laborie, P., Rogerie, J.: Reasoning with Conditional Time-Intervals. In: FLAIRS Conference. (2008) 555–560
5. Pesant, G.: A Regular Language Membership Constraint for Finite Sequences of Variables. In: Proc. of the 10th International Conference on Principles and Practice of Constraint Programming (CP-04). (2004) 482–495
6. Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. John Wiley & Sons (1988)
7. Hentenryck, P.V., Michel, L.: Constraint-Based Local Search. The MIT Press (2005)
8. Pralet, C., Verfaillie, G.: Dynamic Online Planning and Scheduling Using a Static Invariant-based Evaluation Model. In: Proc. of the 23th International Conference on Automated Planning and Scheduling (ICAPS-13). (2013)
9. Pralet, C., Verfaillie, G.: Time-dependent Simple Temporal Networks: Properties and Algorithms. RAIRO Operations Research (2013)
10. Dechter, R., Meiri, I., Pearl, J.: Temporal Constraint Networks. Artificial Intelligence **49** (1991) 61–95
11. Shaw, P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: Proc. of the 4th International Conference on Principles and Practice of Constraint Programming (CP-98). (1998) 417–431