

# Gestion du vidage de données satellite avec incertitude sur les volumes

Cédric Pralet<sup>1</sup>    Adrien Maillard<sup>1</sup>    Gérard Verfaillie<sup>1</sup>  
Emmanuel Hébrard<sup>2</sup>    Nicolas Jozefowicz<sup>2</sup>    Marie-Jo. Huguet<sup>2</sup>  
Thierry Desmousseaux<sup>3</sup>    Pierre Blanc-Paques<sup>3</sup>    Jean Jaubert<sup>4</sup>

<sup>1</sup> Onera - The French Aerospace Lab, Toulouse

<sup>2</sup> LAAS/CNRS, Toulouse

<sup>3</sup> Airbus Defence and Space, Toulouse

<sup>4</sup> CNES, Toulouse

FirstName.LastName@onera.fr    FirstName.LastName@laas.fr  
FirstName.LastName@astrium.eads.net    FirstName.LastName@cnes.fr

## Résumé

Les satellites d'observation de la Terre sont des senseurs qui acquièrent des données, les compressent et les mémorisent à bord, puis les vident vers le sol. Du fait de l'utilisation d'algorithmes de compression de plus en plus sophistiqués, le volume de données résultant d'une acquisition est de moins en moins prédictible. Dans de telles conditions, planifier les activités de vidage du satellite hors-ligne au sol est de plus en plus problématique. Dans ce papier, nous rapportons les résultats d'une étude visant à évaluer le bénéfice d'une planification des activités de vidage à bord, quand le volume de données produit par chaque acquisition est connu.

Le problème de vidage des données à résoudre à bord est un problème d'affectation et d'ordonnement avec des ressources non partageables, des contraintes de précédence, des durées minimum dépendantes du temps et un critère d'optimisation complexe. La librairie générique InCELL [18] est utilisée pour modéliser contraintes et critère, vérifier les contraintes non temporelles, propager les contraintes temporelles et évaluer le critère. Au dessus de cette librairie, des algorithmes de recherche gloutonne et locale ont été développés pour produire des plans de vidage, compte-tenu du temps et des ressources de calcul limités disponibles à bord.

## Abstract

Earth observation satellites are space sensors which acquire data, compress and record it on board, and then download it to the ground. Because of the use of more and more sophisticated compression al-

gorithms, the amount of data resulting from an acquisition is less and less predictable. In such conditions, planning satellite data download activities offline on the ground is more and more problematic. In this paper, we report the results of a work aiming at evaluating the positive impact of planning downloads on-board when the amount of data produced by each acquisition is known.

The data download problem to be solved on board is an assignment and scheduling problem with unsharable resources, precedence constraints, time-dependent minimum durations, and a complex optimization criterion. The generic InCELL library [18]) is used to model constraints and criterion, to check non temporal constraints, to propagate temporal constraints, and to evaluate the criterion. On top of this library, greedy and local search algorithms have been designed to produce download plans with limited time and computing resources available on board.

## 1 Introduction

Earth observation satellites are space sensors which acquire data, compress and record it on board, and then download it to the ground. Because of the use of more and more sophisticated compression algorithms, the amount of data that results from an acquisition and thus is recorded on board and must be downloaded to the ground is more and more unpredictable. It depends on the data that has been acquired. For example, in the case of optical instruments, the presence of clouds over the observed area allows high compression rates and results in a low amount of data to be

recorded and downloaded.

In such conditions, the usual way of managing Earth observation satellites becomes more and more problematic. Indeed, until now, all the decisions are made offline on the ground and the satellite is a simple executive which neither makes, nor changes any decision. Typically, every day, a satellite activity plan, involving acquisition and download activities with precise starting times, is built on the ground for the next day. This plan is uploaded to the satellite through any ground control station and is executed without any change by the satellite. In a context where the amount of data to be recorded and downloaded is uncertain, if maximum volumes are considered when building plans, plans never fail, but the system may be underused. If expected volumes or any volumes lower than maximum are considered, plans may fail.

An alternative option is to change at least partially the way of managing these satellites and to postpone download decisions as late as possible, when acquisitions have been performed and generated volumes are known. Because these satellites are not continuously accessible by a ground control station and because generated volumes are known on board, such decisions must and can be made on board. For example, just before a ground reception station visibility window where downloads are possible, the satellite can build a download plan taking into account the volumes generated by all the acquisitions that have been already performed.

In this study, we considered the context of the future Post-Pleiades satellites: the generation of Earth observation satellites that will follow the agile Earth optical observation Pleiades satellites that are currently operational. In such a context, new download constraints must be considered such as the recording of data over several memory banks, the use of several download channels, or the use of several data encryption keys. Moreover, downloading criteria to be optimized are the number and the importance of downloaded acquisitions, the delay between acquisitions and downloads, and the fair sharing of the satellite usage between users.

The first goal of this study was to design on top of a generic constraint library efficient algorithms able to make download decisions autonomously on board. The second goal was to assess the operational impact of using such on-board algorithms with regard to the usual way of managing satellites completely from the ground.

The paper is organized as follows. In the next section, the organization we assume for mixed ground and board decision-making is described. Then, the data download planning problem is informally described and a constraint-based model of this problem is proposed and analyzed. Works related to data download planning are discussed before describing the algorithmic approach that has been chosen, based on the ideas of *Constraint-based Local Search*

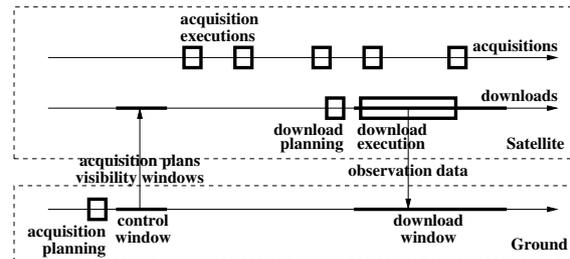


FIGURE 1 – Decision-making organization.

(CLS [9]) and on the use of the generic InCELL library (*Invariant-based Constraint Evaluation Library* [18]). After a presentation of the various scenarios used for evaluation, experimental results allow the efficiency of several algorithmic variants to be compared and the positive impact of onboard decision-making to be assessed.

## 2 Decision-making Organization

We consider the context of the future Post-Pleiades satellites with the following assumptions about the physical system:

- The observation instrument is body-mounted on the satellite, but the data download antenna is mobile within some limits.
- To observe a ground area, the observation instrument and thus the whole satellite must be pointed to it. To download data to a station, the download antenna must be pointed to it.
- Thanks to gyroscopic actuators, the satellite is agile and able to move quickly around its gravity center along its three axes while moving along its orbit.
- Acquisitions and downloads can be performed in parallel.

In this context, Fig. 1 shows the decision-making organization we assume.

**Acquisition planning** Acquisition plans are built offline on the ground as it is usually done until now. This is justified by the fact that building them is computationally expensive and that all the information about acquisition requests is available on the ground. These plans consider neither memory and download limitations, nor download activities. From the acquisition plan, it is possible to deduce, for any ground reception station  $st$ , the windows over which downloading data to  $st$  is effectively possible (pointing the mobile antenna towards  $st$  is possible, taking into account the satellite position and orientation, and the maximum antenna orientation angle). The acquisition plan and the resulting visibility windows are then uploaded to the satellite through any ground control station.

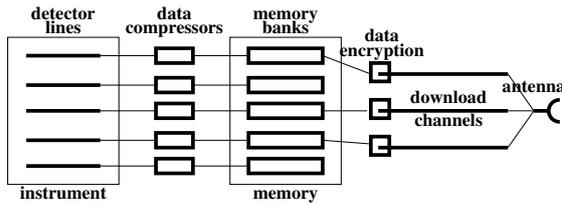


FIGURE 2 – How data is produced, recorded, and downloaded.

**Acquisition execution** These acquisition plans are then executed on board without any modification except when an acquisition  $a$  would lead to a memory overflow : in this case,  $a$  is performed if and only if it is possible to free enough memory by removing from memory lower priority acquisitions.

**Download planning** Download plans are built on board before any visibility window or set of overlapping visibility windows. This is justified by the fact that information about the volumes of data generated by acquisitions is available on board. These plans take into account exact volumes to be downloaded for all the acquisitions that have been already performed and maximum volumes for all the acquisitions that will finish during the visibility window(s).

**Download execution** These download plans are then executed in a flexible way, taking into account the fact that the volumes generated by the acquisitions that finish during the visibility window(s) may be smaller than the maximum volumes that have been taken into account when planning. This may allow downloads to be started earlier than expected in the plan.

### 3 Data Download Planning Problem

Fig. 2 shows how data is recorded, memorized, and then downloaded.

**Data production and recording** Each acquisition activates a subset of detector lines, allowing a multi-frequency observation and resulting in a set of data files, each one recorded in one memory bank with some size that depends on the compression rate.

**Data downloading** Data downloading can use several concurrent emission channels. Each file can be downloaded using one channel. Interrupting a file download is not acceptable. The data downloading rate is a piecewise constant function of the satellite-station distance. Due to the movement of the satellite on its orbit and of the Earth on itself, the satellite-station distance evolves and hence the download duration of a file depends on the time at which download starts. The files that result from an acquisition can be downloaded in any order using any channels, but must be

all downloaded within one visibility window. Channels and memory banks are unsharable resources. This means that, if two files are recorded on the same memory bank or use the same channel, their downloads cannot overlap.

**Data encryption** Data is encrypted before download. One encryption key is associated with each user. Physically, one encryption component is associated with each channel, allowing data associated with several users to be downloaded concurrently. Moreover, one key change table which contains key changes and their precise times is associated with each channel, allowing data associated with several users to be downloaded sequentially on each channel. The number of changes it is possible to record in this table is limited. Resetting this table takes some time.

**Download windows** Data downloading is only possible within visibility windows. Moreover, depending on the user, only some stations (and thus some visibility windows) are allowed for data download. We assume that a visibility window may involve several download windows, each one with an associated key change table. Hence, between any two successive download windows, a minimum time is necessary to reset the key change table. Moreover, if the successive download windows are associated with two different stations, moving from the first to the second takes some time to point the download antenna towards the new station. This transition duration depends on the time at which transition starts, due to the movement of the satellite on its orbit and on itself and to the movement of the Earth on itself.

**Download release and due dates** Every acquisition download must start after acquisition and finish before a delivery deadline beyond which data is no longer valuable.

Fig. 3 represents a valid download plan, where assignment and scheduling decisions have been made, but download starting times have not been set yet (temporally flexible plan). This plan involves 4 acquisitions A, B, C, and D, each one resulting in 5 files (A1, A2, A3, A4, and A5 for acquisition A) distributed over the 5 memory banks, 2 overlapping visibility windows (Vw1 towards station S1 and Vw2 towards station S2), one download window per visibility window, and 3 channels.

Only temporal precedence constraints are represented : (1) sequences of file downloads on each channel, (2) sequences of file downloads from each memory bank, (3) downloads within visibility windows, transition duration between two successive download windows due to (4) antenna move and to (5) key change table reset, (6) acquisitions before downloads (for acquisition C which ends during visibility window Vw1 at time  $Ea(C)$ ), and (7) downloads before deadline (for acquisition D whose delivery deadline is  $Dl(D)$ ). File download durations (8) are implicitly represented by the rectangles associated with each file.

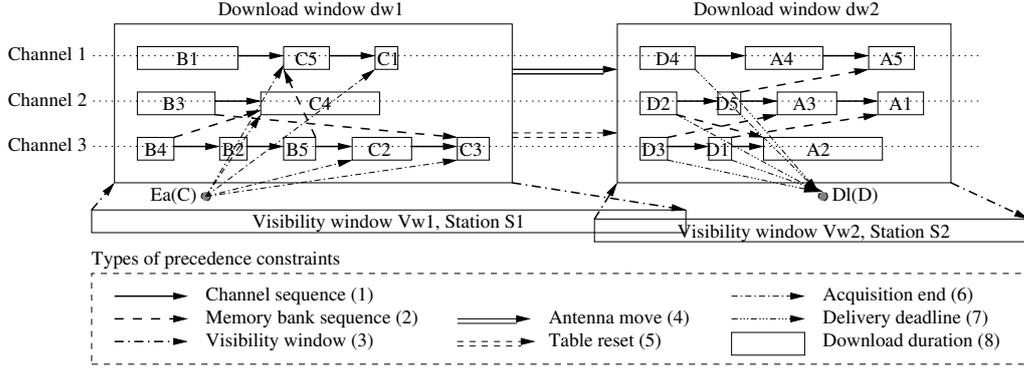


FIGURE 3 – Example of download plan with temporal precedence constraints.

Non temporal constraints are not represented: (9) all the files resulting from an acquisition downloaded within one visibility window whose associated station is allowed and (10) maximum number of key changes on each channel within each download window.

## 4 Model

### 4.1 Decision variables

Decision variables can be partitioned into non temporal and temporal variables. Non temporal decision variables include :

- a number  $ndw$  of download windows and the sequence  $dwSeq$  (of length  $ndw$ ) of download windows ;
- for each acquisition  $a$ , the download window  $dw_a$  within which it is downloaded and for each associated file  $f$  the channel  $ch_{a,f}$  on which it is downloaded ;
- for each download window  $w$ , the associated visibility window  $vw_w$ , for each channel  $c$  the sequence  $cSeq_{w,c}$  of file downloads on  $c$ , and for each memory bank  $m$  the sequence  $mSeq_{w,m}$  of file downloads from  $m$ .

Temporal decision variables include :

- for each acquisition  $a$  and each associated file  $f$ , the starting and ending times  $sdf_{a,f}$  and  $edf_{a,f}$  of download of  $f$  ;
- for each download window  $w$ , the starting and ending times  $sdw_w$  and  $edw_w$  of  $w$ .

### 4.2 Constraints

From this definition of the problem variables, all the constraints from (1) to (10) informally presented at the end of Sect. 3 can be expressed.

Many of them are simple temporal constraints [6] such as Constraint (3) which expresses that any download win-

dow must be included in its associated visibility window, with  $\mathbf{Sw}_{vw}$  and  $\mathbf{Ew}_{vw}$  the starting and ending times of visibility window  $vw$  :

$$\forall w \in \llbracket 1; ndw \rrbracket : (\mathbf{Sw}_{vw_w} \leq sdw_w) \wedge (edw_w \leq \mathbf{Ew}_{vw_w})$$

or Constraints (6) and (7) which express that any acquisition download must start after acquisition and finish before acquisition delivery deadline, with  $\mathbf{Na}$  the number of acquisitions,  $\mathbf{Nf}_a$  the number of files produced by acquisition  $a$ ,  $\mathbf{Ea}_a$  the ending time of  $a$ , and  $\mathbf{Dl}_a$  its delivery deadline :

$$\forall a \in \llbracket 1; \mathbf{Na} \rrbracket, \forall f \in \llbracket 1; \mathbf{Nf}_a \rrbracket : (\mathbf{Ea}_a \leq sdf_{a,f}) \wedge (edf_{a,f} \leq \mathbf{Dl}_a)$$

Some are time-dependent simple temporal constraints [19] such as Constraint (8) which expresses the duration of any file download as a function of the download starting time, with  $\mathbf{V}_{a,f}$  the volume of file  $f$  from acquisition  $a$  and  $\mathbf{DuDl}$  the download duration as a function of the volume to be downloaded, of the visibility window, and of the download starting time within this window :

$$\forall a \in \llbracket 1; \mathbf{Na} \rrbracket, \forall f \in \llbracket 1; \mathbf{Nf}_a \rrbracket : edf_{a,f} - sdf_{a,f} = \mathbf{DuDl}(\mathbf{V}_{a,f}, vw_{dw_a}, sdf_{a,f})$$

Some of them are not temporal such as Constraint (9) which expresses that any acquisition must be downloaded towards an allowed station, with  $\mathbf{St}_{vw}$  the station associated with visibility window  $vw$ ,  $\mathbf{Ss}_u$  the set of stations that are allowed by user  $u$ , and  $\mathbf{U}_a$  the user who requires acquisition  $a$  :

$$\forall a \in \llbracket 1; \mathbf{Na} \rrbracket : \mathbf{St}_{vw_{dw_a}} \in \mathbf{Ss}_{\mathbf{U}_a}$$

### 4.3 Optimization criterion

For the sake of simplification, the optimization criterion considered in this paper is purely utilitarian and does not

consider the objective of fair sharing of the satellite usage between users. It is defined as a vector of utilities, one per priority level. At each priority level  $p$ , the utility  $U_p$  is simply defined as the sum of the weights of the downloaded acquisitions of priority  $p$ , weighted by their freshness coefficient, in order to favour short delays between acquisitions and downloads :

$$\forall p \in \llbracket 1; \mathbf{Np} \rrbracket : U_p = \sum_{a \in \llbracket 1; \mathbf{Na} \rrbracket | \mathbf{P}_a = p} \mathbf{W}_a \cdot \mathbf{Fr}_a(eda_a)$$

where  $\mathbf{Np}$  is the number of priority levels,  $\mathbf{Na}$  the number of acquisitions,  $\mathbf{P}_a$  and  $\mathbf{W}_a$  the priority and weight of acquisition  $a$ ,  $\mathbf{Fr}_a$  the freshness level of  $a$  (between 0 and 1) as a monotonically decreasing function of the ending time  $eda_a$  of the download of  $a$ :  $eda_a = \max_{f \in \llbracket 1; \mathbf{Nf}_a \rrbracket} edf_{a,f}$ .

Two download plans are compared by comparing the two associated utility vectors lexicographically from priority 1 to  $\mathbf{Np}$ : any improvement at any priority level is preferred to any improvement at lower priority levels.

## 5 Problem analysis

For the sake of simplification too, we assume in this paper that at most one download window (maybe none) is associated with any visibility window and that the resulting download windows are ordered according the starting times of their associated visibility windows. See the example of Fig. 3 where we consider two download windows  $dw1$  and  $dw2$ , and the sequence  $[dw1, dw2]$ .

In such conditions, roughly speaking, the problem we face combines three connected subproblems :

1. an assignment problem, where a download window (maybe none) is associated with each acquisition : variables  $dw_a$  ;
2. a scheduling problem, where channels are assigned to files and file downloads are ordered on each channel and each memory bank : variables  $ch_{a,f}$ ,  $cSeq_{w,c}$ , and  $mSeq_{w,m}$  ;
3. a temporal problem, where download starting and ending times can be fixed : variables  $sdf_{a,f}$ ,  $edf_{a,f}$ ,  $sdw_w$ , and  $edw_w$ .

**Assignment problem** The assignment problem is close to a *Multi-knapsack* problem [11] where objects are acquisitions and sacks are windows, but first sacks are preferred and conflicts may exist between sacks (overlapping windows).

**Scheduling problem** In each download window, the scheduling problem is close to a *Flexible Open-shop Scheduling* problem [16] with two types of unsharable resources : channels and memory banks. Each file download

requires one resource of each type, but the choice of the channel is free, whereas the bank is pre-allocated. The file download order is free.

**Temporal problem** When the assignment and scheduling problems are solved *i.e.*, when all the non temporal variables are assigned, the resulting temporal problem has the form of a *Simple Temporal Network* (STN [6]). The only exceptions are the constraints of download duration and of transition between download windows which are time dependent (download and transition durations depend on the time at which they start). The result is a *Time-dependent* STN (TSTN [19]) for which STN techniques can be extended and polynomial algorithms can decide on consistency/inconsistency and compute the earliest/latest times for all the temporal variables.

## 6 Related works

**Data download planning** Whereas the problem of selecting and scheduling acquisitions for Earth observation satellites has been extensively studied (see [13, 7] for partial surveys), studies about data download planning are more seldom.

The problem of planning (offline on the ground) data downloads from a Mars orbiter to Earth has been studied in [15, 3]. In [14, 20], it has been shown that this problem can be modeled as a Max-Flow problem and thus solved using either Max-Flow or Linear Programming polynomial algorithms. To use a similar approach in our context, we should linearize the criterion and relax many constraints, for example the constraints that enforce that a file download cannot be interrupted and distributed over several channels or download windows. Hence, a Max-Flow formulation of our problem can only produce upper bounds on the maximum utility.

In the context of the EO-1 experiment [5], acquisition requests may be generated either by ground users, or autonomously on board following the detection of ground phenomena such as volcanic eruptions, floods, or ice breakups by onboard data analysis algorithms. In such a context, acquisition and download plans are built or adapted on board using an iterative repair approach [4] : local search algorithms implemented in the generic CASPER planning tool.

**Uncertain volumes** The situation where the amount of data that is produced and must be downloaded is variable and uncertain is present in planetary exploration as well as in Earth observation and surveillance missions.

In [2] and [23], onboard planning and scheduling (including data download planning and scheduling) is studied in order to allow a planetary exploration rover to adapt itself to what it detects. More precisely, in [21], what is downloaded is selected in order to meet download limitations and to maximize the information value that scientists will be able to extract from it.

In [14], an iterated procedure is designed to lower filling peaks in memory banks in order to manage uncertainty about the volumes of data to be downloaded by a Mars orbiter and to limit the risk of memory bank overwriting. In [20], (Integer) Linear Programming formulations are proposed to address this problem.

In [5], uncertainty is managed using a reactive iterative repair planning approach.

In [22], the operational context is an electromagnetic Earth surveillance mission where the variability of the generated data volumes is very high. Several mechanisms, especially designed for onboard download decision-making, are compared in terms of computing time, number of downloads, and window utilization: decision rules, reactive planning, or sampling + planning.

## 7 Planning Algorithms

Globally, the planning algorithms we developed are non chronological heuristic greedy algorithms which choose acquisitions and add them to the download plan one after the other, without any backtrack to previous choices. This choice is justified by the limited computing time and resources available on board. It must be stressed that such a choice is not definitive and that other choices such as, for example, squeaky wheel optimization [10], tabu search [8], or simulated annealing [12] are possible on top of the same model. It must be also stressed that the greedy algorithms we developed include sophisticated mechanisms to make assignment choices, to schedule downloads, to check non temporal constraints, and to propagate temporal constraints. Following the problem analysis in Sect. 5, these algorithms involve three parts: download assignment, download scheduling, and constraint checking and propagation.

### 7.1 Download assignment

Assignment algorithms receive as input a current consistent download plan and a set of candidate acquisitions (non assigned yet). They produce as output an acquisition  $a$  (selected among the candidates) and a download window for  $a$ . We developed two assignment algorithms: **MaxWeight** and **MinRegret**.

**MaxWeight** is the most basic algorithm. At each step, it selects an acquisition  $a$  of maximum priority and maximum weight, with random tie-breaking, and selects the first window within which inserting  $a$  is possible, by exploring them chronologically.

**MinRegret** is a bit more sophisticated. It maintains for each acquisition  $a$  the first and the second windows ( $w1_a$  and  $w2_a$ ) within which inserting  $a$  is possible, and the regret that would result from not choosing the first. If  $eda1_a$  and  $eda2_a$  are the respective ending times of the download

of  $a$  when  $w1_a$  and  $w2_a$  are chosen, the regret  $R$  can be defined as follows :

$$\forall a \in \llbracket 1; \mathbf{Na} \rrbracket : R(a) = \mathbf{W}_a \cdot (\mathbf{Fr}_a(eda1_a) - \mathbf{Fr}_a(eda2_a))$$

Getting inspiration from classical knapsack heuristics, we select at each step an acquisition  $a$  of maximum priority and maximum ratio  $R(a) / \sum_{f \in \llbracket 1; \mathbf{Nf}_a \rrbracket} \mathbf{V}_{a,f}$  between regret and volume, and select the first window  $w1_a$  for it.

### 7.2 Download scheduling

Scheduling algorithms receive as input a current consistent download schedule, a download window  $w$ , and an acquisition  $a$  to add to  $w$ . They produce a new consistent download schedule, including  $a$  in  $w$ , or a failure when such an assignment has not been produced.

We developed three scheduling algorithms: **EnQueue**, **IdleFill**, and **Scheduler**. The first two are greedy, whereas the third uses local search mechanisms. Moreover, the first two build schedules where acquisition downloads are totally ordered: downloading  $a1$  before  $a2$  implies that, on every channel and every memory bank, no file of  $a2$  is downloaded before a file of  $a1$ . On the contrary, the third algorithm builds schedules where interleaving acquisition downloads is possible.

**EnQueue** is the most basic algorithm. When trying to insert an acquisition  $a$  in a window  $w$ , it inserts it systematically at the end of the current sequence of acquisition downloads.

**IdleFill** fixes the most obvious drawback of **EnQueue**: because some acquisitions finish during visibility windows, some downloads must wait for end of acquisition; this may result in idle times in the schedule. To fix that, **IdleFill** inserts acquisition  $a$  at the first position in the current sequence of acquisition downloads where at least one channel is idle and insertion is possible.

For both algorithms, following a most constrained first heuristics, the insertion of all the files of  $a$  is performed sequentially from the largest to the smallest file volume. In order to limit idle times in the schedule, for each file  $f$ , if there is a channel  $c$  on which the last file  $f'$  comes from the same memory bank as  $f$  does, then  $f$  is placed on  $c$  just after  $f'$ ; otherwise  $f$  is placed on a channel  $c$  that allows the earliest download.

**Scheduler** is a local search algorithm, inspired from efficient mechanisms used for solving classical scheduling problems. It is called in case of failure of the previous greedy algorithms and can start from the schedule they produced. In this schedule, constraints on visibility windows are violated and the goal is to produce a shorter schedule that meets these constraints. For that, the algorithm performs a sequence of local moves whose goal is to reduce the length of the schedule critical path (the sequence of downloads whose length induces the schedule length; if

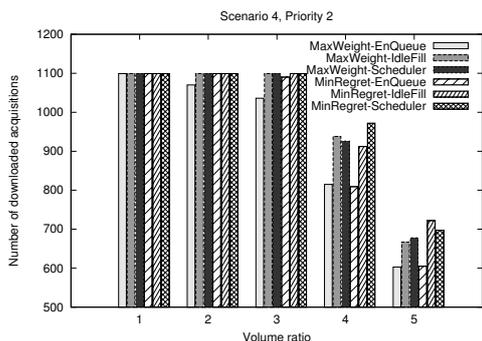


FIGURE 4 – Number of downloaded acquisitions at priority level 2 on Scenario 4 (higher is better).

	Scenario 1		Scenario 4	
	mean	max	mean	max
MaxWeight-EnQueue	0.041	0.221	0.048	0.460
MaxWeight-IdleFill	0.035	0.221	0.049	0.686
MaxWeight-Scheduler	0.232	4.149	0.655	21.708
MinRegret-EnQueue	0.161	1.434	0.239	3.531
MinRegret-IdleFill	0.152	1.544	0.215	2.839
MinRegret-Scheduler	0.157	1.521	2.116	78.511

TABLE 1 – Mean and max. computing time (in sec.) on Scenarios 1 and 4 (Intel i5-520 processor, 1.2GHz, 4GBRAM).

nothing is changed in this sequence, the schedule length will not be reduced and constraints on visibility windows will remain violated). Local moves are designed to guarantee that they never produce precedence cycles.

### 7.3 Constraint checking and propagation

The InCELL library (*Invariant-based Constraint Evaluation Library* [18]) is used to model variables, constraints and criterion, to check non temporal constraints, to propagate temporal constraints, and to evaluate the optimization criterion. InCELL draws its inspiration from the ideas of *Constraint-based Local Search* (CLS [9]).

In CLS, as in other declarative constrained optimization approaches, the user defines a model of its problem in terms of decision variables, constraints, and optimization criterion. Then, she/he defines a greedy or local search algorithm [1] in the variable assignment space. The model is not used to propagate constraints as in usual tree search algorithms, but for checking constraints and evaluating the optimization criterion as a function of the current variable assignment. Because the number of local moves performed within a limited time is a key to the success of greedy and local search algorithms, efficient techniques are

used to perform each local move as quickly as possible. These techniques use a translation of the model (variables, constraints, and criterion) into a DAG (*Directed Acyclic Graph*) of so-called invariants. Each invariant has a set of inputs and an output. It maintains a given function from inputs to the output, for example the fact that a variable is equal to the sum of other variables:  $x = \sum_{i=1}^N y_i$ . It maintains it incrementally: on this example, if the value of some variable  $y_k$  is changed, it is not necessary to recompute the whole sum from scratch; it suffices to add to  $x$  the difference between the new and the old value of  $y_k$ . Globally, after each local change in the variable assignment, the DAG of invariants is lazily and incrementally re-evaluated in a topological order: only the invariants whose one of the inputs is modified are incrementally re-evaluated.

InCELL is an implementation of CLS ideas with a focus on scheduling problems, and thus on time and resource management. In InCELL, multiple-input multiple-output invariants allow expressions, arithmetic and logical constraints, temporal and resource constraints to be expressed. Moreover, in InCELL, simple temporal constraints [6] of the form  $y - x \leq D$ , where  $x$  and  $y$  represent two temporal positions and  $D$  is constant, are handled in a specific way: temporal variables are not assigned as the other non temporal variables are; STN techniques are used to propagate temporal constraints, to check their consistency, and to compute earliest and latest values for each temporal variable. This specific treatment is justified by the existence of polynomial algorithms able to check the consistency/inconsistency of any STN and to compute earliest/latest values for each STN variable. Such algorithms generally do not exist for non temporal constraints. Moreover, InCELL allows time-dependent simple temporal constraints [19] of the form  $y - x \leq D(x, y)$ , where  $D$  is no longer a constant, but a function of  $x$  and  $y$ , to be expressed and handled the same way.

Once non temporal constraints have been checked and temporal constraints have been propagated by InCELL, the result is a download plan which assigns its earliest time to each temporal variable, because it is preferable to download data as early as possible. From this download plan, key change tables are set with precise change times for each download window and each channel.

## 8 Execution Algorithms

In the chosen decision-making organization, download plans are built before any visibility window(s) and assume maximum volumes for all the acquisitions that will finish during the visibility window(s). If actual volumes are smaller than maximum, it may be possible to start downloads earlier than expected in the plan, and thus to improve on the plan quality without modifying the assignment and scheduling decisions in the plan.

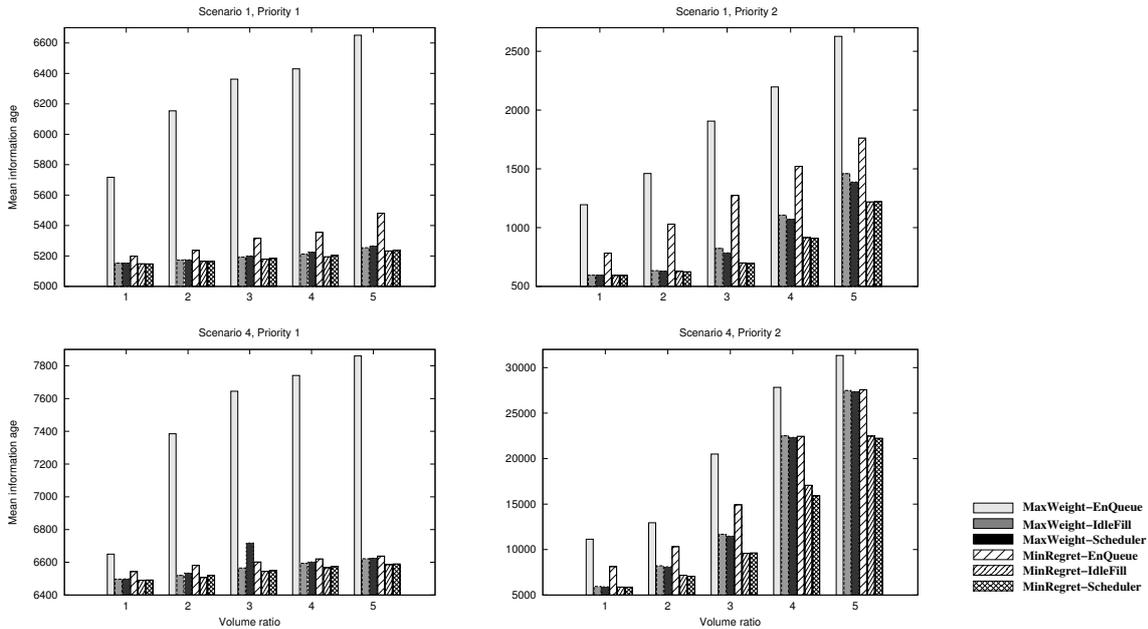


FIGURE 5 – Mean information age (in sec.) at priority levels 1 and 2 (left and right) on Scenarios 1 and 4 (top and bottom) (lower is better).

To implement such a flexible reactive way of executing plans, we draw our inspiration from the *Partial Order Schedule* approach (POS [17]) and build from any download plan an execution precedence graph (a DAG) which represents all the precedences that must be met by execution. Once this precedence graph has been built, it can be executed in a topological order: any node is executed as soon as all its predecessors in the graph have been executed. It can be easily shown that, because of the maximum volumes taken into account when planning, such an execution never leads to violation of the visibility window and download deadlines.

## 9 Experimental Results

### 9.1 Scenarios

The decision-making organization and the various planning algorithms have been experimented on several realistic scenarios provided by Airbus Defence and Space. Each scenario covers one day of satellite activity.

These scenarios involve 5 memory banks, 3 channels, 5 users, 2 priority levels, from 3 to 23 ground reception stations, from 20 to 115 associated visibility windows, and 1364 acquisitions to be downloaded. Because each acquisition produces 5 files, the number of files to be downloaded is equal to 6820. If  $V_{max}$  is the maximum volume of a file (without any compression), its actual volume is randomly generated between  $V_{max}/4$  and  $V_{max}$ . Scenarios

differ from each other according to the number of available ground reception stations and to the way acquisitions are distributed between users and priority levels.

We present the results that have been obtained on two typical scenarios: Scenario 1 which involves a large number of ground reception stations (23) resulting in many download opportunities, and Scenario 4 which involves a small number of ground reception stations (3) resulting in only few download opportunities. Moreover, we present the results that have been obtained on both scenarios with several assumptions about the volume of data produced by detectors before compression: volume ratio varying from 1 to 5 and leading to more and more oversubscribed scheduling problems. For example, Scenario 1 with a volume ratio equal to 1 is strongly undersubscribed: every acquisition can be downloaded in the first visibility window following acquisition. On the contrary, Scenario 4 with a volume ratio equal to 5 is strongly oversubscribed: acquisitions cannot be all downloaded and many of them remain in memory at the end of the day.

### 9.2 Comparison between planning algorithms

First, we compared the six download planning algorithms that result from combining assignment and scheduling algorithms: **MaxWeight** or **MinRegret** for assignment, and **EnQueue**, **IdleFill**, or **Scheduler** for scheduling.

With regard to the number of downloaded acquisitions,

Scenario 1 : Number of downloaded acquisitions						
volume ratio		1	2	3	4	5
prio. 1	ground	269	269	269	269	269
	board	269	269	269	269	269
prio. 2	ground	1087	1087	1087	988	729
	board	1087	1087	1087	1087	1087

Scenario 1 : Mean Information age						
volume ratio		1	2	3	4	5
prio. 1	ground	5167	5200	5232	5268	5419
	board	5149	5165	5179	5195	5233
prio. 2	ground	610	692	993	1505	2403
	board	593	628	700	915	1219

Scenario 4 : Number of downloaded acquisitions						
volume ratio		1	2	3	4	5
prio. 1	ground	244	244	244	244	244
	board	244	244	244	244	244
prio. 2	ground	1099	1099	595	297	246
	board	1099	1099	1099	912	722

Scenario 4 : Mean Information age						
volume ratio		1	2	3	4	5
prio. 1	ground	6639	6725	6758	6914	7125
	board	6490	6507	6544	6567	6586
prio. 2	ground	6700	12898	13539	13771	11784
	board	5875	7180	9591	17073	22474

TABLE 2 – Comparison between ground and board planning : number of downloaded acquisitions (left) and mean information age (in seconds ; right) at priority levels 1 and 2 on Scenarios 1 and 4 (top and bottom).

all the acquisitions are downloaded on Scenario 1, and all the acquisitions of priority 1 are downloaded on Scenario 4, but not all the acquisitions of priority 2 for high volume ratios. Fig. 4 shows the number of downloaded acquisitions of priority 2 on Scenario 4 as a function of the volume ratio (from 1 to 5).

Fig. 5 shows the mean information age in seconds (mean distance between acquisition and download over all the downloaded acquisitions) at priority levels 1 and 2 (left and right) on Scenarios 1 and 4 (top and bottom) as a function of the volume ratio (from 1 to 5). The fact that the information age at priority 1 is greater than at priority 2 is due to a smaller number of allowed ground reception stations for acquisitions of priority 1.

These results, in terms of number of downloaded acquisitions and mean information age, show that, for assignment, **MinRegret** is superior to **MaxWeight**, and that, for scheduling, **IdleFill** is clearly superior to **EnQueue**, but that **Scheduler** is only slightly superior to **IdleFill**. So, in terms of plan quality, **MinRegret-IdleFill** and **MinRegret-Scheduler** are the best algorithms.

Tab. 1 shows the mean and maximum computing time in seconds (over all the calls to planning, each one before a set of visibility windows) on Scenarios 1 and 4, only for a volume ratio of 1 : these computing times do not significantly change with higher ratios, from 2 to 5.

These results, in terms of computing time, show that **MaxWeight-EnQueue** and **MaxWeight-IdleFill** are the most efficient. Replacing **MaxWeight** by **MinRegret** multiplies the computing time by about 5. Replacing **EnQueue** or **IdleFill** by **Scheduler** may multiply it by 30.

As a consequence, the choice of the best algorithm depends on the computing resources available on board. If they are sufficient, **MinRegret-IdleFill** seems to be the best candidate. If not, this is **MaxWeight-IdleFill**.

### 9.3 Comparison between planning on the ground or on board

Then, we compared what can be obtained by planning downloads on the ground (planning on the ground over a one-day horizon, assuming maximum volumes, assigning precise times to downloads, and allowing no onboard flexibility) and by planning them on board (download planning and flexible execution on board). In both cases, a **MinRegret-IdleFill** planning algorithm is used.

Tab. 2 shows the number of downloaded acquisitions (left) and the mean information age (right) at priority levels 1 and 2 on Scenarios 1 and 4 (top and bottom) as a function of the volume ratio (from 1 to 5).

These results show the positive impact of onboard planning in terms of number of downloaded acquisitions which can be multiplied by 3 for acquisitions of priority 2 from volume ratios 3 or 4, and in terms of mean information age which can be divided by 2 for acquisitions of priority 2. In some cases, we can however observe that onboard planning increases the mean information age. This is due to a greater number of downloaded acquisitions that are downloaded later.

Moreover, these results demonstrate the right behavior of onboard planning algorithms which favour acquisitions of priority 1 : whatever the scenario and the volume ratio are, acquisitions of priority 1 are all downloaded and, as the volume ratio increases, the mean information age grows more slowly for acquisitions of priority 1 than for those of priority 2.

## 10 Conclusion

In this paper, we showed how uncertainty about the amount of data generated by acquisitions can be managed by using onboard simple but efficient download planning

and execution algorithms which combine a greedy or local search, a constraint-based model, and calls to a generic constraint evaluation/propagation tool.

We also showed the operational advantages that can be taken from autonomous onboard decision-making about downloads: increase in the number of downloaded acquisitions, decrease in the age of the downloaded data, possible use of fewer ground reception stations and/or fewer download windows, and possible increase in acquisition data size (wider instrument swath and/or higher image resolution).

## Références

- [1] E. Aarts and J. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.
- [2] R. Castano, T. Estlin, R. Anderson, D. Gaines, A. Castano, B. Bornstein, C. Chouinard, and M. Judd. OASIS: Onboard Autonomous Science Investigation System for Opportunistic Rover Science. *Journal of Field Robotics*, 24(5):379–397, 2007.
- [3] A. Cesta, G. Cortellessa, Simone Fratini, A. Oddi, and N. Policella. An Innovative Product for Space Mission Planning: An A Posteriori Evaluation. In *Proc. of the 17th International Conference on Automated Planning and Scheduling (ICAPS-07)*, Providence, RI, USA, 2007.
- [4] S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabi-deau. Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling. In *Proc. of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-00)*, pages 300–307, Breckenridge, CO, USA, 2000.
- [5] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabi-deau, R. Castano, A. Davies, R. Lee, D. Mandl, S. Frye, B. Trout, J. Hengemihle, J. D’Agostino, S. Shulman, S. Ungar, T. Brakke, D. Boyer, J. VanGaasbeck, R. Greeley, T. Doggett, V. Baker, J. Dohm, and F. Ip. The EO-1 Autonomous Science Agent. In *Proc. of the 3rd Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*, pages 420–427, New York City, USA, 2004.
- [6] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49:61–95, 1991.
- [7] A. Globus, J. Crawford, J. Lohn, and R. Morris. A Comparison of Techniques for Scheduling Earth Observing Satellites. In *Proc. of the 16th Conference on Innovative Applications of Artificial Intelligence (IAAI-04)*, San Jose, CA, USA, 2004.
- [8] F. Glover and M. Laguna. Tabu Search. In *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–141. Blackwell Scientific Publishing, 1993.
- [9] P. Van Hentenryck and L. Michel. *Constraint-based Local Search*. MIT Press, 2005.
- [10] D. Joslin and D. Clements. Squeaky Wheel Optimization. *Journal of Artificial Intelligence Research*, 10:353–373, 1999.
- [11] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.
- [12] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.
- [13] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille. Selecting and Scheduling Observations of Agile Satellites. *Aerospace Science and Technology*, 6:367–381, 2002.
- [14] A. Oddi and N. Policella. A Max-Flow Approach for Improving Robustness in a Spacecraft Downlink Schedule. In *Proc. of the 4th International Workshop on Planning and Scheduling for Space (IWSPSS-04)*, Darmstadt, Germany, 2004.
- [15] A. Oddi, N. Policella, A. Cesta, and G. Cortellessa. Generating High Quality Schedules for a Spacecraft Memory Downlink Problem. In *Proc. of the 9th International Conference on Principles and Practice of Constraint Programming (CP-03)*, pages 570–584, Cork, Ireland, 2003.
- [16] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 2012.
- [17] N. Policella, S. Smith, A. Cesta, and A. Oddi. Generating Robust Schedules through Temporal Flexibility. In *Proc. of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 209–218, Whistler, Canada, 2004.
- [18] C. Pralet and G. Verfaillie. Dynamic Online Planning and Scheduling using a Static Invariant-based Evaluation Model. In *Proc. of the 23rd International Conference on Automated Planning and Scheduling (ICAPS-13)*, Rome, Italy, 2013.
- [19] C. Pralet and G. Verfaillie. Time-dependent Simple Temporal Networks: Properties and Algorithms. *RAIRO Operations Research*, 47(2):173–198, 2013.
- [20] G. Righini and E. Tresoldi. A Mathematical Programming Solution to the Mars Express Memory Dumping Problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 40(3):268–277, 2010.
- [21] D. Thompson, T. Smith, and D. Wettergreen. Information-Optimal Selective Data Return for Autonomous Rover Traverse Science and Survey. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA-08)*, pages 968–973, 2008.
- [22] G. Verfaillie, G. Infantes, M. Lemaître, N. Théret, and T. Natolot. On-board Decision-making on Data Downloads. In *Proc. of the 7th International Workshop on Planning and Scheduling for Space (IWSPSS-11)*, Darmstadt, Germany, 2011.
- [23] M. Woods, A. Shaw, D. Barnes, D. Price, D. Long, and D. Pullan. Autonomous Science for an ExoMars Rover-Like Mission. *Journal of Field Robotics*, 26(4):358–390, 2009.