Satellite Data Download Management with Uncertainty about the Generated Volumes

Cédric Pralet and **Gérard Verfaillie** and **Adrien Maillard**
Onera, Toulouse, France
FirstName.LastName@onera.fr

Emmanuel Hébrard and **Nicolas Jozefowiez** and **Marie-Josée Huguet**
LAAS/CNRS, Toulouse, France
FirstName.LastName@laas.fr

Thierry Desmousceaux and **Pierre Blanc-Paques**
Astrium, Toulouse, France
FirstName.LastName@astrium.eads.net

Jean Jaubert
CNES, Toulouse, France
FirstName.LastName@cnes.fr

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 algorithms, the amount of data resulting from an acquisition is more and more unpredictable. 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 onboard when the amount of data produced by each acquisition is known.

The data download problem to be solved is an assignment and scheduling problem with unsharable resources, sequences of activities, precedence constraints, time-dependent minimum durations, and a complex optimization criterion. The generic InCELL library (Pralet and Verfaillie 2013a)) is used to model constraints and criterion, to check non temporal constraints, to propagate temporal constraints, and to evaluate the optimization 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 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 executive. 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 be 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 generation of the agile Earth optical observation Pleiades satellites that are currently operational. In such a context, new download constraints should 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, implement, and experiment efficient algorithms able to make download decisions autonomously on board. The second goal was to assess the operational impact of using such onboard algo-

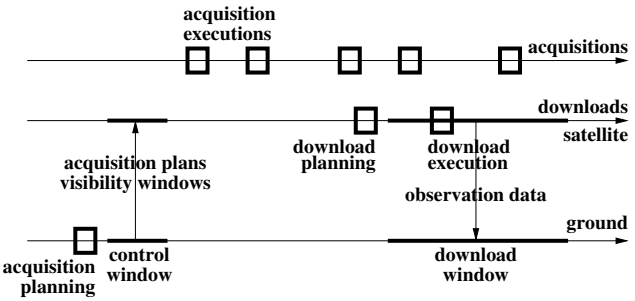

Figure 1: Decision-making organization.

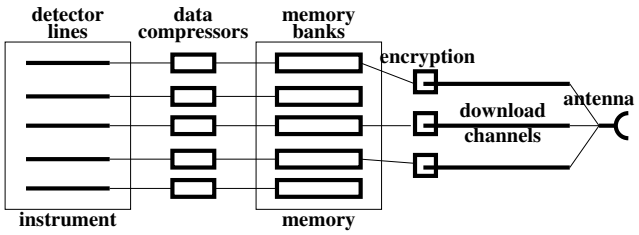

Figure 2: How data is produced, recorded, and downloaded.

rithms 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* (CLS (Hentenryck and Michel 2005)) and on the use of the generic InCELL library (*Invariant-based Constraint EvaLuation Library* (Pralet and Verfaillie 2013a)). 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). The acquisition plan and the resulting visibility windows are then uploaded to the satellite through any ground control station.

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 offline 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 planning These download plans are then executed online on board 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 offline 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

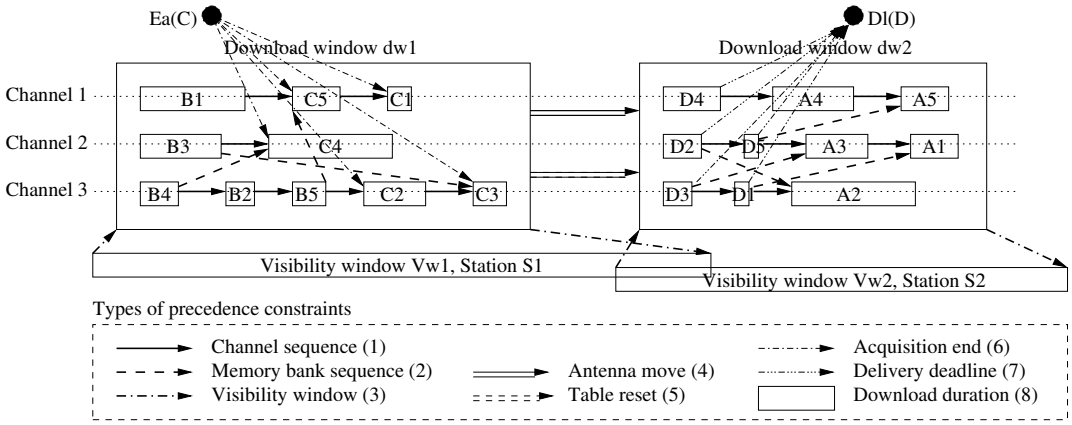

Figure 3: Example of download plan with temporal precedence constraints.

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. If two 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 has no longer value.

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

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

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 (Dechter, Meiri, and Pearl 1991) such as Constraint (3) which expresses that any download window must be included in its associated visibility window, with $\mathbf{Sw}_{vw}$ and $\mathbf{Ew}_{vw}$ the starting and ending times of any visibility window vw:

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

or Constraints (6) and (7) which express that any acquisition download must start after acquisition and finish before acquisition delivery deadline, with **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:

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

Some are time-dependent simple temporal constraints (Pralet and Verfaillie 2013b) 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 of acquisition a, and **Dr** the download rate as a function of the visibility window and of the download starting time within this window:

$$\forall a \in [\![1; \mathbf{Na}]\!], \forall f \in [\![1; \mathbf{Nf}_a]\!] : \\ edf_{a,f} - sdf_{a,f} = \mathbf{V}_{a,f} / \mathbf{Dr}(vw_{dw_a}, sdf_{a,f})$$

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

$$\forall a \in [\![1; \mathbf{Na}]\!] : \ \mathbf{St}_{vw_{dw_a}} \in \mathbf{Ss}_{\mathbf{U}_a}$$

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:

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

where **Np** is the number of priority levels, **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 its delivery time, $\mathbf{Nf}_a$ the number of files produced by a, and $eda_a = \max_{f \in [\![1; \mathbf{Nf}_a]\!]} edf_{a,f}$ the ending time of the download of a.

Two download plans are compared by comparing the two associated utility vectors lexicographically from priority 1 to **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 (Kellerer, Pferschy, and Pisinger 2004) where objects are acquisitions and sacks are windows, but first sacks are preferred.

Scheduling problem In each download window, the scheduling problem is close to an *Flexible Open-shop Scheduling* problem (Pinedo 2012) 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 acquisition and 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 (Dechter, Meiri, and Pearl 1991)). 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 at which they start). The result is a *Time-dependent* STN (TSTN (Pralet and Verfaillie 2013b)) for which STN techniques can be extended and polynomial algorithms can decide on consistency/inconsistency and compute the arliest/latest times for all the temporal variables.

Moreover, continuous state features, such as the satellite position and orientation, must be managed because of their impact on temporal constraints.

Finally, the criterion to be maximized is a complex hierarchical criterion whose value depends on assignment and scheduling decisions and on their temporal consequences (earliest download times).

6 Related works

Data download planning Whereas the problem of selecting and scheduling acquisitions for Earth observation satellites has been extensively studied (see (Lemaître et al. 2002; Globus et al. 2004) for partial surveys), studies about data download planning are more seldom.

However, the problem of planning (offline on the ground) data downloads from a Mars orbiter to Earth has been studied in (Oddi et al. 2003). In (Oddi and Policella 2004; Righini and Tresoldi 2010), it has been shown that this problem can be modelled 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 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 cannot produce valid solutions. It can only produce upper bounds on the problem optimum.

In the context of the EO-1 experiment (Chien et al. 2004), 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 (Chien et al. 2000) (local search algorithms implemented in the generic CASPER 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 (Castano et al. 2007) and (Woods et al. 2009), 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 (Thompson, Smith, and Wettergreen 2008), 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 (Oddi and Policella 2004), 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 (Righini and Tresoldi 2010), (Integer) Linear Programming formulations are proposed to address this problem.

In (Chien et al. 2004), uncertainty is managed using the basic reactive iterative repair planning approach.

In (Verfaillie et al. 2011), 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, sampling and 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. Precise algorithmic choices are not definitive and other choices will be possible on top of the same model when implementing actual planning algorithms. However, these globally simple algorithms 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, they 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 Δ^{w2}_{w1} can be defined as follows:

$$\forall a \in [\![1; \mathbf{Na}]\!] : \ \Delta^{w2}_{w1}(a) = \mathbf{W}_a \cdot (\mathbf{Fr}_a(eda1_a) - \mathbf{Fr}_a(eda2_a))$$

Getting inspiration from classical Knapsack heuristics, we select at each step an acquisition a of maximum priority and maximum ratio $\Delta^{w2}_{w1}(a) / \sum_{f \in [\![1; \mathbf{Nf}_a]\!]} \mathbf{V}_{a,f}$ between regret and volume, and select the first window $w1_a$ for it.

7.2 Download scheduling

Scheduling algorithms receive as input a download window w, a current consistent download schedule in w, and an acquisition a to add to w. They produce a new consistent download schedule in w, including a, 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.

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' and the file f come from the same memory bank, 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 and download 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 nothing is changed in this sequence, the schedule length will not be reduced and constraints on visibility and download windows will remain violated). Local moves are designed to guarantee that they never produce precedence cycles. Local search stops when a given maximum number of local moves is exceeded or when no improving local move has been found (local optimum).

7.3 Constraint checking and propagation

The InCELL library (*Invariant-based Constraint EvaLuation Library* (Pralet and Verfaillie 2013a)) 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 (Hentenryck and Michel 2005)).

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 its greedy or local search algorithm (Aarts and Lenstra 1997) 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 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 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 the modeling and solving of 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 (Dechter, Meiri, and Pearl 1991) 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 (*Simple Temporal Network*) 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 (Pralet and Verfaillie 2013b) 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. To allow onboard implementation, InCELL has been designed without any dynamic memory allocation.

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.

To implement such a flexible reactive way of executing plans, we draw our inspiration from the *Partial Order Schedule* approach (POS (Policella et al. 2004)) 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

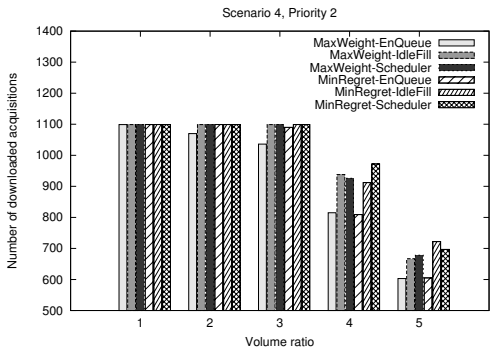

Figure 4: Number of downloaded acquisitions at priority level 2 on Scenario 4.

	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 maximum computing time (in seconds) on Scenarios 1 and 4, using a Intel i5-520 processor with 1.2GHz and 4GBRAM.

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 such an execution may allow downloads to be performed earlier and, because of the maximum volumes taken into account when planning, 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 Astrium. 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 $Vmax$ is the maximum volume of a file (without any compression), its actual volume is randomly generated between $Vmax/4$ and $Vmax$. 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.

The size of the InCELL model able to deal with these scenarios is of 70MB.

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 these two 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, 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 data delivery 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**.

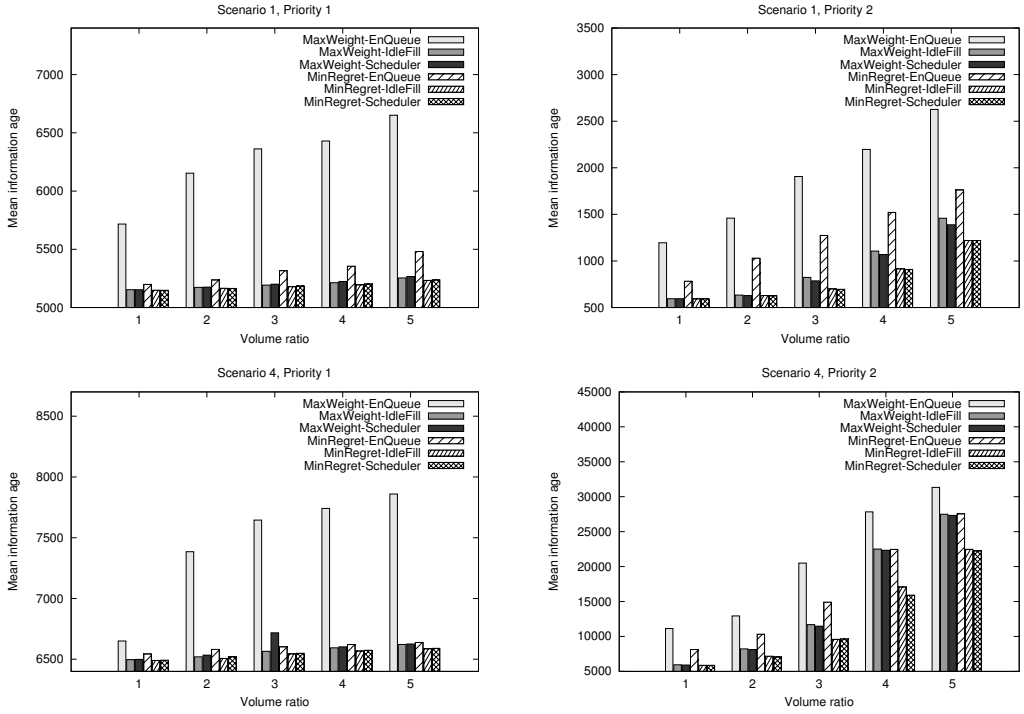

Figure 5: Mean information age (in seconds) at priority levels 1 and 2 (left and right) on Scenarios 1 and 4 (top and bottom).

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 show 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 show the operational advantages that can be taken from autonomous onboard decision-making about downloads: less data to be downloaded, 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).

References

Aarts, E., and Lenstra, J., eds. 1997. *Local Search in Combinatorial Optimization*. John Wiley & Sons.

Castano, R.; Estlin, T.; Anderson, R.; Gaines, D.; Castano, A.; Bornstein, B.; Chouinard, C.; and Judd, M. 2007. OASIS: Onboard Autonomous Science Investigation System for Opportunistic Rover Science. *Journal of Field Robotics* 24(5):379–397.

Chien, S.; Knight, R.; Stechert, A.; R.Sherwood; and Rabideau, G. 2000. 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)*, 300–307.

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).

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davies, A.; Lee, R.; Mandl, D.; Frye, S.; Trout, B.; Hengemihle, J.; D'Agostino, J.; Shulman, S.; Ungar, S.; Brakke, T.; Boyer, D.; VanGaasbeck, J.; Greeley, R.; Doggett, T.; Baker, V.; Dohm, J.; and Ip, F. 2004. The EO-1 Autonomous Science Agent. In *Proc. of the 3rd Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*, 420–427.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal Constraint Networks. *Artificial Intelligence* 49:61–95.

Globus, A.; Crawford, J.; Lohn, J.; and Morris, R. 2004. A Comparison of Techniques for Scheduling Earth Observing Satellites. In *Proc. of the 16th Conference on Innovative Applications of Artificial Intelligence (IAAI-04)*.

Hentenryck, P. V., and Michel, L. 2005. *Constraint-based Local Search*. MIT Press.

Kellerer, H.; Pferschy, U.; and Pisinger, D. 2004. *Knapsack Problems*. Springer.

Lemaître, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.-M.; and Bataille, N. 2002. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology* 6:367–381.

Oddi, A., and Policella, N. 2004. 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 (IWPSS-04)*.

Oddi, A.; Policella, N.; Cesta, A.; and Cortellesa, G. 2003. 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)*, 570–584.

Pinedo, M. 2012. *Scheduling: Theory, Algorithms, and Systems*. Springer.

Policella, N.; Smith, S.; Cesta, A.; and Oddi, A. 2004. Generating Robust Schedules through Temporal Flexibility. In *Proc. of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 209–218.

Pralet, C., and Verfaillie, G. 2013a. 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)*.

Pralet, C., and Verfaillie, G. 2013b. Time-dependent Simple Temporal Networks: Properties and Algorithms. *RAIRO Operations Research* 47(2):173–198.

Righini, G., and Tresoldi, E. 2010. A Mathematical Programming Solution to the Mars Express Memory Dumping Problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 40(3):268–277.

Thompson, D.; Smith, T.; and Wettergreen, D. 2008. Information-Optimal Selective Data Return for Autonomous Rover Traverse Science and Survey. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA-08)*, 968–973.

Verfaillie, G.; Infantes, G.; Lemaître, M.; Théret, N.; and Natolot, T. 2011. On-board Decision-making on Data Downloads. In *Proc. of the 7th International Workshop on Planning and Scheduling for Space (IWPSS-11)*.

Woods, M.; Shaw, A.; Barnes, D.; Price, D.; Long, D.; and Pullan, D. 2009. Autonomous Science for an ExoMars Rover-Like Mission. *Journal of Field Robotics* 26(4):358–390.