Scheduling Scientific Experiments on the Rosetta/Philae Mission

G. Simonin, C. Artigues, E. Hebrard, and P. Lopez

CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, LAAS, F-31400 Toulouse, France
{gsimonin,artigues,hebrard,lopez}@laas.fr

Abstract. The Rosetta/Philae mission was launched in 2004 by the European Space Agency (ESA). It is scheduled to reach the comet $67P$/Churyumov-Gerasimenko in 2014 after traveling more than six billion kilometers. The Philae module will then be separated from the orbiter (Rosetta) to attempt the first ever landing on the surface of a comet. If it succeeds, it will engage a sequence of scientific exploratory experiments on the comet.
In this paper we describe a constraint programming model for scheduling the different experiments of the mission. A feasible plan must satisfy a number of constraints induced by energetic resources, precedence relations on activities, or incompatibility between instruments. Moreover, a very important aspect is related to the transfer (to the orbiter then to Earth) of all the data produced by the instruments. The capacity of inboard memories and the limitation of transfers within visibility windows between lander and orbiter, make the transfer policy implemented on the lander's CPU prone to data loss. We introduce a global constraint to handle data transfers. The goal of this constraint is to ensure that data-producing activities are scheduled in such a way that no data is lost.
Thanks to this constraint and to the filtering rules we propose, mission control engineers are now able to compute feasible plans in a few seconds for scenarios where minutes or even hours were previously often required. Moreover, in many cases, data transfers are now much more accurately simulated, thus increasing the reliability of the plans.

1 Introduction

The international Rosetta/Philae project is an European consortium mission approved in 1993, and is under the leadership of the German Aerospace Research Institute (DLR) and ESA[1]. The spacecraft was launched in 2004 by Ariane 5, and is set to travel more than six billion kilometers to finally reach and land on the comet $67P$/Churyumov-Gerasimenko in 2014 in order to analyze the comet structure. It will follow a complex trajectory which includes four gravity assist maneuvers (3 x Earth, 1 x Mars) before finally reaching the comet and enter its orbit. Then, the lander Philae will be deployed and will land on the surface of the comet. Philae features ten instruments, each developed by a European laboratory, to accomplish a given scientific experiment when approaching, or once landed on the comet. For instance, *CIVA* and *ROLIS* are two imaging

[1] European Space Agency

instruments, used to take panoramic pictures of the comet and microscopic images. The Alpha Proton X-ray Spectrometer (*APXS*) experiment analyses the chemical composition of the landing site and its potential alteration during the comet's approach to the Sun. The data obtained will be used to characterize the surface of the comet, to determine the chemical composition of the dust component, and to compare the dust with known meteorite types.

The exploratory mission will have three phases. The first one, SDL (*Separation-Descent-Landing*), will run for 30 minutes during which many experiments will be done. For the second phase, FSS (*First Science Sequence*), the execution of experiments will last 5 days. This phase is critical because the execution of the most energetically greedy experiments requires battery power. The quality of this schedule conditions the longevity of the batteries and is therefore key to the sucess of the mission. Finally, during the LTS phase (*Long Term Science*), scientific activities will be resumed at a much slower pace, using the lander's own solar panels to partially reload the batteries. This phase will continue for months until the probe is destroyed due to the extreme temperatures of the Sun.

This project is a collaboration with CNES[2] in Toulouse (France). The goal of the *Scientific Operation and Navigation Centre* (SONC) is to plan the sequence of experiments and maneuvers to be done in each of these phases while making the best use of the available resources. This project has many similarities with the (interrupted) Net-Lander program [6]. A first software (called MOST) has been developed on top of the Ilog-Scheduler/Solver library by an industrial subcontractor. Every instrument, subsystem and experiment has been modeled precisely in this framework, and it is therefore possible to check solutions with a high degree of confidence on their feasibility.

The main scientific experiments need to be scheduled to satisfy a number of constraints involving the concurrent use of energy (batteries), and of the main CPUs as well as each instrument's memory. Moreover, each experiment produces data that must be transferred to Earth. Each experiment has its own memory, collecting data as it is produced. This data is then transferred to a central mass-memory, then sent to Rosetta (the orbiter) when it is in *visibility*, i.e., above the horizon of the comet with respect to Philae. All transfers from the experiments to the mass memory, and from the mass memory to the orbiter are executed (that is, computed onboard) by the *Command and Data Management System* (CDMS). The transfer policy of the CDMS may lead to data loss when an experiment produces more data than its memory can store and its priority is not high enough to allow a transfer to the mass-memory. This is modeled within MOST using RESERVOIR constraints [5]. Data-producing activities fill the reservoir, while multiple pre-defined data transfer tasks of variable duration empty it. There are numerous problems related to data transfers in spatial applications [4,2], however the problem at hand is significantly different since plans are computed on the ground, and the data transfer policy is beyond our control.

This modeling choice has several drawbacks and it quickly became apparent that it was the critical aspect of the problem to tackle in order to find better solutions faster. The first problem with this model is that data transfers are not accurately represented. For each experiment, a sequence of tasks standing for data transfers are pre-defined.

[2] Centre National d'Etudes Spatiales

Their duration is constrained so that the experiment with the highest priority is allowed to transfer as much as possible, and no overlap is allowed among transfers. In the current implementation there is a transfer task every 120 seconds over the horizon, with a maximum duration of 120 seconds. This is too few to accurately represent the policy of the CDMS, however, this is already too much for Ilog-Scheduler to handle (the planning horizon may be up to one day, i.e., about 700 transfer tasks for each experiment).

Instead we propose to encapsulate data transfers into a global constraint. The decision variables are start times of data-producing activities (data-producing rate and duration are known in advance) and the priority permutation. This allows us to very quickly check the satisfiability of a schedule with respect to data transfer. Moreover, we can compute bounds allowing to filter out the domain of the variables standing for start time of the data-producing activities. Unfortunately enforcing arc consistency or even bounds consistency on this constraint is NP-hard, so we do not give a complete filtering algorithm. However, our approach reduces the solving time dramatically: from hours in some cases to seconds in all scenarios currently considered by the SONC. Moreover, the result is much more accurate, to the point that some scenarios for which MOST could not show that transfers were feasible can now be solved efficiently.

In Section 2 we briefly outline the energetic aspect of the problem and more formally define the data transfer aspect. Then, in Section 3 we introduce our approach to modeling data transfers. In particular we give an efficient satisfiability checking procedure and two filtering rules for the introduced global constraint. Last in Section 5, we report experimental results and compare results between old and new models.

2 Problem Description

Each experiment and subsystem can be seen as a list of activities to be scheduled. Notwithstanding data transfers, the problem can be seen as a scheduling problem over a set of experiments with relatively standard constraints.

Precedences: Activities within the same experiment might have precedence constraints; for instance the Lander also carries a Sampling Drilling and Distribution device (*SD2*), which will drill more than 20 cm into the surface, collect samples and deposit them in an oven. The oven then rotates to a position whereby it can be connected to the inlet of the gas management system of another instrument: Ptolemy. At this point, sample volatiles can be released into the analytical system of Ptolemy by heating the oven, and the analysis can begin.

Cumulative Resources: Activities within the experiments concurrently use the energy from a centralized source, mainly from batteries. All the energy sources (batteries and solar panels) are centralized on a main power line. The energy needed to run each task is supplied by an auxiliary power line. Each auxiliary power linked to a converter, and each converter is linked to the main power line. At each level, the total instant power delivered (by a line or a converter) cannot exceed a given threshold. Each activity is therefore associated to a unique auxiliary power line and to a unique converter. For each auxiliary power line, all the activities supplied by this line are constrained by

CUMULATIVE constraint [1] with capacity equal to this threshold. Similarly another a CUMULATIVE constraint is associated to each converter, and a last one is associated to the main power line, involving all activities of the problem.

State Resources: Each instrument can have multiple states along the schedule. Some activities can trigger the modification of the state of an instrument, and the processing of certain activities might be subject to some instruments being a given state. This is modeled using predefined state resources constraints in Ilog-Scheduler.

Data Transfer and Memory Constraints: Every experiment has its own memory. Some activities produce data, temporarily stored on the experiment's memory. Then this data will be transferred onto the mass-memory and subsequently to the orbiter.

The onboard CDMS controls all data transfers, from the experiments to the mass memory, and from the mass memory to the orbiter. Within a plan, experiments are (totally) ordered according to a priority function. Apart from this ordering, the CDMS is completely autonomous. It simply transfers data from the experiment with highest priority among those with transferable data. Moreover, it transfers data from the mass-memory to the orbiter whenever possible, that is, when there is some visibility. However, it does not ensure that all produced data will eventually be transferred back to Earth. When too much data is produced simultaneously and not enough can be transferred on the mass-memory, or when there is no visibility with the orbiter and therefore the mass-memory cannot be emptied, the capacity of an experiment's memory may be overloaded and data is lost.

Data transfers must be taken into account accurately in order to ensure that activities are sequenced in such a way that no data will be lost with respect to the CDMS policy.

3 A Global Constraint for Data Transfer

Except for the data transfer aspect, all the constraints above can be modeled using the standard methods and algorithms [7] all available in Ilog-Scheduler. Hence, we focus on data transfers and propose a global constraint to reason about this aspect of the problem.

From now on, we consider a set $\boldsymbol{\{E_1, \ldots, E_m\}}$ of m experiments. An *experiment* $\boldsymbol{E_k = \{t_{k1}, \ldots, t_{kn}\}}$ is a set of data-producing tasks[3], and is associated with a memory of capacity $\boldsymbol{M_k}$. A task $\boldsymbol{t_{ki}}$ produces data which is sent during a time period $\boldsymbol{p_{ki}}$ at a rate $\boldsymbol{\pi_{ki}}$ to the experiment's memory. The lander possesses a mass memory of capacity $\boldsymbol{M_0}$, where data can be transferred from experiments.

The CDMS is given as input a priority ordering on experiments. For $i \in [1, \ldots, m]$, we denote by $\boldsymbol{P(i)}$ the experiment at rank i in this ordering and its dual $R(k)$ standing for the rank of experiment E_k in the priority ordering ($P(i) = k \Leftrightarrow R(k) = i$). We shall say that experiment E_k has higher priority than experiment E_j iff $R(k) < R(j)$.

Data can only be transferred out to the orbiter when it is in *visibility*, that is in the line of sight of the lander over the horizon of the comet. Visibility is represented

[3] To simplify the notations, we assume that all experiments have the same number of activities. This is of course not the case, however it does not affect the methods we introduce.

as a set of intervals $\{[a_1, b_1], \dots, [a_v, b_v]\}$ in the scheduling horizon which lengths and frequencies depend on the chosen orbit. We shall use $V(t)$ as a Boolean function which equals true iff time t is included in one of the visibility intervals. Moreover, data is transferred in and out memories by *block* units of 256 bytes.

We consider the following decision variables: $s_{11}, \dots, s_{mn}$, with domain in $[0, \dots, H]$, standing for the start times of data-producing tasks $\{t_{11}, \dots, t_{mn}\}$, respectively. Here we will assume that the priority permutation is fixed. Indeed there are often few experiments and exploring all permutations would be easy.

The fact that data loss should be avoided can be seen as a relation (i.e., a constraint) between the decision variables above. It is relatively easy to understand this relation procedurally since the CDMS policy is deterministic. Given a priority ordering and a fixed schedule of the data-producing activities, one can unroll the rules outlined above to check whether the CDMS policy will lead to data loss or not.

In this section, we formally define the constraint DATATRANSFER$([s_{11}, \dots, s_{mn}])$ ensuring that the schedule of tasks $\{t_{11}, \dots, t_{mn}\}$. is such that no data is lost.

First, in Section 3.1 we discuss an "exact" definition based on following the transfer of each block of data individually. However, this formulation is not practical, so we propose an alternative model in Section 3.2. The basic idea is to represent all the data produced by an activity as a continuous quantity. With this viewpoint, tasks that do not produce data as fast as it can be transferred are challenging to model. Indeed, the CDMS actually waits for a block of data to be completed before transferring it, and can therefore use this waiting time to transfer data from other experiments with lower priority. We approximate this "vertical" partition of the data bus, by a "horizontal" partition, i.e., we consider that the bandwidth can be divided over parallel transfers. This model allows us to represent memory usage very precisely, with a computational complexity independent on the time horizon and on the amount of data produced.

3.1 CDMS Policy

In this section we detail the policy of the CDMS, then we define what would be a constraint modeling exactly the relation between the start time of data-producing activities it induces.

The CDMS transfers data by blocks of 256 bytes. Its policy is relatively simple and can be described using a simple automated earliest transfer scheduling algorithm (AETS). AETS runs the two following processes in parallel:

- Repeat: Scan experiments by order of priority until one with at least one block of data on its memory is found. In that case, transfer one block from this experiment to the mass memory unless the mass memory is full.
- Repeat: If the orbiter is visible, and there is at least one block of data on the mass memory, then dump one block (transfer from the mass memory to the orbiter).

So we can define the DATATRANSFER constraint as the relation allowing only assignments of start times and priorities such that given the CDMS policy (AETS), no block of data is produced while the memory of the experiment is full.

In order to specify this constraint as precisely as possible we would need to consider each block of data, and its associated transfer task, individually. More precisely, we need

$\pi_{ki}p_{ki}$ transfer tasks for each data-producing activity t_{ki}. The release time of the j^{th} block's transfer task is $s_{ki} + j(1/\pi_{ki})$, where s_{ki} is the start time of t_{ki}. Moreover, the start times and durations of these transfer tasks are functionally dependent on the start times of data-producing activities and experiment priorities. This dependence relation is a consequence of the AETS procedure.

The DATATRANSFER constraint is NP-complete to satisfy, hence NP-hard to filter. Indeed, consider the particular case where memory capacities are all of exactly one block of data, and the mass-memory is unlimited. In this case, each activity must transfer a block to the mass-memory as soon as it is produced. In other words, we can see a transfer task as non-interruptible. However, since there is a single transfer channel to the mass-memory, no overlap is possible between these tasks. Since we have time windows on the variables s_{ki}, this particular case is therefore equivalent to a disjunctive unary resource, i.e., it is strongly NP-hard.

3.2 Approximated Definition

It is difficult to capture very precisely the behavior of the CDMS. Moreover, it is not practical since it involves manipulating a very large number of transfer tasks. When we consider a data-producing activity in isolation, the number of transfer tasks and their frequency is easy to compute. However, when we consider several data-producing activities with different priorities and unknown start times, this viewpoint becomes impractical. We therefore propose an alternative model that approximate very closely the amount of transferred data with a reasonable time and space complexity.

The basic idea is straightforward. Consider a task t_{ki} that produces more data than it can transfer $\tau_{r(k,t)} \leq \pi_{ki}$, with $\tau_{r(k,t)}$ the transfer rate at time t from an experiment E_k to mass-memory. Suppose first that there is no task with higher priority. The transfer can be seen as a continuous task of duration $\frac{\pi_{ki}p_{ki}}{\tau_{r(k,t)}}$. However there are three difficulties.

First, blocks are transferred from experiments memories to the mass memory at a constant rate. However, when seeking which experiment to transfer from, the length of the scanning process depends on the number of *active* experiments and on the priority of the experiment eventually selected. An experiment E_k is active between the start of its first activity and the end time of its last activity, or if the experiment memory is not empty. The transfer rate is thus larger in practice for the higher priority experiments as they are scanned first. To emulate this, we use variable transfer rates. The potential transfer rate $\boldsymbol{\tau_{r(k,t)}}$ depends on the number x of active experiments and on its relative priority y among them. The actual value is read in a table which entries were measured experimentally. The rate above applies to non-visibility periods. According to the same principles, another table gives us the transfer rate $\boldsymbol{\tau'_{r(k,t)}}$ while in visibility (it is lower due to the parallelism with memory dumps). Transfers between mass-memory and the orbiter have a constant rate denoted $\boldsymbol{\tau^{mm}}$.

Second, transfer tasks can be interrupted, however, they are different from classic preemptive tasks in that we do not decide when the interruption occurs. When an experiment with higher priority starts producing data, it preempts any current transfer of lower priority. This is easy to model since this is the unique context where an interruption can happen. If there is no experiment with higher priority to interrupt the transfer, the usage of the experiment's memory increases at rate $\pi_{ki} - \tau_{r(k,t)}$ during p_{ki} seconds. Similarly, during $\frac{\pi_{ki} p_{ki}}{\tau_{r(k,t)}}$ seconds the usage of the mass memory increases at rate $\tau_{r(k,t)}$.

The third difficulty concerns tasks producing data at a lower rate than the possible transfer rate (i.e., $\tau_{r(k,t)} > \pi_{ki}$). In this case, data is transferred one block at a time, with a lag between each transfer to wait for the next block to be produced (see Fig. 1).

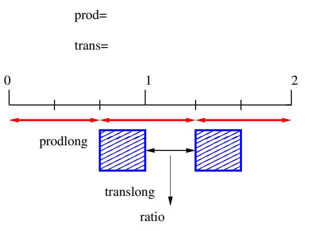

Fig. 1: Example of exact data transfer

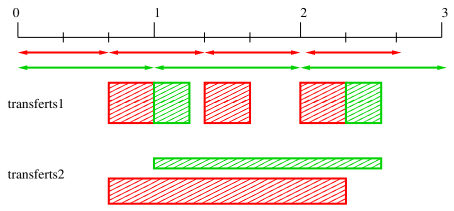

Fig. 2: Example of two data transfer tasks with both model

Other tasks of lower priority with non-empty memory can use these gaps to begin the transfer of a block of data. In other words, the duration of the transfer of highest priority is still very close to $\frac{\pi_{ki} p_{ki}}{\tau_{r(k,t)}}$ seconds, however other transfer can be squeezed in that same period. In order to simulate this, we consider that the data bus has a capacity (bandwidth) normalized to 1. The demand of a task t_{ki} at time t is $\min(1, \frac{\pi_{ki}}{\tau_{r(k,t)}})$. Bandwidth is allocated recursively, according to priority (see Figure 2).

Let $\boldsymbol{m_k^t}$ stand for the quantity of data in the memory of experiment E_k at time t (with $t \in \mathbb{R}$) and $\boldsymbol{m_0^t}$ be the the quantity of data in the mass-memory. Moreover, let π_k^t stand for the data-producing rate on experiment E_k at time t, $p(\tau_k^t)$ stand for the potential transfer rate of experiment E_k at time t if it was of highest priority and let τ_k^t stand for the actual transfer rate from experiment E_k to the mass memory at time t.

Definition 1.

$$\textsc{DataTransfer}(s_{11}, \ldots, s_{mn}) \Leftrightarrow$$

$$\forall t, k, \qquad \pi_k^t = \begin{cases} \pi_{ki} \text{ if } \exists i \; s.t. \; s_{ki} \leq t \leq s_{ki} + p_{ki} \\ 0 \quad \text{otherwise} \end{cases} \tag{1}$$

$$\forall t, k, \quad p(\tau_k^t) = \begin{cases} 0 & \text{if } m_0^t = M_0 \,\vee\, (m_k^t = \pi_k^t = 0) \\ \tau_{r(k,t)} & \text{if } m_0^t < M_0 \,\wedge\, m_k^t > 0 \\ \min(\pi_k^t, \tau_{r(k,t)}) & \text{otherwise} \end{cases} \tag{2}$$

$$\forall t, k, \qquad \tau_k^t = \min(p(\tau_k^t), \tau_{r(k,t)}(1 - \textstyle\sum_{i=1}^{R(k)-1} \frac{\tau_{P(i)}^t}{\tau_{r(P(i),t)}})) \tag{3}$$

$$\forall t, k, \qquad m_k^t = \textstyle\int_0^t (\pi_k^t - \tau_k^t) \mathrm{d}t \tag{4}$$

$$\forall t, \qquad m_0^t = \textstyle\int_0^t (\sum_{k=1}^m \tau_k^t - V(t)\tau^{mm}) \mathrm{d}t \tag{5}$$

Equation 1 simply states that if a data-producing activity t_{ki} is running at time t, then the data-producing rate π_k^t of an experiment k at that time is equal to the data-producing rate of t_{ki}, and it is null otherwise.

Equation 2 defines the *expected transfer rate* $p(\tau_k^t)$ of experiment E_k at time t if it is not trumped by other experiments of higher priority. If there is no data on the memory and no data being produced, or if the mass memory is full, this rate is null. Otherwise, if there is some data on memory, it can be transferred at the maximum available rate $(\tau_{r(k,t)})$. If the memory is empty, but data is being produced, we assume that it cannot be transferred at a higher rate than it is produced.

Equation 3 gives the *real* transfer rate, i.e., taking into account experiments with higher priorities. The experiment with highest priority uses the bandwidth proportionally to the ratio between its expected transfer rate $p(\tau_k^t)$ and the maximum transfer rate $\tau_{r(k,t)}$. Then the residual bandwidth is attributed using recursively the same rule.

Finally, equations 4 and 5 link the usage of the different memories to the sum of the in and out transfer rates (π_k and τ_k are used here as functions of t).

In Figure 3 we show the difference between the two models.

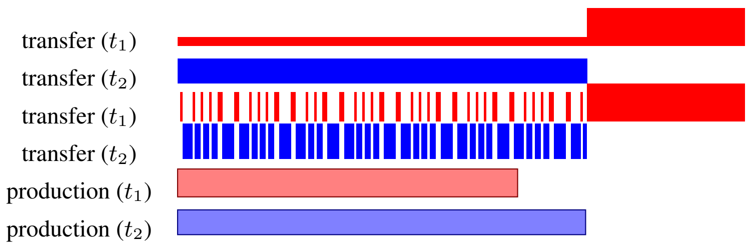

Fig. 3: Comparison of the two representations: two data-producing activities t_1 and t_2 (bottom); The “exact” view of the corresponding transfers, sharing the transfer bus because of gaps due to the low data-producing rate (middle); The alternative reformulation, where this is modeled as sharing the bandwidth (top).

4 Checking and Filtering Algorithms

In this section we introduce a filtering procedure for the DATATRANSFER constraint. We first introduce an efficient $O(nm \log(nm))$ procedure for computing transfers and memory usage of a given schedule. This procedure execute a *sweep* of the horizon similar to that described in [3]. Besides checking whether the constraint is violated we shall also use this algorithm to compute lower bounds in order to filter the domains.

4.1 Data Transfer Verification

Given a complete schedule of the data-producing activities, and a priority ordering, we now describe an algorithm that computes the effective transfer rate (in the sense of Definition 1) and the memory usage for each experiment over the whole horizon in time $O(nm \log(nm))$. Notice that both are step functions, moreover we will see that there are at most $O(nm)$ breaking points, so they can be stored on $O(nm)$ bits. This algorithm can be used to verify whether an assignment is consistent by simply checking that the usage of all experiments remains within the memory's capacity. We shall also use it to compute bounds on the memory usage of extreme scenarios (e.g., all tasks set to their earliest start time). It sweeps the time horizon chronologically, computing variations of various parameters only when certain *events* occur.

First, we build the list of events. Each event is time tagged and there are six types (for $O(nm)$ events in total): *Start/end of visibility*; *Start/end of a data-producing activity*; *Start/end of experiments*. Then, we sort them in chronological order and explore them in that order. For each time point t where at least an event occurs, we go through all events occurring at t and update the following arrays accordingly:

– $visibility$ stands for whether there is a visibility line at time t. It is flipped whenever encountering a "Start of visibility" or "End of visibility" event;
– $production(k)$ stands for the data-producing rate of experiment E_k at time t. It is increased (resp. decreased) by the data-producing rate of the activity whenever encountering a "Start of production" (resp. "End of production") event;
– $active$ stands for the number of active experiments at time t. It is increased (resp. decreased) by one whenever encountering a "Start of experiment" (resp. "End of experiment") event.

At each step of the loop, we therefore know the complete state (data-producing rate on each experiment, whether we are in visibility or not, and how many experiments are active). Moreover, we also keep track of the memory usage with another array: $memory$. We then compute what are the current transfers, and partition the bandwidth between them. For each experiment E_k (visited by order of priority), if it has data on memory, or if it is currently producing data, and if the bandwidth is not null, we create a transfer. We first compute its potential transfer rate $\tau_{r(k,t)}$ according to the rules described above. If it has some data on memory, all of the bandwidth is attributed to this transfer. Otherwise, its actual transfer rate is equal to the minimum between the nominal transfer rate and the current data-producing rate: $\tau = \min(production(k), \tau_{r(k,t)})$. The ratio $\tau/\tau_{r(k,t)}$ of the bandwidth is allocated to this transfer.

Then, for each experiment currently in transfer, we compute a theoretical deadline, i.e., the date at which it will be emptied at this rate of transfer if nothing changes. Notice that it can be *never*. Similarly, we compute a theoretical deadline for filling the mass-memory. If the earliest of all these deadlines happens earlier than the next scheduled event, we add it to the list of events. This type of events will do nothing on its own, however, it will allow the algorithm to recompute the transfers according to the new situation (the mass-memory being filled, or an experiment's memory being empty).

Finally, the usage of each memory at time t is updated according to the transfers.

This algorithm has a worst case time complexity of $O(nm \log(nm))$. The list of events has initially $O(nm)$ elements. There are two for each data-producing task, two for each visibility window, and two for each experiment (we assume that the number of visibility windows is less than nm). Sorting them can therefore be done in $O(nm \log nm)$ time. In the main loop, events are processed only once, and this takes at most $O(m)$ time. Moreover, in some cases, "deadline" events can be added during the exploration of the event list. However, at most one such event can be added for each event initially in the list. Indeed, consider a deadline event. It is added to the list only if no other event yet to process has an earlier date. In other words, transfer and data-producing rates as well as visibility do not change. The experiment memory that was emptied will therefore stay empty at least until the next standard event. The same is true for deadline event triggered by filled mass-memory: it will stay full at least until the next visibility event. Therefore, the worst case time complexity of the main loop of this algorithm is $O(nm)$.

4.2 Filtering Rules

In this section introduce two propagation rules for the DATATRANSFER constraint.

Minimal transfer span: The first rule tries to guess a lower bound on the total span of a subset of activities of the same experiment E_k. The intuition is that if data is produced at a higher rate than it can be transferred out, the capacity of a memory could be reached and data will be lost. In other words, given a set $\Omega \subseteq E_k$ of data-producing activities of an experiment E_k, the total amount of data produced by these activities, minus what can be stored on the memory of E_k, need to be transferred out. The duration of this transfer is a lower bound on the span of this set of activities, i.e., the duration between the minimum start time and maximum end time of any activity in this set.

The total amount of data produced by activities in Ω is equal to $\sum_{t_{ki} \in \Omega} \pi_{ki} p_{ki}$. At most M_k can be stored on the experiment's own memory, hence at least $\sum_{t_{ki} \in \Omega} \pi_{ki} p_{ki} - M_k$ has to be transferred out *before* the end of the last data-producing activity. Let τ be the highest possible transfer rate for data out of the experiment's own memory. We can use this rate to derive a lower bound on the total duration of Ω:

$$(\max_{t_{ki} \in \Omega}(e_{ki}) - \min_{t_{ki} \in \Omega}(s_{ki})) \geq \frac{\sum_{t_{ki} \in \Omega} \pi_{ki} p_{ki} - M_k}{\tau} \qquad (6)$$

In real scenarios, data-producing activities of a given experiment cannot overlap, and in many cases the order is known a priori. Assuming that the activities in Ω are ordered, with t_{kf} being the first task and t_{kl} being the last task in Ω, we can often induce the simpler constraint: $e_{kl} - s_{kf} \geq \frac{\sum_{t_{ki} \in \Omega} \pi_{ki} p_{ki} - M_k}{\tau}$.

Example 1. Figure 4 depicts the application of this rule. We have two activities t_1, t_2, the former producing $\pi_1 = 5$ blocks/sec and the latter $\pi_2 = 4$ blocks/sec, both for 70 seconds. Therefore, $\pi_1 p_1 + \pi_2 p_2 = 630$ blocks are produced. Assume that the memory of this experiment has a capacity of 250 blocks. Consequently, 380 blocks need to be transferred out in order to avoid data loss. Since the maximum transfer rate is 2 blocks/seconds, this transfer will take at least 190 seconds. We can conclude that the end of t_2 is at least 190 seconds after the start of t_1. The grey scale gives the evolution of the memory for t_2 finishing exactly 190 seconds after the start of t_1.

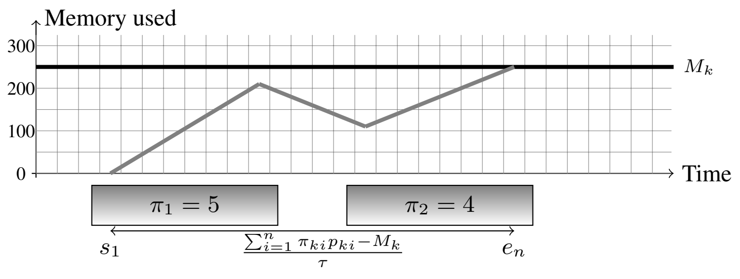

Fig. 4: Example of minimal span constraint.

Moreover, we can take into account the data produced by activities of experiments of higher priority, since their data transfers will preempt those of lower priority.

Consider an interval of time $[a, b]$. Any data produced by experiments of higher priority during this period must be transferred out before E_k can be allowed to transfer.

Let $\min(|t_{ki} \cap [a, b]|)$ be the minimum size of a common interval between $[a, b]$ and $[s_{ki}, s_{ki}+p_{ki}]$ for any value of s_{ki}. If $|[a, b] \cap [c, d]|$ stands for the size of the intersection of intervals $[a, b]$ and $[c, d]$, then :

$$\min(|t_{ki} \cap [a, b]|) = \min(|[a, b] \cap [\min(s_{ki}), \min(s_{ki}) + p_{ki}]|, |[a, b] \cap [\max(s_{ki}), \max(s_{ki}) + p_{ki}]|)$$

We can compute a lower bound $T_k(a, b)$ on the time required to transfer the data produced by experiments of higher priority than k on the interval $[a, b]$ as a lower bound on the data produced, divided by the maximum transfer rate:

$$T_k(a, b) = (\sum_{j=1}^{j<R(k)} \sum_{i=1}^{n} |t_{P(j)i} \cap [a, b]| * \pi_{P(j)i})/\tau$$

Given a subset $\Omega \subseteq E_k$ of experiment E_k, consider the time interval $[a, b]$ between the latest start time of any task in Ω ($a = \max_{t_{ki} \in \Omega}(\min(s_{ki}))$) and the earliest end time of any task in Ω ($b = \min_{t_{ki} \in \Omega}(\max(s_{ki}) + p_{ki})$). The lower bound on the span given above assumes continuity of the transfer, and by definition this duration must include the interval $[a, b]$. Therefore, any interruption of the transfer during this period induces the same delay on the minimal span of Ω. In other words, any time taken to transfer data of experiments with higher priority during $[a, b]$ ($T_k(a, b)$) can be simply added to the lower bound above.

Hence we can tighten the constraint 6 as follows (with $a = \max_{t_{ki} \in \Omega}(\min(s_{ki}))$ and $b = \min_{t_{ki} \in \Omega}(\max(s_{ki}) + p_{ki})$):

$$(\max_{t_{ki} \in \Omega}(e_{ki}) - \min_{t_{ki} \in \Omega}(s_{ki})) \geq \frac{\sum_{t_{ki} \in \Omega} \pi_{ki} p_{ki} - M_k}{\tau} + T_k(a, b) \qquad (7)$$

We apply this rule for every set of consecutive activities (with respect to their earliest start times) of every experiment. There are $n^2 m$ such sets, and computing the lower bound takes at most $O(nm)$ time. The whole procedure hence has a worst case time complexity of $O(n^3m^2)$.

Mass memory saturation: Since transfers from the lander to the orbiter are possible only during visibility, the data can only accumulate on the mass-memory while not in visibility. As a consequence, the period that precedes a visibility window is critical since the mass memory can be saturated hence blocking all transfers. When this happens, data produced by an experiment remains on its memory at least until the next visibility window, and it is possible to lose data when the experiment's memory itself is saturated.

We use this observation to deduce that data-producing activities that would generate too much data to hold on the mass memory and on their own memory should be either advanced or postponed. Suppose that we know that at time $\bar{t}$, the mass-memory will necessarily be filled. It will remain so until the next visibility. Now, if an activity t_{ki} produces more data in the interval between $\bar{t}$ and the next visibility window than its own memory can hold, it will be lost. Indeed, no data can be transferred onto the mass-memory as long as it is full, and it will start to be emptied only when the visibility allows it. We can thus deduce that the activity t_{ki} must start either early enough to produce before $\bar{t}$ or late enough so that the data in excess will be produced during the visibility period (in order to have a chance to be transferred).

First, we show how to compute an upper bound $\bar{t}$ on the time when the mass-memory will reach its maximum before a given visibility window. We consider a single visibility cycle $\mathcal{V} = (a, v, b)$, where $a < v < b$ denote, respectively, the end of the previous cycle, the start of a visibility window, and the end of that visibility window. Let $\Omega(\mathcal{V})$ be the set of data-producing activities that are necessarily scheduled within the interval $[a, b]$.

Proposition 1. *Scheduling all data-producing activities in $\Omega(\mathcal{V})$ to their latest start time minimizes the memory usage of the mass-memory (m_0^t) for all $t \in [a, b]$.*

Proof (sketch). Clearly if we consider a data-producing activity t_{ki} in isolation, setting its start time to the latest possible time point $(\max(s_{ki}))$ delays the transfer onto the mass memory hence its memory usage for any time point in $[a, b]$.

When multiple data-producing activities can run in parallel, experiments of high priority can preempt transfer intervals of experiment of lower priority. Therefore, one could advance a data-producing activity t_{jl} in time in order to use the resource and therefore delay the transfer of some of the data produced by t_{ki}. However, since the transfer rate increases with the priority, for any time interval where the transfer of the data produced by t_{jl} preempts that produced by t_{ki}, data is being transferred to the mass memory at a higher rate. Thus, advancing a data-producing activity t_{ji} of higher priority never helps minimizing the mass memory usage. $\square$

Given a visibility cycle $\mathcal{V} = (a, v, b)$, we can therefore get a lower bound on m_0^t on the usage of the mass memory for any t in the interval $[a, b]$ using the sweep algorithm. For every task in $\Omega(\mathcal{V})$, we tentatively fix it to its latest start time and execute the sweep algorithm. Hence, we can easily compute $\bar{t}$, the smallest value of t for which $m_0^t = m_0^v$.

Given an experiment E_k. We can bound the amount of data that can be produced by any task of this experiment in the period $[\bar{t}, v]$ *and* stored without loss. There are $m_0^{\bar{t}}$

blocks of data already on the mass-memory, so $M_0 - m_0^{\overline{t}}$ is free. Moreover, up to M_k can be stored on the experiment's own memory, for a total of $\delta_k = M_0 + M_k - m_0^{\overline{t}}$ blocks. Above this threshold, data produced by activities of experiment E_k between $\overline{t}$ and v will be lost. If $|t_{ki} \cap [\overline{t}, v]|$ stands for the length of the overlap between an activity t_{ki} and the interval $[\overline{t}, v]$, an activity t_{ki} produces $|t_{ki} \cap [\overline{t}, v]| * \pi_{ki}$ blocks of data in the interval $[\overline{t}, v]$. Therefore, the following relation must hold: $\sum_{i=1}^{n} (\min(|t_{ki} \cap [\overline{t}, v]|) * \pi_{ki}) \leq \delta_k$ from which we can deduce the following implied constraint:

$$|t_{ki} \cap [\overline{t}, v]| \leq (\delta_k - \sum_{j \neq i \in [1,n]} (\min(|t_{kj} \cap [\overline{t}, v]|) * \pi_{kj}))/\pi_{ki} \qquad (8)$$

We run the sweep algorithm once to obtain the value of $\overline{t}$. Then, for each experiment, we can compute δ_k and in time $O(n)$ the values of $\min(|t_{kj} \cap [\overline{t}, v]|$ for each activity t_{ki}. Finally we compute the implied constraint also in time $O(n)$ (it takes constant time for each activity, once $\min(|t_{kj} \cap [\overline{t}, v]|$ is known). Finally we apply it only when it collapses to a simple lower or upper bound on the start time s_{ki} of an activity t_{ki}. The total time complexity of this filtering rule is thus $O(nm \log(nm) + nm) = O(nm \log(nm))$.

5 Experimental Results

All the previous algorithms and filtering rules have been implemented on the latest version of MOST. We ran experimentations on different scenarios provided by the group SONC of CNES. Each scenario consists in one, two or three experiments which must be scheduled on a time window between 10 hours and 1 day. For each subset of experiments, several variations are tested in order to assess uncertain parameters. For instance, the visibility cycle depends on the exact mass and shape of the comet, the orbit selected by Rosetta, and the landing site chosen for Philae, all of which are unknown. Some scenarios have continuous visibility, while other have different periods for the visibility cycles. The hardware onboard the probe will have travelled in extreme temperatures for ten years, so the exact charge and efficiency of the batteries is also uncertain. Moreover, engineers of SONC test a range of variations on other parameters such as the memory capacity simply to stress-test the system (MOST).

5.1 Search effort

We ran 8 scenarios and compared the results of the current version of MOST against the ad-hoc propagator introduced in this paper. Both were run on quad-core Sun $T5120$ running Solaris 2.10 with $8GB$ of RAM. The current version of MOST (denoted MOST+ILCRESERVOIR) models data transfers using Ilog-Scheduler ILCRESERVOIR constraints. In our version (denoted MOST+DATATRANSFER) we use only the first filtering rule described in Section 4.2.[4]

We report the results in Table 1. We present for each scenario the set of experiments involved, the memory capacities, and whether the visibility is continuous or not. Then

[4] The second filtering rule was not implemented when the experiments were run.

Scenario	Parameters			MOST+ILC-RESERVOIR			MOST +DATATRANSFER		
	M_k	M_0	Visi.	Fail	Init. time	Search time	Fail	Init. time	Search time
Consert	500	17456	Periodic	295	$4,06$	$20,07$	0	$0,88$	$0,08$
Consert/Romap	$500/250$	17456	Periodic	7112	$11,13$	Time out	0	$1,17$	$0,1$
Consert/Romap	$500/250$	37456	Periodic	7051	$11,03$	Time out	0	$1,17$	$0,1$
SD2/Ptolemy	$64/2000$	17456	Periodic	234	$26,71$	$41,72$	0	$3,37$	$0,09$
SD2/Ptolemy	$64/2000$	17456	Continuous	211	$32,78$	$79,48$	0	$3,25$	$0,08$
SD2/Cosac/Civa	$64/24000/4000$	37456	Periodic	407	$50,20$	$181,91$	0	$2,75$	$0,14$
SD2/Cosac/Civa	$64/24000/4000$	17456	Periodic	413	$50,84$	$179,19$	0	$2,95$	$0,15$
SD2/Cosac/Civa	$64/24000/4000$	17456	Continuous	390	$25,12$	$91,08$	0	$1,82$	$0,10$

Table 1: Old vs. new version of MOST on 8 standard scenarios

we give the number of fails calculated by Ilog-Scheduler during search, the initialization time and finally the solving time.

We observe first that using our approach, solutions can be obtained without any fail, whereas the previous model explored a much larger search tree. The reformulation using ILCRESERVOIR constraints was indeed very loose, and did not allow to detect inconsistencies early. Moreover, to overcome this weakness, the scenarios produced by the group SONC are overly constrained in order to cut possibilities and allow the solver to converge more easily. Moreover, our propagator is relatively light and therefore more time effective, compared to the model using a large amount of transfer tasks throughout the horizon for each reservoir constraint.

In fact, the model was so large that the initialization time is very high. The few seconds of initialization time in our approach correspond to the rest of the model (cumulative and unary resources) which is common to both implementations.

In two cases, no solution was found by MOST+ILCRESERVOIR within the 600 seconds time cutoff. However, this is not explained (only) by performance issues. In fact, these two scenarios do not have a valid solution under the old model, whereas they are feasible.

5.2 A more accurate modeling

In MOST+ILCRESERVOIR, activities with very low data-producing rate are treated differently because of rounding issues: It is assumed that the data is produced all at once at the end of the activity. Therefore in these scenarios, transfers can be delayed by a substantial amount compared to the real behavior of the CDMS.

Moreover, since transfer tasks have a frequency of 120 seconds, they cannot accurately model situations where the CDMS frequently switches between different transfers. The scenario Consert/Romap highlights this problem on SONC's version of MOST. Both experiments have small data-producing activities and small memory capacities. Therefore, switches between transfers from these two experiments are extremely frequent. However, with MOST+ILCRESERVOIR it is not possible to switch frequently enough, hence there is no solution.

Figure 5 is a screenshot of the MOST's GUI showing a zoom on a plan (a solution) of the scenario SD2/Ptolemy. The bottom bars represents the transfer of Ptolemy, and the bars just above are data-producing activities. We can see that the transfer task does

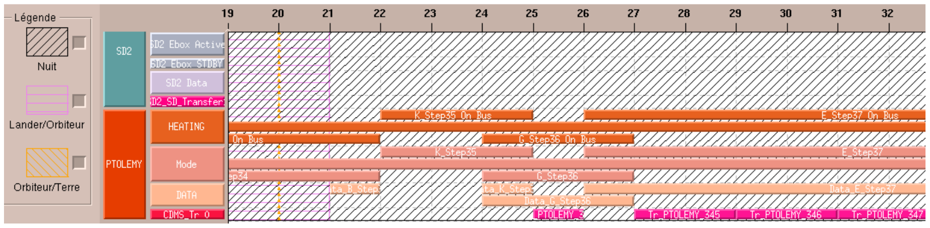

Fig. 5: Example of MOST+ILC-RESERVOIR

not coincide with the data-producing. Indeed there is a gap, because data-producing rate is too low to trigger a transfer task earlier.

6 Conclusion

In this paper we have presented an application of constraint programming for the international spatial mission ROSETTA/PHILAE. We have identified that the main problem is the management of data transfers and in particular, data loss. We shown that the previous constraint programming approach was not well-adapted to this problem and we introduced a global constraint to forbbid data-loss. In particular we proposed an efficient sweep algorithm which checks and computes the feasibility of data transfers. We also have presented two propagation rules for the data transfer constraint. Overall, our approach greatly improves the results both for computing times, and accuracy of the solutions.

References

1. Abderrahmane Aggoun and Nicolas Beldiceanu. Extending chip in order to solve complex scheduling and placement problems. *Mathematical and Computer Modelling*, 17(7):57 – 73, 1993.
2. Grégory Beaumet, Gérard Verfaillie, and Marie-Claire Charmeau. Feasibility of autonomous decision making on board an agile earth-observing satellite. *Computational Intelligence*, 27(1):123–139, 2011.
3. Nicolas Beldiceanu and Mats Carlsson. Sweep as a generic pruning technique applied to the non-overlapping rectangles constraint. In *CP 2001*, pages 377–391. 2001.
4. Robert Morris Jeremy Frank, Ari Jónsson and David E. Smith. Planning and Scheduling for Fleets of Earth Observing Satellites. In *6th i-SAIRAS*, pages 18–22, 2001.
5. Philippe Laborie. Algorithms for propagating resource constraints in ai planning and scheduling: existing approaches and new results. *Artif. Intell.*, 143(2):151–188, February 2003.
6. Catherine Mancel and Pierre Lopez. Complex Optimization Problems in Space Systems. In *13th International Conference on Automated Planning & Scheduling (ICAPS'03)*, 2003.
7. Wim Nuijten Philippe Baptiste, Claude Le Pape. *Constraint Based Scheduling*. Springer, 2001.