Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)

Building Resource-Dependent Conditional Plans for an Earth Monitoring Satellite

Cédric Pralet,[1] David Doose,[1] Julien Anxionnat[2], Jérémie Pouly[2]

[1] ONERA, Université de Toulouse, 2 av. Edouard Belin, BP 74025 F-31055 Toulouse Cedex 4, France
[2] CNES, 18 av. Edouard Belin, 31401 Toulouse Cedex 9, France
{cedric.pralet,david.doose}@onera.fr, {jeremie.pouly,julien.anxionnat}@cnes.fr

Abstract

This paper introduces a conditional planning and execution system for an Earth monitoring satellite. This system builds plans containing optional acquisitions that are activated or not at execution time depending on the amount of energy actually available. One originality is that the energy activation thresholds computed on the ground take into account the capacity of the satellite to use an heliocentric pointing or switch off the payload when acquisitions are canceled. Another originality is that the conditional planner proposed uses several energy propagation models, from conservative models containing margins on power production to optimistic models that allow opportunistic acquisitions to be planned.

Introduction

In this paper, we introduce a planning and execution system for a low Earth orbit nanosatellite developed by the French Space Agency (CNES) and that should be launched in 2022-2023. The mission of this nanosatellite is to perform acquisitions over areas at the Earth surface and download collected data towards ground reception stations. During acquisitions, the instrument must be pointed to the center of the Earth (geocentric pointing), while during data downlink, the emission antenna must be pointed to a ground station. The satellite can also use an heliocentric pointing (solar panels orthogonal to the Solar rays) to better recharge its batteries. In this context, based on technical documentations of the nanosatellite, our goal is first to define a planning strategy (the content of this paper) and then to implement the concept proposed on the real system (the next step for us).

Basically, the mission planning system is responsible for building acquisition and download plans covering as many ground areas as possible. It must take into account features like the duration of maneuvers between different pointings, the duration required to switch the payload on or off, or the limited amount of energy available on-board, this last aspect being the main bottleneck in terms of resources. As usual in the space domain, the baseline approach is to build a *unique safe plan* on the ground by considering conservative margins, to cope with the uncertainty about parameters that influence for instance the total amount of energy available on-board (satellite temperature, actual orientation of the solar panels, actual power consumed by the payload, etc.). However, in the first simulations we performed, the nanosatellite that we have can make acquisitions only 12.4% of the time based on conservative models, whereas when using average or optimistic margins, it is able to perform acquisitions 17.6% and 20% of the time respectively. As a result, our objective is to go beyond the conservative approach, and for this we propose to build conditional plans (Peot and Smith 1992; Pryor and Collins 1996). More precisely, we propose to produce plans containing optional acquisitions that are then triggered or not depending on the actual level of energy available at execution time.

Using plans conditioned by resource thresholds is not new in the space domain. For instance, in an experiment for the DLR BIROS satellite (Lenzen et al. 2014), the mission center computes plans defined as sequences of plan fragments where each fragment has an energy activation threshold. In other studies on Earth observing satellites, the ground system computes an energy activation threshold for each observation with the objective to guarantee that all high-priority observations are always fulfilled (Maillard et al. 2015). Flexible download plans were also considered in these studies to manage uncertain memory consumptions (Maillard et al. 2016), but memory is not a bottleneck in our case. Last, in works on the NASA Mars 2020 rovers, each activity planned offers a set of ranked alternatives called *switch groups*, with for each alternative an energy activation threshold computed on the ground (Agrawal et al. 2021). In all these experiments, conditional plans are compatible with the low computational resources available onboard, and they allow the mission center to keep control of the decisions made.

With regards to these previous works, the approach proposed in this paper brings several contributions:

- first, when acquisitions are canceled, we can change the pointing of the satellite to improve power production;
- second, this opportunistic pointing adaptation strategy is exploited on the ground to compute more permissive activation thresholds for the acquisitions;
- third, a least-commitment approach allows the conditional decisions to be postponed as much as possible while managing setup operations like maneuvers;
- fourth, conditional plans are obtained from several energy models, including pessimistic and optimistic ones.

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The paper is organized as follows. We first describe the features of the nanosatellite considered. We then define the algorithm used on the ground to build conditional plans and the onboard procedure that executes these plans. Last, experimental results and validation issues are discussed, and some perspectives are provided. The long-term idea is that the real nanosatellite demonstration will first use a baseline conservative non-conditional planner and then switch to the conditional planning approach defined in this paper.

System Specifications

System states Fig. 1 gives a standard activity plan for the satellite. The latter can be seen as a disjunctive resource that performs acquisitions and data transfers, together with setup operations like maneuvers (in magenta). In between these activities, the satellite uses one of the *waiting modes* in set $\mathcal{W} = \{ON, OFF, SBY\}$. In the ON mode, the satellite maintains a geocentric pointing and the payload is on. In the OFF mode, the satellite maintains a geocentric pointing and the payload is off. In the SBY mode, the satellite maintains an heliocentric pointing and the payload is off.

Fig. 2 details the possible transitions between these waiting modes for the acquisition process, that also involves an ACQ mode standing for a state where the satellite maintains a geocentric pointing, the payload is on, and the instrument collects data. Each transition corresponds to setup operations that take some time. For instance, transition $[SBY \rightarrow ON]$ requires to perform a maneuver (up to several minutes) and to switch on the payload (a few tens of seconds). Transition $[ON \rightarrow ACQ]$ requires to load the acquisition to be performed (a few seconds). Transition $[ON \rightarrow OFF]$ requires to switch off the payload (a few seconds). The duration of these setup operations is constant, except for maneuvers whose duration is time-dependent (e.g., the maneuver duration between a geocentric pointing and an heliocentric pointing depends on the time at which the maneuver is triggered). *In the following, we assume that the duration of a maneuver is always greater than the duration required to switch the payload on or off.*

Plan An acquisition plan $\pi = [a_1, \ldots, a_n]$ is a sequence of successive acquisitions a_i that have a fixed start time $Start(a_i)$, a fixed duration $Du(a_i)$, and a fixed end time $End(a_i) = Start(a_i) + Du(a_i)$. For an acquisition a that does not overlap any acquisition in π, we denote by $\pi^+(a)$ the sequence of acquisitions placed after a in π.

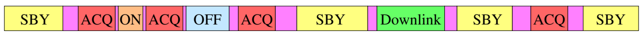


Figure 1: Satellite activity plan

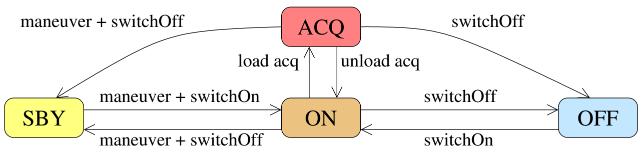


Figure 2: Satellite running modes for the acquisition process

Waiting modes between acquisitions One key aspect of the system considered is that the waiting mode $W(a, b) \in \mathcal{W}$ to use between two successive acquisitions a and b is completely determined by predefined rules depending on the duration $\delta = Start(b) - End(a)$ available between the end of a and the start of b. Let us detail these predefined rules. Let m_1 be the minimum duration of a maneuver from a geocentric pointing at time $End(a)$ to an heliocentric pointing. Let m_2 be the minimum duration of a maneuver starting from an heliocentric pointing and reaching a geocentric pointing at time $Start(b)$. Let m_3 be the minimum duration required by the chain of transitions $[ACQ \rightarrow OFF \rightarrow ON \rightarrow ACQ]$. Then, the rules are:

$$\begin{array}{l} \text{IF } \delta \geq m_1 + m_2 \text{ THEN } W(a,b) = SBY \\ \text{ELSEIF } \delta \geq m_3 \text{ THEN } W(a,b) = OFF \\ \text{ELSE } W(a,b) = ON \end{array} \quad (1)$$

We assume here that for two successive acquisitions a and b, we always have $Start(b) - End(a) \geq m_4$ where m_4 stands for the minimum duration of the chain of transitions $[ACQ \rightarrow ON \rightarrow ACQ]$, so that using waiting mode ON between a and b is always feasible from a temporal point of view. The objectives of the previous rules are to use an heliocentric pointing as much as possible to maximize power production, and to switch off the payload as much as possible to minimize power consumption. Minimum OFF or SBY durations could also be easily added in the rules.

Decision dates The acquisition plans considered contain optional acquisitions that can be activated or not. The decision to trigger an acquisition a must be made at a time that allows to perform all relevant setup operations before a. Depending on the current waiting mode, there are actually three possible decision times referred to as $\tau_{SBY}(a)$, $\tau_{OFF}(a)$, $\tau_{ON}(a)$ (see Fig. 3), where $\tau_w(a)$ stands for the latest date at which mode w can be left if w is the current waiting mode and the next goal is to perform a.

Downlink activities The communication windows with ground stations are imposed and we assume that the maximum power required to communicate is consumed over these windows. This ensures that the mission center can always communicate with the satellite when needed.

Energy limitations The nanosatellite has a limited memory capacity but the main bottleneck of the system is the capacity of the batteries. The amount of energy available cannot be greater than a maximum value $Emax$ corresponding to a maximum state of charge, and it must never be less than a minimum value $Emin$. To check this constraint, we dispose of a model for computing the energy evolution profile induced by a given plan π. More formally, an energy model M is defined by a function Φ_M that takes as inputs:

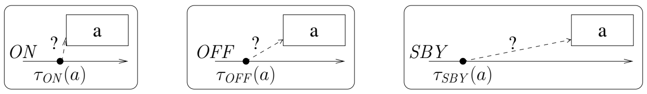


Figure 3: Latest dates at which the activation of an acquisition can be decided depending on the current waiting mode

- an initial time $t_0 \in \mathbb{R}$, an initial mode $w_0 \in \mathcal{W}$, and an initial energy level $e_0 \in [Emin, Emax]$;
- a final time t_1 and a final waiting mode $w_1 \in \mathcal{W}$;
- a plan π to execute over $[t_0, t_1]$, with the assumption that, from a temporal point of view, this plan is feasible over $[t_0, t_1]$ with the given initial and final modes;
- a time $t \in [t_0, t_1]$.

Then, $\Phi_M(t_0, w_0, e_0, t_1, w_1, \pi, t) \in [0..Emax]$ gives the amount of energy available at time t when performing plan π and using the specified initial and final conditions. Function Φ_M takes into account all switch on/off operations and maneuvers induced by π, the power produced through the solar panels depending on the satellite pointing induced by π, the power consumed by the activities of π, the power consumed during all communication windows with ground stations, the energy saturation at value $Emax$, and the temperature onboard the satellite. In its detailed version, function Φ_M is not linear in the duration of the activities of the plan due to non linear effects on the state of charge of the batteries.

From this, given a minimum level of energy e_1 required at the end, plan π is said to be *valid* over $[t_0, t_1]$ if and only if (1) the energy evolution profile has no value less than $Emin$, *i.e.* $\Phi_M(t_0, w_0, e_0, t_1, w_1, \pi, t) \geq Emin$ holds for every $t \in [t_0, t_1]$, and (2) $\Phi_M(t_0, w_0, e_0, t_1, w_1, \pi, t_1) \geq e_1$. In this case, we write $Valid_M(t_0, w_0, e_0, t_1, w_1, e_1, \pi) = true$.

On the opposite, to define conditional plans, it is useful to compute the minimum initial level of energy e_0 from which plan π is valid over $[t_0, t_1]$, given initial and final modes w_0, w_1, and given a final energy level e_1 requested at t_1. Formally, this minimum initial energy level is obtained by a function Θ_M defined as:

$$\begin{aligned}\Theta_M(t_0, w_0, t_1, w_1, e_1, \pi) = \min\{e_0 \in [Emin, Emax] \mid \\ Valid_M(t_0, w_0, e_0, t_1, w_1, e_1, \pi) = true\}\end{aligned} \quad (2)$$

Eq. 2 is computed using an iterative method that considers a new candidate value for $\Theta_M(t_0, w_0, t_1, w_1, e_1, \pi)$ at each step. This value can be obtained by dichotomy or by linear interpolation techniques measuring the (positive or negative) distance between the minimum level of energy requested and the estimated energy evolution profile. In both cases (dichotomy or interpolation), the idea is to converge to $\Theta_M(t_0, w_0, t_1, w_1, e_1, \pi)$, up to a given precision, by successive forward propagations through function Φ_M.

Conditional Planning on the Ground

From the previous definitions, we can introduce the algorithm proposed to build conditional plans on the ground.

Initial plan We consider as an input the acquisition plan π_0 produced by the baseline greedy planner currently designed for the mission. This planner starts from an empty plan and tries to add one more acquisition at each step, using a specific ranking policy for choosing the next candidate acquisition to insert. For each acquisition insertion attempt, the algorithm updates the download plan and checks memory-related constraints. To check the validity of the energy evolution profile, it builds a full plan (containing all waiting modes, downlink activities, maneuvers, etc.), and then uses a *pessimistic energy model* M_0 involving margins on power production and consumption. If any constraint is violated, the acquisition is rejected. The process continues until there is no more candidate acquisition.

Overview of the algorithm proposed To try and get more efficient plans, we go beyond the single pessimistic energy model M_0 and consider several energy models $M_1, \ldots M_K$ that are more and more permissive, meaning that an acquisition plan valid for model M_k is also valid for model M_{k+1}. For instance, we can consider an *average* energy model M_1 and an *optimistic* energy model M_2.

The main idea is then to start from initial plan π_0 and perform a sequence of planning phases to cover more and more acquisitions. Fig. 4 illustrates this strategy. Fig. 4a gives a plan $\pi_0 = [A, B, C]$ produced by the baseline mission planner, based on model M_0. According to M_0, the energy evolution profile reaches value $Emin$ at some point. But for energy model M_1, plan π_0 is too conservative since there is a gap between the energy profile estimated by M_1 and value $Emin$. As shown in Fig. 4b, a second planning phase based on model M_1 leads to a new plan $\pi_1 = [A, D, B, E, F, G, C]$ containing four new (optional) acquisitions. If M_1 corresponds to an average model, we have a good chance to perform many acquisitions in π_1 since periods where the real power consumed is higher than in M_1 are likely to be compensated by periods where it is lower than in M_1. For each optional acquisition a in π_1, we also compute an energy threshold to determine whether a can be triggered while guaranteeing that all *remaining* acquisitions in π_0 can still be performed. On Fig. 4b again, we see that optimistic energy model M_2 estimates that there is still some energy left. A third planning phase based on M_2 can lead to plan $\pi_2 = [H, A, D, B, E, I, F, G, J, C]$ shown in Fig. 4c. For π_2, model M_2 alone may estimate that there are still some acquisition opportunities left, but the algorithm will forbid the addition of optional acquisitions that jeopardize some remaining acquisitions planned at the previous levels.

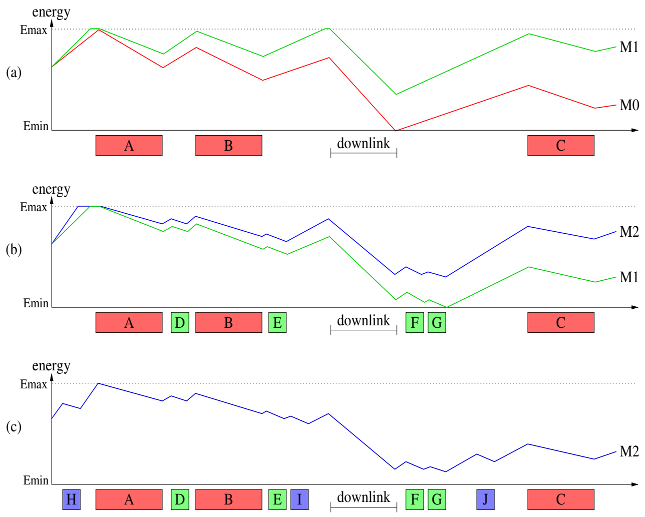


Figure 4: Addition of acquisitions for energy models M_1 and M_2, starting from plan $\pi_0 = [A, B, C]$ obtained with M_0

Conditional planner: main function The *plan* function given in Algo. 1 is the main procedure of the approach proposed. It builds a plan over time frame $[t_0, t_1]$ given initial and final modes w_0 and w_1 at times t_0 and t_1, and a minimum level of energy e_1 required at t_1. It first computes an initial plan π_0 using the baseline planner (Line 2). This plan contains the so-called *mandatory acquisitions*. Then, successive planning phases are performed to add *optional acquisitions*, based on energy models M_1 to M_K (Lines 4-8). The kth planning phase leads to a plan π_k, and each value $k \in [0..K]$ is called a *planning level*. All acquisitions a added from π_{k-1} to π_k have planning level $\lambda(a) = k$, and for each of them the algorithm computes so-called *left energy thresholds* defining the amount of energy from which the acquisition can be triggered (Line 8, more details later on this point). For every acquisition a involved in final plan π_K, the algorithm also computes so-called *right energy thresholds* defining the waiting mode to use at the end of each acquisition (Line 10, more details later on this point).

Planning at each level In Algo. 1, function *nextPlan* specifies how a new plan is obtained at each planning level k. It first computes the set Σ of acquisitions that are not already covered by acquisitions in π_{k-1} (Line 13). After that, while there is a candidate acquisition left, an acquisition $a \in \Sigma$ is selected based on the same ranking function as in the baseline planner (Line 16). It is added to the current plan and the system constraints are checked (Line 17). If the check succeeds, set Σ is updated to remove acquisitions that cover the same ground area as a. Otherwise, a is rejected.

Constraint checks In Algo. 1, the *check* function first verifies that the memory capacity is not exceeded given the current acquisition plan π and the set of downlink windows (Line 22). It then checks that π is valid with regards to energy model M_k (Line 23). For each model M_k, functions Φ_{M_k} and Θ_{M_k} are denoted here by Φ_k and Θ_k, and $e_{0,k}$ stands for the initial energy level considered in M_k.

After that, function *check* verifies that the addition of acquisition a is accepted by energy models $M_0, \dots, M_{k-1}$. In particular, one objective here is to forbid the addition of acquisitions that are incompatible with the *remaining* mandatory acquisitions in π_0. To do this, we consider the scenario where the waiting mode is equal to *ON* just before a. In this case, the time at which the triggering decision for a must be made is $tDec = \tau_{ON}(a)$ (Line 24), and model M_k provides an estimation of the level of energy *eDec* available at that time (Line 25). If starting from configuration $(tDec, ON, eDec)$, there exists a planning level $k' < k$ such that model $M_{k'}$ forbids to perform a followed by $\pi^+_{k'}(a)$, then acquisition a is rejected. Doing so, we intentionally combine on one hand the kth energy model on a plan prefix to get configuration $(tDec, ON, eDec)$, and on the other hand the k'th energy model on the suffix to validate the feasibility of plan $[a] \cdot \pi^+_{k'}(a)$. With this process, an acquisition a can first be rejected at planning level 0, where energy checks are performed by considering *all* mandatory acquisitions, and then accepted as an optional acquisition at level $k > 0$ when state $(tDec, ON, eDec)$ allows to execute a and all mandatory acquisitions *following* a.

Algorithm 1: Conditional planning on the ground

```
1  Function plan($t_0, w_0, t_1, w_1, e_1$)
2      $\pi_0 \leftarrow initPlan(t_0, w_0, t_1, w_1, e_1)$
3      for $a \in \pi_0$ do $\lambda(a) \leftarrow 0$
4      for $k = 1$ to $K$ do
5          $\pi_k \leftarrow nextPlan(t_0, w_0, t_1, w_1, e_1, [\pi_0 ... \pi_{k-1}])$
6          for $a \in \pi_k \setminus \pi_{k-1}$ do
7              $\lambda(a) \leftarrow k$
8              $setLeftThresholds(a, t_1, w_1, e_1, [\pi_0 ... \pi_{k-1}])$
9      for $a \in \pi_K$ do
10         $setRightThresholds(a, t_1, w_1, e_1, [\pi_0 ... \pi_K])$
11     return $\pi_K$

12 Function nextPlan($t_0, w_0, t_1, w_1, e_1, [\pi_0 ... \pi_{k-1}]$)
13     $\Sigma \leftarrow getCandidateAcquisitions(\pi_{k-1})$
14     $\pi \leftarrow \pi_{k-1}$
15     while ($\Sigma \neq \emptyset$) do
16         pick $a$ from $\Sigma$ and add $a$ to $\pi$
17         if $check(t_0, w_0, t_1, w_1, e_1, \pi, a, [\pi_0 ... \pi_{k-1}])$ then
18             $\Sigma \leftarrow updateCandidateAcquisitions(\Sigma, a)$
19         else remove $a$ from $\pi$
20     return $\pi$

21 Function check($t_0, w_0, t_1, w_1, e_1, \pi, a, [\pi_0 ... \pi_{k-1}]$)
22     if $\neg checkMemory(\pi)$ then return false
23     if $\neg Valid_k(t_0, w_0, e_{0,k}, t_1, w_1, e_1, \pi)$ then return false
24     $tDec \leftarrow \tau_{ON}(a)$
25     $eDec \leftarrow \Phi_k(t_0, w_0, e_{0,k}, t_1, w_1, \pi, tDec)$
26     for $k' = 0$ to $k - 1$ do
27         if $\neg Valid_{k'}(tDec, ON, eDec, t_1, w_1, e_1, [a] \cdot \pi^+_{k'}(a))$
               then return false
28     return true
```

Left thresholds As mentioned before, *left thresholds* determine whether an optional acquisition a can be triggered depending on the actual level of energy observed during execution. As the uncertain execution of conditional plans implies that an optional acquisition a might be triggered from different waiting modes, the algorithm computes a set of thresholds $\{\theta^L_{SBY}(a), \theta^L_{OFF}(a), \theta^L_{ON}(a)\}$ where $\theta^L_w(a)$ stands for the minimum amount of energy required to trigger a from waiting mode w at time $\tau_w(a)$. These left thresholds are computed by function *setLeftThresholds* given in Algo. 2. Basically, for each acquisition a planned at level k and each planning level $k' < k$, model $M_{k'}$ allows to perform a from waiting mode w only if it estimates that the amount of energy available at time $\tau_w(a)$ suffices to perform a and all following acquisitions in $\pi_{k'}$, *i.e.* if this amount is not less than $e_{k'} = \Theta_{k'}(\tau_w(a), w, t_1, w_1, e_1, [a] \cdot \pi^+_{k'}(a))$. Threshold $\theta^L_w(a)$ then simply corresponds to the maximum of these $e_{k'}$ quantities over all $k' < k$ (Line 3). In particular, as M_0 is pessimistic, threshold $\theta^L_w(a)$ ensures that all mandatory acquisitions following a are always feasible if a is activated, while the execution of acquisitions planned at levels $k' \in [1..k-1]$ are guaranteed only relatively to $M_{k'}$.

Right thresholds As mentioned before, *right thresholds* determine the waiting mode to choose at the end of an acquisition a at execution time. Such a decision is illustrated in

Algorithm 2: Computation of left and right thresholds

```
1  Function setLeftThresholds($(a, t_1, w_1, e_1, [\pi_0 ... \pi_{k-1}])$)
2      for $w \in \{ON, OFF, SBY\}$ do
3          $\theta^L_w(a) \leftarrow \max_{k' \in [0..k-1]} \Theta_{k'}(\tau_w(a), w, t_1, w_1, e_1, [a] \cdot \pi^+_{k'}(a))$

4  Function setRightThresholds($(a, t_1, w_1, e_1, [\pi_0 ... \pi_K])$)
5      $w_0 \leftarrow W(a, next(a, \pi_0))$
6      for $(w, w_0) \in \{(ON,SBY),(OFF,SBY),(ON,OFF)\}$ do
7          if $B \neq \emptyset$ where $B = \{b \in \pi^+_K(a) \mid W(a,b) = w\}$ then
8              $k \leftarrow \min_{b \in B} \lambda(b)$
9              $b^*_w \leftarrow argmin_{b \in B \mid \lambda(b)=k}\, Start(b)$
10             $\theta^R_w(a) \leftarrow$
                   $\max_{k' \in [0..k-1]} \Theta_{k'}(End(a), ON, t_1, w_1, e_1, [b^*_w] \cdot \pi^+_{k'}(b^*_w))$
```

Fig. 5, where each box represents an acquisition, and where acquisitions b in block w are those for which $W(a, b) = w$. At the end of a, the simplest approach would be to get the next mandatory acquisition $b_0 = next(a, \pi_0)$ following a in π_0 and systematically use waiting mode $w_0 = W(a, b_0)$. This strategy however under-uses the satellite since it can prevent optional acquisitions placed between a and b_0 from being performed, due to non-preemptive setup operations. This is why, to choose a waiting mode at the end of a, the algorithm computes two right thresholds $\theta^R_{ON}(a)$ and $\theta^R_{OFF}(a)$; for each waiting mode $w \in \{ON, OFF\}$, $\theta^R_w(a)$ stands for the minimum amount of energy from which the satellite is allowed to use mode w at the end of a. As using mode w_0 should always be allowed, it can be shown that defining a third right threshold $\theta^R_{SBY}(a)$ is not mandatory.

The computation of the right thresholds is detailed in function *setRightThresholds* of Algo. 2. This function first determines the next mandatory acquisition b_0 that follows a, and the waiting mode w_0 associated with a direct transition from a to b_0 according to the rules defined in Eq. 1 (Line 5). For waiting modes w placed "before" w_0 (Line 6), the algorithm determines the set of acquisitions b such that $W(a, b) = w$ and the minimum planning level k associated with such acquisitions (Line 8). Among the candidate acquisitions for b, it then considers the earliest one, referred to as b^*_w (Line 9). Intuitively, b^*_w is seen as the best acquisition that justifies using waiting mode w after a instead of default waiting mode w_0. For acquisition b^*_w, the algorithm then computes the minimum amount of energy required, at the end of a, to perform b^*_w and all acquisitions planned before level k, according to models M_0 to M_{k-1} (Line 10). This minimum amount corresponds to the threshold required to use mode w after a. As shown later, if the waiting mode w chosen differs from the mode w_0 recommended by π_0, the onboard plan adaptation process will impose to perform acquisition b^*_w if no other acquisition is made between a and

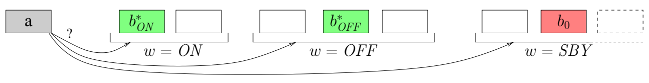


Figure 5: Choice of a waiting mode after an acquisition

b^*_w, so that any deviation from π_0 (and the associated energy recharge loss) is rewarded by at least one more acquisition.

Conditional plans and automatic mode adaptation In the end, as shown in Fig. 6, there are both left and right thresholds for each acquisition a of the plan. These thresholds determine if and how a given acquisition can be inserted in the middle of the waiting modes of the satellite. It is important to note that we never commit waiting modes on the ground, since the precise sequence of acquisitions executed is not known in advance due to the uncertainty about energy. As a result, we get expressive conditional plans from a number of thresholds that is only linear in the number of acquisitions, and as shown later the maneuvers and waiting modes are directly reconstructed onboard.

Incremental computations The algorithm proposed uses numerous calls to functions Θ_k, that themselves call functions Φ_k for computing energy evolution profiles over potentially long time frames. On this point, fast incremental computation techniques can be defined. Indeed, let us consider an acquisition a added at planning level k. For every planning level $k' < k$ and every acquisition $b \in \pi_{k'}$, it is possible to compute, through a backward recursion process, quantities $\theta^L_w(b, k')$ representing the minimum energy level required, according to energy model $M_{k'}$, to execute b from mode w and then all acquisitions following b in $\pi_{k'}$. From these quantities, it can be shown that the left threshold $\theta^L_w(a)$ required at time $t = \tau_w(a)$ for acquisition a is given by:

$$\max_{k' \in [0..k-1]} \Theta_{k'}(t, w, \tau_{w_{k'}}(b_{k'}), w_{k'}, \theta^L_{w_{k'}}(b_{k'}, k'), [a]) \quad (3)$$

where $b_{k'}$ denotes the first acquisition in $\pi^+_{k'}(a)$ and $w_{k'}$ denotes waiting mode $W(a, b_{k'})$. Eq. 3 is easier to compute than the formula at Line 3 in Algo. 2 since it uses $\Theta_{k'}$ over plans containing a single acquisition. The computation of the right thresholds at Line 10 in Algo. 2 and the validity checks at Line 27 in Algo. 1 can be handled in a similar way.

Execution of Conditional Plans

We now formalize the second piece of the puzzle, that is the onboard process that is in charge of reading the conditional plans received from the ground and emitting the appropriate low-level telecommands depending on the amount of available energy observed. This onboard process is neither a complex deliberation algorithm nor the basic execution layer of the satellite, therefore we call it the *Conditional Plan Interpreter* (CPI). The detailed description is focused on the acquisition process and omits the download process. The latter is taken into account in the full implementation of

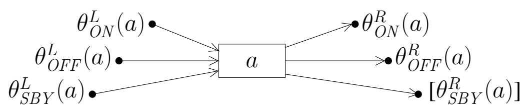


Figure 6: Left and right thresholds for an acquisition a (right threshold $\theta^R_{SBY}(a)$ can be added for the sake of fault detection, but we do not detail this point here)

the CPI by simply canceling data downlink for acquisitions that have been canceled (and without changing the pointing during communication windows). We also assume that two conditional plans P1, P2 sent by the ground mission center for two successive planning periods are independent from each other and that the satellite always waits in mode SBY at the end of a planning period.

Acquisition telecommands Regularly, the CPI receives a sequence of high-level acquisition telecommands $acqSeq = [acq_1, \ldots, acq_N]$ ordered by increasing start times. Each telecommand acq in this sequence has several parameters:

- **acq.id**: unique identifier of the acquisition;
- **acq.start**, **acq.end**: acquisition start and end times;
- **acq.params**: settings for the acquisition instrument;
- **acq.level**: level at which the acquisition has been planned (quantity $\lambda(a)$ introduced previously);
- **acq.duSbyToStart**: duration of a maneuver from the heliocentric pointing to the geocentric pointing if the goal is to reach this second pointing at time $acq.start$; this duration is provided by the ground since the onboard software has no module estimating maneuver durations;
- **acq.duEndToSby**: duration of a maneuver from the geocentric pointing to the heliocentric pointing if this maneuver starts at time $acq.end$;
- **acq.socMinLeft[w]**, for $w \in \{ON, OFF, SBY\}$: minimum level of energy (*state of charge*) at time $\tau_w(acq)$ to trigger the acquisition and the required setup operations (corresponds to threshold $\theta^L_w(a)$);
- **acq.socMinRight[w]**, for $w \in \{ON, OFF\}$: minimum level of energy required to use waiting mode w at the end of the acquisition (corresponds to threshold $\theta^R_w(a)$).

Several other attributes can be derived from the basic inputs, to make the definition of the CPI easier:

- **acq.idx**: index of the acquisition in sequence $acqSeq$;
- **acq.triggerTime[w]**, for $w \in \{ON, OFF, SBY\}$: latest time at which waiting mode w can be left to perform the acquisition (quantity $\tau_w(a)$ seen previously); this time is given by $acq.start - duSetup(acq, w)$ where:
  $duSetup(acq, ON) = DuLoadAcq$
  $duSetup(acq, OFF) = DuOffOn + DuLoadAcq$
  $duSetup(acq, SBY) = acq.duSbyToStart$
- **acq.defaultWaitingMode**: waiting mode used at the end of the acquisition for the strategy that only performs the mandatory acquisitions (value ON, OFF, or SBY);
- **acq.nextAcqIdx[w]**, for $w \in \{ON, OFF, SBY\}$; index of the next reachable acquisition telecommand when choosing waiting mode w at the acquisition end (value $length(acqSeq)$ if there is no such next telecommand);
- **acq.nextBestAcqIdx[w]**, for $w \in \{ON, OFF\}$: index of the best acquisition telecommand justifying to use waiting mode w at the end of the acquisition (value $length(acqSeq)$ if there is no such acquisition); to be consistent with the ground planner, if several acquisitions can justify the use of mode w, this index is given by the acquisition whose planning level is the lowest and in case of ties by the acquisition whose start time is the lowest.

Execution state The CPI executes the conditional plan over a rolling horizon and maintains a state S that gives a view of the progression of the execution. This state is composed of several attributes listed below.

- **S.nextDecType**: type of the next decision, with value $START$ for an acquisition triggering choice and END for a waiting mode choice at the end of an acquisition;
- **S.nextDecTime**: next time at which a decision must be made (value $MaxTime$ if there is no such next decision);
- **S.waiting**: waiting mode at the next $START$ decision;
- **S.nextAcqIdx**: index of the next acquisition that must be considered (index in acquisition sequence $acqSeq$);
- **S.committedAcqIdx**: index of the next acquisition that should be performed if there has been a deviation from plan π_0 at the end of an acquisition (this attribute records the index of the acquisition justifying the current waiting mode when the latter is not the default one);
- **S.soc**: amount of energy available (*state of charge*).

Main functions of the CPI Function $decisionLoop$ given in Algo. 3 corresponds to the main procedure of the CPI. It first initializes the state of the CPI by setting that the current waiting mode is SBY, no acquisition is committed (no deviation from the mandatory plan so far), the next acquisition to consider has index 0, and the satellite waits for the next decision time (Lines 2-5). The last point is managed by function *waitNextAcq* that simply waits for the next acquisition $START$ decision given the current waiting mode.

In the main decision loop, the CPI uses a global parameter Δ, referred to as its *latency*, that corresponds to an upper bound on the (small) duration it requires to make decisions over the next decision window. More precisely, if the CPI is called at a time t, its decision process is performed during time frame $[t, t+\Delta[$ and it must decide on all telecommands to execute over window $[t+\Delta, t+2\cdot\Delta[$. While there is some decision left in the current conditional plan (Line 6), the CPI waits for the next decision time minus Δ and calls function *decisionStep* to make decisions at that time (Lines 7-8).

Function *decisionStep* first estimates the state of charge of the batteries at time $S.nextDecTime$ (Line 16). To do this, it reads the state of charge at the current time, that should be equal to $S.nextDecTime - \Delta$ at this point, and applies a maximum discharge rate (parameter $\left(\frac{dSoC}{dt}\right)_{max}$) for duration Δ to get a pessimistic estimated state of charge at time $S.nextDecTime$. After that, while the next decision occurs before the end time of the current decision window, it is handled through function $manageAcqStart$ or $manageAcqEnd$ depending on whether it corresponds to an acquisition triggering choice at an anticipated triggering time or to a waiting mode choice at the end of an acquisition (Lines 20-22). At each decision step, the estimated state of charge is updated using the maximum discharge rate again (Line 23).

Acquisition triggering decisions Acquisition triggering is handled by function *manageAcqStart* given in Algo. 4. At this point, the conditional plan contains two possible branches: trigger the acquisition or not. The choice is made by calling function *checkStartCondition*. The latter specifies that the next acquisition acq must be performed in three

Algorithm 3: Main procedures of the CPI

```
Function decisionLoop($S$, $acqSeq$)
    $S.waiting \leftarrow SBY$
    $S.committedAcqIdx \leftarrow -1$
    $S.nextAcqIdx \leftarrow 0$
    $waitNextAcq(S, acqSeq)$
    while ($S.nextDecTime < MaxTime$) do
        $wait(S.nextDecTime - \Delta)$
        $decisionStep(S, acqSeq)$

Function waitNextAcq($S$, $acqSeq$)
    if $S.nextAcqIdx < length(acqSeq)$ then
        $acq \leftarrow acqSeq[S.nextAcqIdx]$
        $S.nextDecTime \leftarrow acq.triggerTime[S.waiting]$
        $S.nextDecType \leftarrow START$
    else $S.nextDecTime \leftarrow MaxTime$

Function decisionStep($S$, $acqSeq$)
    $S.soc \leftarrow readSoC() - \left(\frac{dSoC}{dt}\right)_{max} \cdot \Delta$
    $EndTime \leftarrow S.nextDecTime + \Delta$
    while ($S.nextDecTime < EndTime$) do
        $t \leftarrow S.nextDecTime$
        if ($S.nextDecType = START$) then
            $manageAcqStart(S, acqSeq)$
        else $manageAcqEnd(S, acqSeq)$
        $S.soc \leftarrow S.soc - \left(\frac{dSoC}{dt}\right)_{max} \cdot (S.nextDecTime - t)$
```

cases: (1) the planning level of acq is equal to 0, or equivalently acq is mandatory; (2) acq corresponds to the acquisition that justified a deviation from the mandatory acquisition plan; (3) the lower estimation of the state of charge is greater than or equal to the left threshold computed on the ground for acq in the current waiting mode.

If the "trigger" branch is activated, then all telecommands required for acq are built, including the ones useful for the setup operations from the current waiting mode (Lines 4-8). Also, no more acquisition is committed and the CPI waits for the acquisition end (Lines 9-11). Otherwise, if acquisition acq is canceled, the CPI moves on to the next acquisition and waits for the corresponding decision time (Lines 13-14).

Waiting mode choice at the end of acquisitions Function *manageAcqEnd* in Algo. 4 is used at the end of an acquisition acq, and in this case there are at most three decision branches corresponding to the three possible waiting modes. The branch choice is determined by calling function *checkEndCondition*. The latter specifies that using waiting mode w is allowed in two possible cases: (1) w is identical to the default waiting mode that would be used to perform only the remaining mandatory acquisitions; (2) there exists a next best acquisition associated with mode w and the current state of charge is greater than or equal to the right threshold computed on the ground for waiting mode w.

Coming back to function *manageAcqEnd*, the ON waiting mode is the preferred one if it is allowed, then the OFF mode is used if possible, and otherwise the SBY mode is selected. In all cases, the telecommands required to reach the waiting mode chosen are committed and the current waiting mode is updated (Lines 19-28). Last, the CPI waits for the

Algorithm 4: Management of left and right decisions

```
1  Function manageAcqStart($S$, $acqSeq$)
2      $acq \leftarrow acqSeq[S.nextAcqIdx]$
3      if $checkStartCondition(S, acq)$ then
4          if $S.waiting = SBY$ then
5              GEO_POINTING($acq.triggerTime[SBY]$)
6          if $S.waiting \in \{SBY, OFF\}$ then
7              TURN_ON_PAYLOAD($acq.triggerTime[OFF]$)
8          LOAD_ACQ($acq.triggerTime[ON]$, $acq.params$)
9          $S.committedAcqIdx \leftarrow -1$
10         $S.nextDecTime \leftarrow acq.end$
11         $S.nextDecType \leftarrow END$
12     else
13         $S.nextAcqIdx \leftarrow S.nextAcqIdx + 1$
14         $waitNextAcq(S, acqSeq)$

15 Function checkStartCondition($S$, $acq$)
16     return $(acq.level = 0) \vee (acq.idx = S.committedAcqIdx)$
           $\vee (S.soc \geq acq.socMinLeft[S.waiting]))$

17 Function manageAcqEnd($S$, $acqSeq$)
18     $acq \leftarrow acqSeq[S.nextAcqIdx]$
19     if $checkEndCondition(S, acq, ON)$ then
20         UNLOAD_ACQ($acq.end$))
21         $S.waiting \leftarrow ON$
22     else if $checkEndCondition(S, acq, OFF)$ then
23         TURN_OFF_PAYLOAD($acq.end$)
24         $S.waiting \leftarrow OFF$
25     else
26         HELIO_POINTING($acq.end$)
27         TURN_OFF_PAYLOAD($acq.end$)
28         $S.waiting \leftarrow SBY$
29     $S.nextAcqIdx \leftarrow acq.nextAcqIdx[S.waiting]$
30     $waitNextAcq(S, acqSeq)$

31 Function checkEndCondition($S$, $acq$, $w$)
32     if $w = acq.defaultWaitingMode$ then return $true$
33     if $(acq.nextBestAcqIdx[w] < length(acqSeq))$
           $\wedge (S.soc \geq acq.socMinRight[w])$ then
34         $S.committedAcqIdx \leftarrow acq.nextBestAcqIdx[w]$
35         return $true$
36     return $false$
```

decision time associated with the next possible acquisition for the waiting mode chosen (Lines 29-30). This can be seen as a *least commitment strategy*, since the algorithm chooses a waiting mode and not a fixed next acquisition.

Experiments and Validation

The ground conditional planner is implemented in Java and the onboard Conditional Plan Interpreter is implemented in C. We developed prototypes whose goal is to show the efficiency and safety of the global approach proposed. In the experiments presented here, we consider simplified energy models. To define these models, we used technical documents coming from the mission team to get representative parameters. We consider one circular polar orbit split into a day period (Sun visibility) and a night period of duration $T = 3000$ seconds each. During the night period, power production is null. During the day period, the power produced

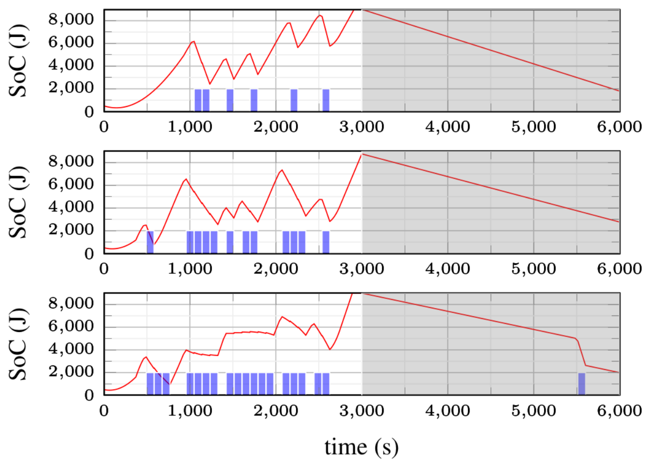
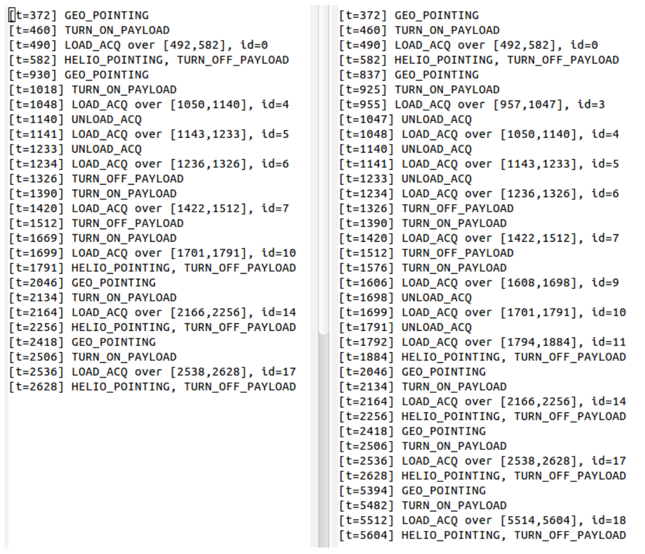


Figure 7: Acquisition plans successively computed for three energy models M_0, M_1, M_2 (in blue), and associated energy evolution profiles (in red)



```
[t=372] GEO_POINTING
[t=460] TURN_ON_PAYLOAD
[t=490] LOAD_ACQ over [492,582], id=0
[t=582] HELIO_POINTING, TURN_OFF_PAYLOAD
[t=930] GEO_POINTING
[t=1018] TURN_ON_PAYLOAD
[t=1048] LOAD_ACQ over [1050,1140], id=4
[t=1140] UNLOAD_ACQ
[t=1141] LOAD_ACQ over [1143,1233], id=5
[t=1233] UNLOAD_ACQ
[t=1234] LOAD_ACQ over [1236,1326], id=6
[t=1326] TURN_OFF_PAYLOAD
[t=1390] TURN_ON_PAYLOAD
[t=1420] LOAD_ACQ over [1422,1512], id=7
[t=1512] TURN_OFF_PAYLOAD
[t=1669] TURN_ON_PAYLOAD
[t=1699] LOAD_ACQ over [1701,1791], id=10
[t=1791] HELIO_POINTING, TURN_OFF_PAYLOAD
[t=2046] GEO_POINTING
[t=2134] TURN_ON_PAYLOAD
[t=2164] LOAD_ACQ over [2166,2256], id=14
[t=2256] HELIO_POINTING, TURN_OFF_PAYLOAD
[t=2418] GEO_POINTING
[t=2506] TURN_ON_PAYLOAD
[t=2536] LOAD_ACQ over [2538,2628], id=17
[t=2628] HELIO_POINTING, TURN_OFF_PAYLOAD
```

```
[t=372] GEO_POINTING
[t=460] TURN_ON_PAYLOAD
[t=490] LOAD_ACQ over [492,582], id=0
[t=582] HELIO_POINTING, TURN_OFF_PAYLOAD
[t=837] GEO_POINTING
[t=925] TURN_ON_PAYLOAD
[t=955] LOAD_ACQ over [957,1047], id=3
[t=1047] UNLOAD_ACQ
[t=1048] LOAD_ACQ over [1050,1140], id=4
[t=1140] UNLOAD_ACQ
[t=1141] LOAD_ACQ over [1143,1233], id=5
[t=1233] UNLOAD_ACQ
[t=1234] LOAD_ACQ over [1236,1326], id=6
[t=1326] TURN_OFF_PAYLOAD
[t=1390] TURN_ON_PAYLOAD
[t=1420] LOAD_ACQ over [1422,1512], id=7
[t=1512] TURN_OFF_PAYLOAD
[t=1576] TURN_ON_PAYLOAD
[t=1606] LOAD_ACQ over [1608,1698], id=9
[t=1698] UNLOAD_ACQ
[t=1699] LOAD_ACQ over [1701,1791], id=10
[t=1791] UNLOAD_ACQ
[t=1792] LOAD_ACQ over [1794,1884], id=11
[t=1884] HELIO_POINTING, TURN_OFF_PAYLOAD
[t=2046] GEO_POINTING
[t=2134] TURN_ON_PAYLOAD
[t=2164] LOAD_ACQ over [2166,2256], id=14
[t=2256] HELIO_POINTING, TURN_OFF_PAYLOAD
[t=2418] GEO_POINTING
[t=2506] TURN_ON_PAYLOAD
[t=2536] LOAD_ACQ over [2538,2628], id=17
[t=2628] HELIO_POINTING, TURN_OFF_PAYLOAD
[t=5394] GEO_POINTING
[t=5482] TURN_ON_PAYLOAD
[t=5512] LOAD_ACQ over [5514,5604], id=18
[t=5604] HELIO_POINTING, TURN_OFF_PAYLOAD
```

Figure 8: Two execution traces obtained from the same conditional plan under different execution conditions

when using an heliocentric pointing is 20W, and the power produced at time $t \in [0, T]$ when using a geocentric pointing is $P(t) = 20{\cdot}cos(-\pi/2 + t\pi/T))$, so that $P(t) = 20$W when the satellite traverses the equator at $t = T/2$. The power produced during a maneuver is obtained by a linear interpolation between the powers produced at the start and end configurations. The power consumed by the payload is 2W when it is off, 10W when it is on without any ongoing acquisition, and 30W during acquisitions. During downlinks, the pessimistic power consumption is approximately 40W, but the evaluation presented here corresponds to one of the numerous orbits without any ground station visibility (no communication). The minimum and maximum energy levels are $Emin = 0J$ and $Emax = 9000J$ respectively, the initial energy level is $500J$ for all energy models, and the minimum level of energy required at the end is $500J$.

The duration of maneuvers is approximated by a constant (2 minutes per maneuver), the durations required to switch the payload on and off are 30 seconds and 1 second respectively, and the durations required to load and unload an acquisition are 2 seconds and 1 second respectively. Over the 100 minutes of an orbit, we define successive candidate acquisitions of duration 90 seconds each, separated by 3 seconds, to cover all ground positions over which the satellite flies. The rank of each acquisition, used on the ground to choose the next acquisition to insert, is randomly chosen.

Fig. 7 shows the acquisition plans and the energy evolution profiles obtained with three energy models M_0, M_1, M_2, where M_1 considers the nominal power values, while M_0 and M_2 respectively take 20% and -20% margins on power consumption. It can be seen that M_1 and M_2 allow to plan optional acquisitions, especially during the day period.

Next, simulations were performed to check the behavior of the onboard CPI. In the final system, the latter will run on a dedicated partition (Time and Space Partitioning paradigm), and the very small runtime and memory footprint of our CPI implementation in C is fully compatible with the specifications of this partition. Fig. 8 gives two lists of time-tagged telecommands produced by the CPI for the same conditional plan, over two distinct executions where the energy available at each decision point is randomly chosen in $[Emin, Emax]$. Such a random choice is not necessarily realistic in terms of energy evolution, but it allows to test the robustness of the approach by generating various execution conditions. The idea here is to make no assumption on the real energy evolution model but the single one that model M_0 is pessimistic. For validation issues, two properties were analyzed on the output of the CPI: first that the basic telecommands produced onboard achieve all mandatory acquisitions planned at level 0, which is a key point for the end-users (the conditional plans must never do worse than the standard conservative plans); and second that these telecommands are identical to the ones that would have been computed on the ground to execute the same acquisitions. For this, 300 input scenarios differing on acquisition ranking were generated, and 100 random executions were performed for each scenario. Simulating these 30.000 executions took about 1 hour on a standard machine (Intel i5 1.2GHz 4GBRAM). Comparisons of output files showed that the two properties analyzed were satisfied for all runs.

Conclusion

This paper presented a planning system for an Earth monitoring nanosatellite whose main bottleneck is the amount of energy available onboard. This system combines offline and online decision making (Filippo, Lombardi, and Milano 2021). It handles temporal constraints and resource constraints, as well as conditional planning and conditional execution. For the last point, the search for a simple and efficient onboard decision process was a task in itself. The two main perspectives are first the simulations on more complex models, and second the forthcoming in-flight experiment.

References

Agrawal, J.; Chi, W.; Chien, S.; Rabideau, G.; Kuhn, S.; Gaines, D.; Vaquero, T.; and Bhaskaran, S. 2021. Enabling Limited Resource-Bounded Disjunction in Scheduling. *Journal of Aerospace Information Systems*, 18(6): 322–332.

Filippo, A. D.; Lombardi, M.; and Milano, M. 2021. Integrated Offline and Online Decision Making under Uncertainty. *Journal or Artificial Intelligence Research*, 70: 77–117.

Lenzen, C.; Wörle, M. T.; Göttfert, T.; Mrowka, F.; and Wickler, M. 2014. Onboard Planning and Scheduling Autonomy within the Scope of the FireBird Mission. In *Proc. of the 13th International Conference on Space Operations (SpaceOps-14)*.

Maillard, A.; Verfaillie, G.; Pralet, C.; Jaubert, J.; Sebbag, I.; and Fontanari, F. 2015. Postponing Decision-Making to Deal with Resource Uncertainty on Earth-Observation Satellites. In *9th International Workshop on Planning & Scheduling for Space (IWPSS'15)*.

Maillard, A.; Verfaillie, G.; Pralet, C.; Jaubert, J.; Sebbag, I.; Fontanari, F.; and L'Hermitte, J. 2016. Adaptable Data Download Schedules for Agile Earth-Observing Satellites. *Journal of Aerospace Information Systems*, 13(8): 280–300.

Peot, M.; and Smith, D. E. 1992. Conditional Nonlinear Planning. In *Proc. of the 1st International conference on Artificial Intelligence Planning Systems (AIPS'92)*, 189–197.

Pryor, L.; and Collins, G. 1996. Planning for Contingencies: a Decision-based Approach. *Journal or Artificial Intelligence Research*, 4: 287–339.