Constraints

Scheduling Scientific Experiments for Comet Exploration

--Manuscript Draft--

Manuscript Number:	CONS-D-13-00027
Full Title:	Scheduling Scientific Experiments for Comet Exploration
Article Type:	Application Paper (Helmut Simonis)
Keywords:	Resource Scheduling, Global Constraints
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 tasks, and incompatibility between instruments. Moreover, a very important aspect is related to the transfer (to the orbiter then to the 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 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 tasks are scheduled in such a way that no data is lost. Thanks to this constraint and to the filtering rules we propose, mission control is now able to compute feasible plans in a few seconds for scenarios where minutes were previously often required. Moreover, in many cases, data transfers are now much more accurately simulated, thus increasing the reliability of the plans.

Scheduling Scientific Experiments for Comet Exploration

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 tasks, and incompatibility between instruments. Moreover, a very important aspect is related to the transfer (to the orbiter then to the 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 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 tasks are scheduled in such a way that no data is lost.
Thanks to this constraint and to the filtering rules we propose, mission control is now able to compute feasible plans in a few seconds for scenarios where minutes were previously often required. Moreover, in many cases, data transfers are now much more accurately simulated, thus increasing the reliability of the plans.

Keywords: global constraint, scheduling, data transfer, energy and memory constraints, space experiments

1 Introduction

Following the fly-by of the comets Halley and Grigg-Skjellerup by the spacecraft Giotto, an even more ambitious mission, including the landing of a robotic module on the comet nucleus, was approved by European Space Agency in 1993. This project involves more than 50 contractors from 14 European countries, Canada and the United States for developing the instruments necessary to a deeper study of the comet. The *Rosetta* spacecraft, embarking these scientific instruments, was then launched in 2004 by Ariane 5, and was set to travel more than six billion kilometers to finally reach the comet $67P$/Churyumov-Gerasimenko in 2014. Its complex trajectory includes four

gravity assist maneuvers (three times around Earth and once around Mars) before finally reaching the comet and enter its orbit. During its travel, the probe has met two asteroids (Steins and Lutetia), and collected data and pictures. Upon arrival at $67P$, Rosetta will enter orbit around the comet and follow it on its journey towards the Sun. Finally, a lander module, called *Philae*, will then be deployed and attempt the first ever landing on the surface of a 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. The instruments are designed to measure the molecular, mineralogical, and isotopic composition of the comet's surface and subsurface material, and also to measure characteristics of the nucleus such as near-surface strength, density, texture, porosity, ice phases and thermal properties. For instance, *ÇIVA* and *ROLIS* are two imaging instruments, used to take panoramic pictures of the comet and microscopic images, whilst the Alpha Proton X-ray Spectrometer (*APXS*) instrument analyzes the chemical composition of the landing site and its potential alteration during the comet's approach to the Sun. The obtained data 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. First, SDL (*Separation-Descent-Landing*) will run for 30 minutes during which many experiments will be done. Second, FSS (*First Science Sequence*) will last 5 days. This phase is critical because the execution of the most energetically greedy experiments requires battery power. The quality of the schedule conditions the longevity of the batteries and is therefore a key to the success of the mission. Finally, during the LTS phase (*Long Term Science*), scientific tasks 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[1] in Toulouse (France). The goal of the *Science Operations 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) NetLander program [6]. A first software (called MOST) has been developed on top of the Ilog-Scheduler/Solver library by an industrial subcontractor. Every instrument 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 experiments of the FSS and LTS and the maneuvers of the SDL are modeled in MOST as a scheduling problem with two main types of resources.

First, the tasks within the experiments concurrently use the energy from a centralized source, mainly from batteries, and up to some extent from solar panels. There is an upper limit on the instant power that can be delivered to the experiments. Moreover, experiments are linked to the batteries through several power lines, themselves linked to a different converter. Each line and each converter entails another instant power threshold. Within MOST this is modeled as a hierarchy of CUMULATIVE constraints [1], one for the total instant power, one for each converter, and one for each line.

[1] Centre National d'Etudes Spatiales

Second, each experiment produces data that must be transferred back 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. The orbiter thus acts as a relay and transfers the data to Earth (this also requires visibility between Rosetta and Earth[2]). 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) following a greedy rule: priorities are assigned off-line to experiments. Then, on-line, data is systematically transferred from the experiment of highest priority with available data to the mass-memory, unless the mass-memory is full, in which case all transfers are blocked. This transfer policy 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 production tasks fill the reservoir, while multiple pre-defined data transfer tasks of variable duration empty it.

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. 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 tasks (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 tasks. 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 formally define a new global constraint and oppose exact and approximate models to transfer blocks of data. In Section 4 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] Nasa and Russian relays are used to cover the whole surface.

2 Problem Description

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

Disjunctive resources: Each task of an experiment may require during its processing one or several instruments. Several tasks using the same instrument cannot be scheduled simultaneously, which correspond to a standard disjunctive resource.

Precedences: Tasks within the same experiment might have precedence constraints; for instance the lander also carries a Sampling Drilling and Distribution device (*SD2* instrument), which will drill more than 20 cm into the surface, collect samples, and deposit them in different ovens, which are mounted on a circular and rotatable carousel. Four ovens of this carousel are dedicated to the Ptolemy instrument. With an appropriate sample loaded into one of the ovens, the carousel rotates to a position whereby a device referred to as a "tapping station" is used to connect the oven to the inlet of the gas management system of the Ptolemy instrument. At this point, the oven must be heated so that volatile samples are analyzed by Ptolemy. In this typical experiment, we have three tasks using three different instruments, each task preceding each other w.r.t. standard precedence constraints.

Cumulative Resources: Tasks within the experiments concurrently use the energy from a centralized source, mainly from batteries and up to some extent from solar panels. All the energy sources (batteries and solar panels) are centralized on a main power line (see Figure 1). The energy needed to run each task is supplied by an auxiliary power line. Each auxiliary line is linked to a converter, and each converter is linked to the main power line. At each level, the total instant power delivered cannot exceed a given threshold. For each auxiliary power line, all the tasks supplied by this line are constrained by a CUMULATIVE constraint [1] with capacity equal to this threshold. Similarly another CUMULATIVE constraint is associated to each converter, and a last one is associated to the main power line, involving all tasks of the problem.

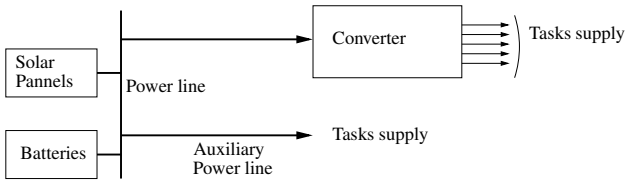

Fig. 1: Illustration of the global power system

State Resources: In addition to instrument availability, which is modeled with a disjunctive constraint, each instrument can have multiple states along the schedule. Some

tasks can trigger the modification of the state of an instrument, and the processing of certain tasks might be subject to some instruments having a given state. This is modeled using state resource constraints in Ilog-Scheduler.

Data Transfer and Memory Constraints: Every experiment has its own memory. Some tasks produce data, temporarily stored on the experiment's memory. Then this data will be transferred onto the mass-memory and subsequently to the orbiter. Note that for a given experiment, the precedence constraints between tasks are such that data-producing tasks do not overlap. Therefore, at every time only one task of each experiment shall write in the experiment's memory.

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 to the orbiter. 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.

The Mars Express mission, launched in 2003 and still in operation, also featured a data transfer planning problem, similar to Rosetta's in many respects. In both cases, data-producing tasks are to be scheduled, data is kept into a number of memory storage devices on board and periodically transmitted to the Earth during visibility windows. However, a critical difference is that in Mars Express, the transfers are actually decisions to be made at the planning level. A flow model was proposed to address the so-called Memory Dumping Problem in [4,7] and further improved in [8]. In our case, however, the CDMS policy is given. To be more accurate, it is actually possible to change the experiments' priorities. However this cannot be done at any time since it corresponds to a global configuration change of the CDMS. Hence, only a few priority changes can be allowed in the planning horizon, which yields the need of considering the fixed priority case. When priorities are fixed, data loss can only be controlled through the schedule of data-producing tasks.

In other words, we shall consider data transfers as a global constraint on the start times of tasks ensuring that no data will be lost with respect to the CDMS policy.

3 A Global Constraint for Data Transfer

Except for data transfer, all the constraints above can be modeled using the standard methods and algorithms [2], all available in Ilog-Scheduler. Hence, we focus on data transfers and propose a global constraint to reason about this particular aspect of the problem.

From now on, we consider a set $\{E_1, \ldots, E_m\}$ of m experiments. An *experiment* $E_k = \{t_{k1}, \ldots, t_{kn}\}$ is a set of data-producing tasks[3], and is associated with a memory of capacity M_k. A task t_{ki} produces data for a duration p_{ki} at a rate π_{ki} in the experiment's memory. The lander possesses a mass memory of capacity 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 $P(i)$ the index of the experiment coming 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 $R(k) < R(j)$.

For reasons that will be explained later, the rate of the transfer from the experiment memory to the mass memory is not fixed and depends on three factors: (i) the number of *active* experiments, (ii) the relative priority of the experiment being transferred with respect to other active experiments, and (iii) the presence of a simultaneous transfer from the mass memory to the orbiter. An experiment is said to be active if its first task has already started and its last task has not finished, or if it has some data in memory. When there are simultaneous transfers to and from the mass-memory, the transfer rate from experiments is lower. These parameters are function of time, so we define a predicate $\tau_{(k,t)}$ that gives the transfer rate from an experiment E_k to mass-memory according to its priority, to the number of active experiments, and to the presence of a transfer to the orbiter at time t. However, transfers between mass-memory and the orbiter have a constant rate denoted by τ^{orbiter}.

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 as a set of intervals $\{[a_1, b_1], \ldots, [a_v, b_v]\}$ 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. A data-producing task therefore entails as many transfer tasks as blocks of data it produces.

We consider the following decision variables: $s_{11}, \ldots, s_{mn}$, with domain in $[0, \ldots, H]$, standing for the start times of data-producing tasks $\{t_{11}, \ldots, t_{mn}\}$, respectively. The fact that data loss should be avoided can be seen as a relation (i.e., a constraint) between these decision variables. 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 tasks, one can unroll the rules outlined in Section 2 and further detailed in Section 3.1 to check whether the CDMS policy will lead to data loss or not.

First, in Section 3.1 we discuss an "exact" definition of this constraint 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. And finally, in Section 3.3 we study the worst case approximation error of this alternative model.

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

3.1 CDMS Policy

In this section, we detail the CDMS policy, then we define a constraint modeling the relation between the start time of data-producing tasks induced by this policy. 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 quantity of data in the mass-memory.

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).

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 time-dependent duration of the transfer tasks come from time-dependent transfer rates $\boldsymbol{\tau_{(k,t)}}$. Blocks are actually 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 task and the end time of its last task, 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. Furthermore, transfers from the experiment memory are slower when there are simultaneous transfers from the mass memory to the orbiter. To emulate this, we use variable transfer rates. The transfer rate $\boldsymbol{\tau_{(k,t)}}$ is actually read in a table $(\tau^{\text{observed}}_{k,x,y,z})$, which entries were measured experimentally and give the transfer rate for experiment k depending on the number x of active experiments, on its relative priority y among them and on the Boolean presence z of a transfer to the orbiter, at the considered time t.

More precisely, $\boldsymbol{\tau_{(k,t)}} = \tau^{\text{observed}}_{k,|X(t)|,y(k,t),z(t)}$, where the set of active experiments at time t is defined by

$$X(t) = \{k \,|\, \exists i, s_{ki} \leq t \leq s_{ki} + p_{ki} \vee m_k^t > 0\}$$

The relative priority of experiment k at time t is defined by

$$y(k,t) = |\{k' \in X(t) | R(k') \leq R(k)\}|$$

The presence of a transfer to the orbiter is defined by

$$z(t) = V(t) \wedge m_0^t > 0$$

It can be easily checked that $y(k,t) \in \{1, \ldots, |X(t)|\}$. However, it has been somewhat counterintuitively observed that the transfer rate between mass-memory and the orbiter can be considered constant (denoted by $\boldsymbol{\tau}^{\text{orbiter}}$).

We can define a constraint DATATRANSFER($[s_{11}, \ldots, s_{mn}]$) ensuring that the schedule of tasks $\{t_{11}, \ldots, t_{mn}\}$ is such that no data is lost according to the CDMS rule described above.

However, 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, mass-memory is unlimited, and production rates are equal to transfer rates. Since the capacities are all equal to one block, each block produced must be immediately transferred. Moreover, since production and transfer rates are equal, a transfer will stop exactly when production stops. 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.

Moreover, an "exact" formulation is not practical, as considering the transfer of each block of data would be too difficult. Therefore, we propose an alternative model in the next section. The basic idea is to represent all the data produced by a task as a continuous quantity.

3.2 Approximated Definition

We have seen that it is difficult to capture very precisely the behavior of the CDMS. When we consider a data-producing task in isolation, the number and frequency of the transfer tasks is easy to compute. However, when we consider several data-producing tasks with different priorities and unknown start times, this viewpoint becomes impractical. We therefore propose an alternative model that approximates very closely the amount of transferred data with a reasonable time and space complexity.

The basic idea is to consider data produced by a task as continuous quantity. This idea is straightforward in the case where experiments fully use the transfer bus. Indeed, consider a task t_{ki} that produces more data than it can transfer: $\tau_{(k,t)} \leq \pi_{ki}$, with $\tau_{(k,t)}$ 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 $\frac{\pi_{ki} p_{ki}}{\tau_{(k,t)}}$. It is therefore easy to compute how the usage of the memory will be impacted by this transfer. However, when taking into consideration priorities, variable transfer, and production rates, it becomes significantly more complex. We list here three difficulties.

First, as explained in the previous section, we have time-dependent transfer rates $\boldsymbol{\tau_{(k,t)}}$ (from the experiment to the mass memory) due to the scanning process of the AETS procedure.

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 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 $\pi_{ki} - \tau_{(k,t)}$ during p_{ki} seconds. Similarly, during $\frac{\pi_{ki} p_{ki}}{\tau_{(k,t)}}$ seconds the usage of the mass memory increases at rate $\tau_{(k,t)}$.

The third difficulty concerns tasks producing data at a lower rate than the possible transfer rate (i.e., $\tau_{(k,t)} > \pi_{ki}$). 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 Figure 2).

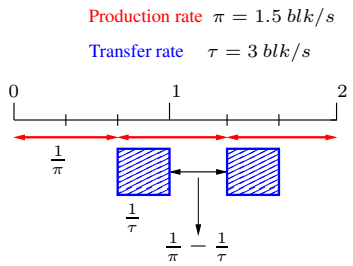

Fig. 2: Exact transfer model

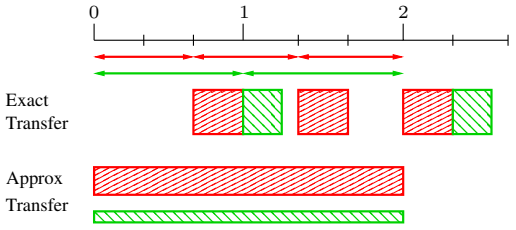

Fig. 3: Example of two data transfer tasks with both model

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 $\frac{\pi_{ki} p_{ki}}{\tau_{(k,t)}}$ seconds, however other transfers 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 $\min(r, \frac{\pi_{ki}}{\tau_{(k,t)}})$, where r stands for the remaining bandwidth. Bandwidth is allocated recursively, according to this demand function and to priority (see Figure 3).

In other words, we approximate the "vertical" partition of the data bus shown in Figure 3 (exact transfers) by a "horizontal" partition, i.e., we consider that the bus has a bandwidth that can be divided over parallel transfers.

We shall see that 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.

We can now formally define the constraint which is satisfied if and only if data production tasks are scheduled such that the approximated view of the CDMS policy would not entail data loss. Let π_k^t stand for the data-producing rate on experiment E_k at time t, let $p(\tau_k^t)$ 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 (approximate) transfer rate from experiment E_k to the mass memory at time t.

Definition 1.

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

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

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

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

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

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

$$\forall t, k, \quad m_k^t \leq M_k \tag{6}$$

Equation 1 states that if a data-producing task 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 *potential transfer rate* $p(\tau_k^t)$ of experiment E_k at time t if it is not trumped by other experiments of higher priority. The first case correspond to a a full mass memory. When this happens, all transfers are stopped, hence the potential transfer rate is null. The second case corresponds to the transfer of data hold on the experimental memory. In this case, it can be transferred at the maximum available rate ($\tau_{(k,t)}$). The last case corresponds to data being transferred as soon as it is produced. In this case, the experimental memory is empty ($m_k^t = 0$) and will remain so, if the production rate is smaller than or equal to the possible transfer rate ($\pi_k^t \leq \tau_{(k,t)}$). Otherwise, data will build up onto the experiment memory, and will be transferred at rate $\tau_{(k,t)}$.

Equation 3 gives the *real* transfer rate (in the approximation scheme), that is, taking into account experiments with higher priorities. The experiment with highest priority uses the bandwidth proportionally to the ratio between its potential transfer rate $p(\tau_k^t)$ and the maximum transfer rate $\tau_{(k,t)}$. Then the residual bandwidth is assigned 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) while Equations 6 ensure that memory capacity is never exceeded.

Figure 4 illustrates the difference between the two models.

3.3 Approximation Error

Here we study the worst case approximation error of the "bandwidth" model. We shall refer to the "real" behavior of the CMDS as the **exact model**, whereas the formulation given in Definition 1 shall be referred to as the **approximate model**. Let $\boldsymbol{m_k^t}$ be the load of the memory of experiment E_k in the exact model, and let $\boldsymbol{m_k^{t\prime}}$ be the load of the

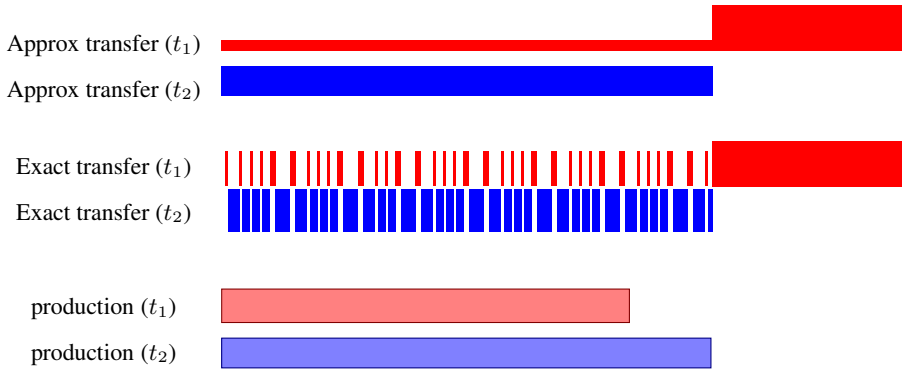

Fig. 4: Comparison of the two representations: two data-producing tasks t_1 and t_2 (bottom); The "exact" view of the corresponding transfers, sharing the transfer bus because of gaps due to the low data-producing rate (middle); The alternative reformulation, where this is modeled as sharing the bandwidth (top).

memory of experiment E_k in the approximate model. We denote by $\boldsymbol{\Delta_k^t = m_k^t - m_k^{t\prime}}$ the discrepancy between the two models. We will show that for all values of t and k, the absolute error $|\Delta_k^t|$ is bounded by $1+\alpha$, where α is the maximum ratio between two transfer rates in the transfer matrix τ^{observed}, for a fixed number of active experiments. Usually, the difference in transfer rates comes from side effects of the CDMS policy and is close to 1.

In order to simplify the proofs, we will use the following notation: π_k (resp. τ_k) for π_k^t (resp. τ_k^t) when the rates are constant over time, and simply π (resp. τ) when there is no ambiguity. We first consider the case of a single experiment, then the general case.

Single experiment:

Theorem 1. *If only one experiment is active, the absolute error is bounded by* 1 *(*$|\Delta_1^t| \leq 1$*)*

Proof. Without loss of generality, we suppose that the considered data production task starts at time 0. There are two cases:

- either $\pi > \tau$.
  Since the transfer starts at time 0 in the approximate model and $1/\pi$ in the exact model (this is the time required to produce the first block of data), the error Δ_1^t is equal to 1 for $t = 1/\pi$. Now, since the transfer rates are equal to τ in both the exact and approximate models, we have $\boldsymbol{\Delta_1^t \leq 1}, \forall t \geq 1/\pi$.
- or $\pi \leq \tau$.
  In the exact model we have a sequence of transfers at rate τ with start times equal to $\frac{i}{\pi}$ where i is a positive integer (see Figure 5). In the approximate model, however, data transfer starts at time 0 at a fixed rate equal to π. In the time interval $[0, 1/\pi]$, the value of Δ_1^t will therefore increase linearly from 0 to 1. Then, in the interval

$[1/\pi, 1/\pi+1/\tau]$, Δ_1^t decreases from 1 to $\frac{\pi}{\tau}$. Finally, in the interval $[1/\pi+1/\tau, 2/\pi]$, it increases from $\frac{\pi}{\tau}$ up to 1. We can observe that there is a cyclic pattern and thus we have a maximum error of $|\boldsymbol{\Delta_1^t}| \leq \mathbf{1}$ for all t.

□

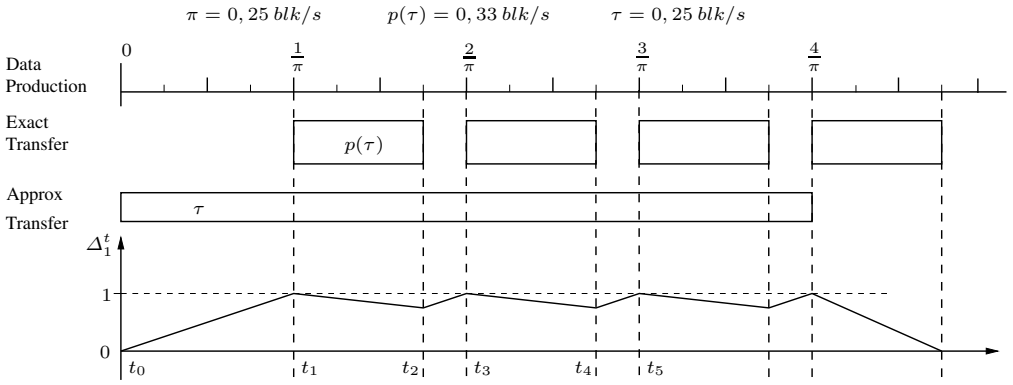

Fig. 5: Value of discrepancy of memory usage Δ_1^t when $\pi \leq p(\tau)$ (single experiment)

Observe that one could propose a slightly different model where the transfer of the data produced by an experiment E_k starts after $1/\pi_k$ seconds, that is, the time it requires to produce the first block of data. By doing this, the error in the simplest case (single experiment and $\pi_k > \tau_k$) would be reduced to 0. However, if a high producing task directly follows a smaller producer of the same experiments, the delay on the first transfer is larger than on the second transfer. This situation may entail an overlap between the two transfers, making this idea very difficult to implement for a very small theoretical gain.

General case:

When multiple experiments are active simultaneously, the analysis is more complex. Consider Figure 6. In this example, we have two experiments E_1 and E_2 such that E_1 has higher priority than E_2. Moreover, we suppose that at time $t_2 = 1/\pi_1 + 1/\tau_1$ (i.e., when the transfer of the first block produced by E_1 ends), E_2 has at least one block of data in memory. Since the transfer channel is free, E_2 will start transferring a block. However, the duration of this transfer may be longer than $1/\pi_1$. Moreover, the transfer of a single block cannot be interrupted. Therefore, the transfer of the next block of data from E_1 might be delayed beyond the expected date $2/\pi_1$. Figure 6 illustrates such a case, where the transfer of the second block produced by E_1 is delayed to $t_3 = \frac{1}{\pi_1} + \frac{1}{\tau_1} + \frac{1}{\tau_2}$.

Results:

We consider a sequence of experiments $E_1, \ldots, E_m$ ordered by decreasing priority.

Let $\alpha = \frac{\tau_1}{\tau_m}$ be the highest ratio of transfer rates for these experiments (experiments with lower priority have lower transfer rates). We analyze the error for the experiment with highest priority. Indeed, the type of disruption described above only comes from the interaction with experiments of lower priorities. Moreover, experiments of lower priorities use the gaps left by the experiment of higher priority. Therefore, accuracy on the latter implies accuracy on the former. Last, we restrict our analysis to the case where the production rate π is lower than the transfer rate τ. Indeed, otherwise the experiment fully uses the bandwidth, and the behavior described above does not happen.

Theorem 2. *If several experiments are active, the absolute error is bounded by* $1 + \alpha$ *(*$|\Delta_1^t| \leq 1 + \alpha$*)*

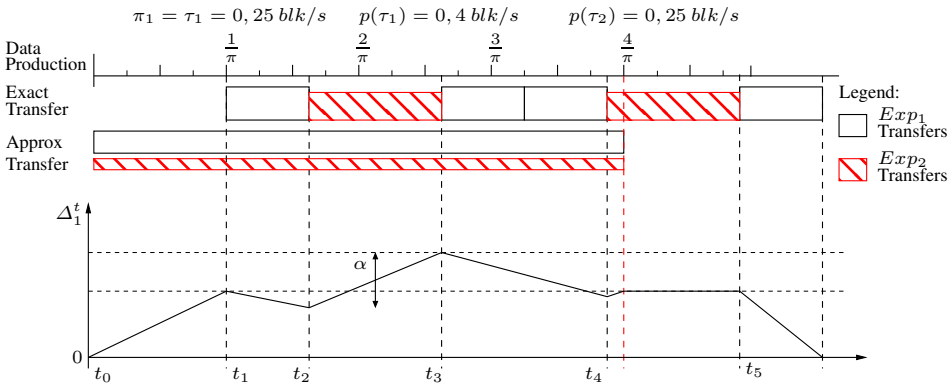

Fig. 6: Value of discrepancy of memory usage Δ_1^t when $\pi \leq p(\tau)$ (multiple experiments)

Proof. As in the previous proof, we first consider the period of transfer of the first block of E_1, starting at time $\frac{1}{\pi_1}$ and ending at time $\frac{1}{\pi_1} + \frac{1}{\tau_1}$. At time $\frac{1}{\pi_1} + \frac{1}{\tau_1}$, experiments of lower priorities might have data in their memory and hence start transferring. However, observe that as soon as the transfer of the block in process at time $\frac{2}{\pi_1}$ ends, experiment E_1 will take precedence since it has the highest priority. Therefore, the maximum delay is the size of the transfer of this block. Let E_k be the experiment that produced this data block. We can analyze the value of $\Delta_1^t = m_1^t - m_1^{t\prime}$ for any t. First, observe that since $\pi_1 < \tau_1$, data is transferred in and out of $m_1^{t\prime}$ at the same rate. Consequently, we have $m_1^{t\prime} = 0$ for all t until the start of another data production task. In other words, we bound the value of $m_1^t = \Delta_1^t$. At time $t_1 = 1/\pi_1$, we have $m_1^{t_1} = 1$ and it will then decrease to π_1/τ_1 at time $t_2 = 1/\pi_1 + 1/\tau_1$ (i.e., $m_1^{t_2} \leq 1$). Now, over the interval $[t_2, t_3] = [1/\pi_1 + 1/\tau_1, 1/\pi_1 + 1/\tau_1 + 1/\tau_k]$, experiment E_1 produces data at rate π_1 during $1/\tau_k$ seconds, and does not transfer any block out. The memory load will therefore increase by π_1/τ_k during this period. Since we have $\pi_1 < \tau_1$, we can bound this value by $\tau_1/\tau_k \leq \alpha$. In other words, $\Delta_1^{t_3} = m_1^{t_3} \leq 1 + \alpha$. Subsequently, since E_1 has the highest priority, it will transfer its data onto the mass memory at rate τ_1 until

strictly less than one block is left its memory. It follows that m_1^t will decrease linearly over an interval $[t_3, t_4]$ with $m_1^{t_4} < 1$. Now we can use a recursive argument: the delay due to the transfer of an experiment of lower priority will increase the usage of the memory of E_1 by at most α, starting from a value smaller than 1. Therefore we have $\Delta_1^t = m_1^t \leq 1 + \alpha$ for all t. □

In theory, the transfer rate should be a constant, and therefore we should have $\alpha = 1$. However, in practice, the observed transfer rates fluctuate with respect to the number of simultaneously active experiments, the relative priority of each experiment, and the presence of simultaneous transfers to the orbiter. This is due to the CDMS scanning procedure, as explained in Section 3.1. Table 1 illustrates typical observed transfer rates for a set of five experiments. The third column gives the transfer rates from experiments to the mass memory. We can see that the highest value of α will be $3.85/2.0 = 1.925$ for 5 active experiments. It follows that our model will emulate the state of the memories with an error of less than 3 blocks of data in this case.

Table 1: Variable transfer rates

Nb. Exp. x	Priority y	$\tau^{\text{observed}}_{k,x,y,0}$	$\tau^{\text{observed}}_{k,x,y,1}$
1	1	6.14	3.77
2	1	5.36	3.99
2	2	4.74	4.09
3	1	4.74	4.09
3	2	4.28	3.77
3	3	3.80	3.45
4	1	4.31	4.09
4	2	3.88	3.55
4	3	3.46	3.08
4	4	3.10	2.37
5	1	3.90	3.85
5	2	3.48	3.30
5	3	3.06	2.80
5	4	2.80	2.30
5	5	2.65	2.00

4 Checking and Filtering Algorithms

In this section, we introduce a filtering procedure for the DATATRANSFER constraint. We first present an efficient $O(nm \log(nm))$ (recall that nm is the total number of tasks) procedure for computing transfers and memory usage of a given schedule. This procedure executes 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 tasks, 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 a list of *events*. Each event is time tagged, there are six types of "static" events, i.e., that are known before executing the sweep procedure (for $O(nm)$ events in total):

- *Start/end of an experiment*;
- *Start/end of a data-producing task*;
- *Start/end of visibility*;

Moreover, "dynamic" events will be created and inserted while executing the sweep. Those events correspond to deadlines due to the memory capacity. In other words, outside the events above, the conditions might change if an experiment memory is emptied (in which case, a transfer might stop or continue at the production rate), or if the mass memory is filled or emptied. We call these *Deadline* events.

We first sort the events in chronological order. Next, in the main loop we explore the list of events 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 task 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 using the principle described in Section 3.2.

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 zero, we create a transfer. We first compute its potential transfer rate $p(\tau_{(k,t)})$ according to Expression (2) and setting t to the event time. We then compute the actual transfer rate τ_k^t using Expression (3). These computations can be done in $O(1)$ for each experiment as the cumulative bandwidth usage $\sum_{i=1}^{R(k)-1} \frac{\tau_{P(i)}^t}{\tau_{(P(i),t)}}$ is computed incrementally.

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 create a dynamic *Deadline* event and we add it to the list of events. Events of this type will do nothing on their own, however, they 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 created 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 is $O(nm)$.

4.2 Filtering Rules

In this section we 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 tasks 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 tasks of an experiment E_k, the total amount of data produced by these tasks, 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 tasks, i.e., the duration between the minimum start time and maximum end time of any task in this set.

The total amount of data produced by tasks 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 task. 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:

$$\max_{t_{ki}\in\Omega}(e_{ki}) - \min_{t_{ki}\in\Omega}(s_{ki}) \geq \frac{\sum_{t_{ki}\in\Omega}\pi_{ki}p_{ki} - M_k}{\tau} \tag{7}$$

In real scenarios, data-producing tasks of a given experiment cannot overlap, and in many cases the order is known a priori. Assuming that the tasks 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 7 depicts the application of this rule. For an experiment k, we have two tasks t_{k1}, t_{k2}, the former producing $\pi_{k1} = 5$ blocks/sec and the latter $\pi_{k2} = 4$ blocks/sec, both for 70 seconds. Therefore, $\pi_{k1} p_{k1} + \pi_{k2} p_{k2} = 630$ 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/sec, this transfer will take at least 190 seconds. We can conclude that the end of t_{k2} is at least 190 seconds after the start of t_{k1}. The grey scale gives the evolution of the memory for t_{k2} finishing exactly 190 seconds after the start of t_{k1}.

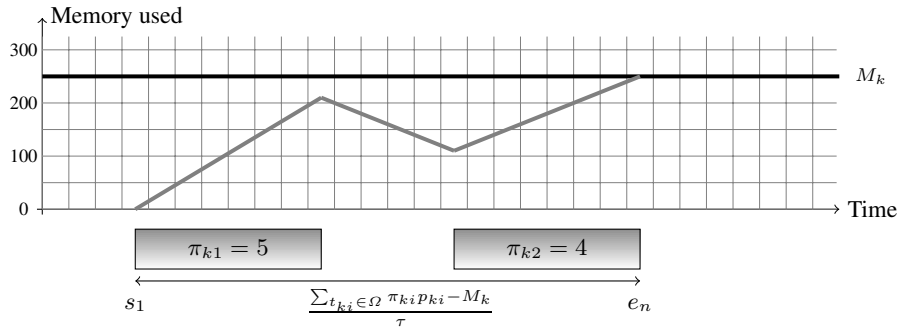

Fig. 7: Example of minimal span constraint.

Moreover, we can take into account the data produced by tasks of experiments with higher priority, since the corresponding data transfer 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 :

$$\begin{aligned} &\min\left(|t_{ki} \cap [a, b]|\right) = \\ &\min\left(|[a, b] \cap [\min(s_{ki}), \min(s_{ki}) + p_{ki}]|, |[a, b] \cap [\max(s_{ki}), \max(s_{ki}) + p_{ki}]|\right) \end{aligned}$$

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 over the interval $[a, b]$ as a lower

bound on the data produced, divided by the maximum transfer rate:

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

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 Constraint 7 as follows (with $a = \max_{t_{ki} \in \Omega}(\min(s_{ki}))$ and $b = \min_{t_{ki} \in \Omega}(\max(s_{ki}) + p_{ki})$):

$$\max_{t_{ki} \in \Omega}(e_{ki}) - \min_{t_{ki} \in \Omega}(s_{ki}) \geq \frac{\sum_{t_{ki} \in \Omega} \pi_{ki} p_{ki} - M_k}{\tau} + T_k(a,b) \tag{8}$$

We apply this rule for every set of consecutive tasks (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 $O(n^3 m^2)$.

Mass memory saturation: Since transfers from the lander to the orbiter are possible only during some visibility windows, 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 tasks which 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 $\overline{t}$, the mass-memory will necessarily be filled. It will remain so until the next visibility. Now, if a task t_{ki} produces more data in the interval between $\overline{t}$ 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 therefore deduce that the task t_{ki} must start either early enough to produce before $\overline{t}$ 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 $\overline{t}$ 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 tasks that are necessarily scheduled within the interval $[a,b]$.

Proposition 1. *Scheduling all data-producing tasks 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 task 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 tasks can run in parallel, experiments of high priority can preempt transfer intervals of experiment of lower priority. Therefore, one could advance a data-producing task 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 task t_{ji} of higher priority never helps minimizing the mass memory usage. □

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 $\overline{t}$, 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 $[\overline{t}, v]$ *and* stored without loss. There are $m_0^{\overline{t}}$ 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 tasks 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 a task t_{ki} and the interval $[\overline{t}, v]$, a task 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:

$$\sum_{i=1}^{n} \left(\min(|t_{ki} \cap [\overline{t}, v]|) * \pi_{ki}\right) \leq \delta_k$$

from which we can deduce the following implied constraint:

$$|t_{ki} \cap [\overline{t}, v]| \leq \left(\delta_k - \sum_{j \neq i \in [1,n]} (\min(|t_{kj} \cap [\overline{t}, v]|) * \pi_{kj})\right) \Big/ \pi_{ki} \tag{9}$$

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 task t_{ki}. Finally we compute the implied constraint also in time $O(n)$ (it takes constant time for each task, 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 a task 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 conducted experiments on different scenarios provided by the

group SONC of CNES. Each scenario consists in several experiments of the SDL or FSS which must be scheduled on a time window between 10 hours and 5 days for the longest scenario. 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 13 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 by MOST+ILCRESERVOIR) models data transfers using Ilog-Scheduler ILCRESERVOIR constraints. In our version (denoted by MOST+DATATRANSFER) we use only the first filtering rule described in Section 4.2.[4]

We report the results in Table 2. We present for each scenario the set of experiments involved, the memory capacities, and whether the visibility is continuous or not. Then 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 backtrack-free, whereas the previous model actually need to branch on the variables standing for transfer durations and therefore requires exploring a sizeable 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. Finally, 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 previous 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 (Consert/Romap scenarios), 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.

On the last scenario, we search a valid solution for the two experimental sequences SDL and FSS in a macroscopic model. One can observe that the time difference between the two versions is very high. In fact the longer the scenario is, the longer the search time will be.

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

Table 2: Old vs. new version of MOST on 8 standard scenarios

Scenario	Parameters			MOST+ILC-RESERVOIR			MOST+DATATRANSFER		
	M_k	M_0	Visi.	Fails	Init. time	Search time	Fails	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
APXS/Sesame	125/2000	17456	Periodic	44	27.74	28.53	0	3.5	0.19
SD2/Ptolemy/Cosac/ Mupus/Sesame	64/2000/24000/ 750/1750	17456	Periodic	1657	265.45	1639.33	0	21.14	6.71
Overview FSS (10 exps)		17456	Periodic	591	125.69	565.09	0	2.12	0.49
SDL $30min$ (8 exps)		17456	Periodic	1145	14.94	20.52	0	9.47	0.23
SDL $360min$ + FSS (15 exps) macro vision		17456	Periodic	44201	217.09	10801.40	0	4.44	1.20

The performance gain obtained by our modification are consequential, from long calculation of several minutes we can find a solution within few seconds of computing (see Table 2). This highlights the fact that the major problem in MOST concerned the modeling and management of the transfer tasks. We were also able to obtain solution that the former version of MOST could not find due to a too long computing time.

5.2 Model accuracy

In the model MOST+ILCRESERVOIR, tasks with very low data-producing rates are treated differently because of rounding issues: It is assumed that the data is produced all at once at the end of the task. 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. Scenarios Consert/Romap highlight this problem on SONC's version of MOST. Both experiments have small data-producing tasks 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 since data-transfer tasks are preallocated every 120 seconds on the time line. As a result, the model using the ILCRESERVOIR constraint has no solution, whereas transfers are actually possible.

Figure 8 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 tasks. We can see that the transfer task does not coincide with the data-producing task. Indeed there is a gap, because data-producing rate is too low to trigger a transfer task earlier.

These results show that our model is closer to the real case, because our approximate representation of data transfers gives a solution very similar to the exact one. One of the most important improvement of our version is the precision and the realism of

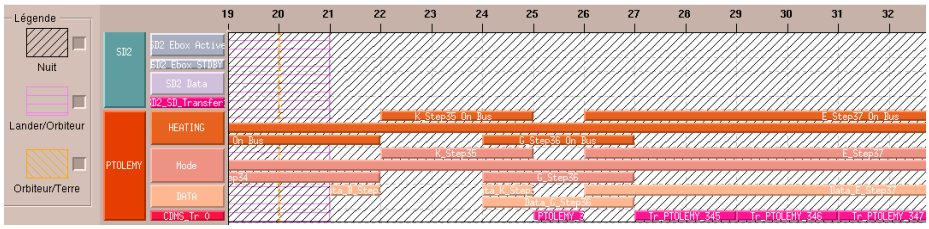

Fig. 8: Example of MOST+ILC-RESERVOIR

our model; we obtain the same behavior of the CDMS than in practice.

6 Conclusion

In this paper, we have presented an application of constraint programming for the international space mission Rosetta/Philae. We have identified that the main problem in the incumbent version of the used tool was the modeling and management of data transfers, particularly data loss. Thus, we have shown that the previous constraint programming approach was not well-adapted to the problem under consideration and we introduced a global constraint to forbid 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 considering computing times and accuracy of the produced solutions.

Acknowledgements

This work has been supported by CNES. We would like to thank members of the Science Operations and Navigation Centre (SONC) in Toulouse for their invaluable input.

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. Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-Based Scheduling*. Springer, 2001.
3. Nicolas Beldiceanu and Mats Carlsson. Sweep as a generic pruning technique applied to the non-overlapping rectangles constraint. In *Seventh International Conference on Principles and Practice of Constraint Programming (CP 2001), LNCS 2239*, pages 377–391. Springer, 2001.
4. Amedeo Cesta, Gabriella Cortellessa, Michel Denis, Alessandro Donati, Simone Fratini, Angelo Oddi, Nicola Policella, Erhard Rabenau, and Jonathan Schulster. Mexar2: AI solves mission planner problems. *IEEE Intelligent Systems*, 22(4):12–19, 2007.

5. Philippe Laborie. Algorithms for propagating resource constraints in AI planning and scheduling: existing approaches and new results. *Artificial Intelligence*, 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), Doctoral Consortium*, 2003.
7. Angelo Oddi and Nicola Policella. Improving robustness of spacecraft downlink schedules. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 37(5):887–896, 2007.
8. Giovanni Righini and Emanuele Tresoldi. A mathematical programming solution to the Mars Express memory dumping problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 40(3):268–277, 2010.