Exploiting opportunistic energy recharging activities within a hybrid ground/onboard planning architecture

Cédric Pralet, David Doose, Julien Anxionnat, Jérémie Pouly

Exploiting opportunistic energy recharging activities within a hybrid ground/onboard planning architecture

Cédric Pralet and **David Doose**
ONERA, Université de Toulouse
2 av. Edouard Belin, BP 74025
F-31055 Toulouse Cedex 4, France

Julien Anxionnat and **Jérémie Pouly**
CNES
18 av. Edouard Belin
31401 Toulouse Cedex 9, France

Abstract

When planning acquisition and downlink activities of a satellite, energy limitations must be taken into account as well as the uncertainty about the actual energy evolution profile. In this context, a basic strategy is to build robust plans on the ground based on conservative margins. Some authors also proposed to compute on the ground conditional plans containing activities activated only when the amount of energy available is high enough. In this work, we go one step further by (1) considering a satellite that can not only cancel activities but also automatically adapt its pointing strategy to maximize power production, (2) computing on the ground activation thresholds that are aware of these opportunistic recharging activities, (3) defining a lightweight embedded mission planner whose behavior is consistent with the assumptions made on the ground. The output of this work is currently a C prototype but there is also an opportunity for an in-flight experiment.

1 Introduction

In this study, we consider a future mission developed by the French Space Agency (CNES), with a launch date planned in 2022. This mission involves a nanosatellite that can perform acquisitions and download data towards ground reception stations. In the basic definition of the system, acquisition and download plans are periodically computed by a mission center located on the ground and sent to the nanosatellite for execution. There is also a key opportunity for an in-flight experiment of an autonomy module. This paper focuses on the development of this module.

In this direction, the first main question is to determine to what extent autonomy can be useful compared to the basic (autonomy-free) system. Another point is to take into account several development constraints, including (1) the need to change as little as possible the format of the basic telecommands uploaded to the satellite, (2) the need for the autonomy module to generate onboard time-tagged telecommands that have the exact same structure as in the autonomy-free system, (3) the impossibility to embed complex thermal or energy models to validate that the plans built on-board are consistent with the resources available, or (4) the absence of onboard functions capable of estimating the maximum time required by a maneuver between two target pointings.

After a system analysis phase, the development of the autonomy module has been focused on the main bottleneck of the system, namely the limited amount of energy available onboard. The main idea is to go beyond the basic full ground planning approach that uses a conservative energy propagation model to guarantee that the plans built are feasible despite the uncertainty about the actual state of charge of the batteries, the power consumed by acquisition and download activities, the actual orientation of the solar panels, or the onboard temperature. To try and better exploit the capabilities of the system, the approach consists of replacing the robust plans currently built on the ground by conditional plans containing activities that are triggered or not depending on the actual state of charge of the batteries.

Using plans conditioned by resource thresholds is not a new idea in the space domain. Indeed, in the VAMOS experiment performed on the DLR BIROS satellite (Wörle and Lenzen 2013; Lenzen et al. 2014), the mission center computes a plan defined by a sequence of plan fragments, with a minimum level of energy required to activate each fragment. In VAMOS, plan fragments can also be directly instantiated onboard following online detections. Conditional acquisition plans were also developed in studies on agile Earth observation satellites (Maillard et al. 2015), with for each acquisition a a precomputation on the ground of an energy threshold required to achieve a and all high-priority acquisitions that follow a given an energy propagation model that takes into account the orientation of the solar panels at any time. Plans containing energy-dependent activities were also proposed for the NASA Mars 2020 rovers (Agrawal et al. 2021). In the latter work, each activity in the plan built on the ground offers a set of ranked alternatives called *switch groups*, with for each alternative an energy activation threshold precomputed on the ground.

In our study, we reuse the idea of producing plans containing activities that can be canceled when the energy level is too low. When computing the energy thresholds on the ground, we however take into account the fact that the satellite is able to perform opportunistic maneuvers to an heliocentric pointing. For instance, starting from the ground plan provided in Figure 1a, the onboard planning system that we define can end up with the plan described in Figure 1b, where some optional activities are canceled *and* the pointing strategy is updated to maximize power production. Our goal

is to exploit such a pointing adaptation capability so as to compute on the ground energy thresholds that are more permissive. Doing so, the boundary between the onboard plan adaptation procedure and the ground planner is pushed one step further. A strong constraint related to this hybridization is that the ground mission center must be able to anticipate the possible decisions made by the satellite. This is why we consider lightweight deterministic onboard adaptation rules instead of a complex deliberation engine.

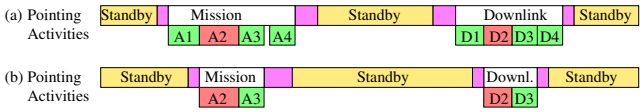

Figure 1: (a) Plan initially sent by the ground mission center (mandatory activities in red, optional activities in green); (b) plan obtained after canceling some optional activities and updating the pointing strategy accordingly

The rest of paper is organized as follows. Section 2 describes the current autonomy-free system. Section 3 details the ground process used to compute energy thresholds. Section 4 details the content of the lightweight onboard mission planner. Section 5 gives some perspectives for this work.

2 Features of the autonomy-free system

We first describe some features of the satellite and then the standard full ground planning approach for the mission.

Platform The platform can be pointed to several directions to achieve its mission and has three *stable* pointing modes:

- `STANDBY`: heliocentric pointing, for maximizing the recharge of the batteries through the solar panels;
- `MISSION`: geocentric pointing for the mission antenna, when performing acquisitions;
- `DOWNLINK`: telemetry antenna pointed to a ground station.

Acquisition and downlink cannot be performed in parallel. As shown in Figure 2, apart from these stable modes, there also exist *transient* pointing modes that correspond to maneuvers. Transitions between pointing modes are triggered by a telecommand referred to as `TC_PF_PTG`. For instance, from the *Standby* pointing mode, the execution of a `TC_PF_PTG`(*Mission*) pointing telecommand triggers a transition to the `MANEUVER_SM` pointing mode and then, after some time, to the *Mission* pointing mode. The parameters of `TC_PF_PTG` are a target pointing mode (`mode`) in $\{Mission, Downlink, Standby\}$ and a station identifier (`stationId`). The latter is useful only when the target pointing mode is *Downlink*. In addition to these two parameters, each `TC_PF_PTG` telecommand has a fixed execution time. Also, in the specification of the mission, the satellite necessarily uses a *Standby* pointing to recharge its batteries between a *Mission* pointing and a *Downlink* pointing.

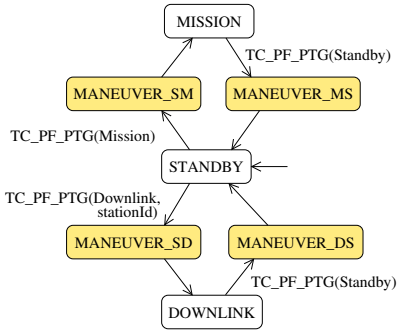

Figure 2: Platform features

Payload The payload has several possible running modes:

- `OFF` (stable mode): payload switched off;
- `ON` (stable mode): payload switch on and without any ongoing activity;
- `OFF_ON` (transient mode): transition from OFF to ON;
- `ON_OFF` (transient mode): transition from ON to OFF;
- `ACQUISITION` (transient mode): ongoing acquisition;
- `DOWNLOAD` (transient mode): ongoing download activity toward a ground reception station;
- `REMOVE` (transient mode): ongoing deletion of an acquisition recorded in the mass memory.

As shown in Figure 3, the running mode of the payload is impacted by several telecommands. The `TC_CU_ON` and `TC_CU_OFF` telecommands respectively allow to switch on and off the payload. The `TC_CU_ACQ`, `TC_CU_DL`, and `TC_CU_RM` telecommands respectively allow to trigger an acquisition, a data download, and a data erasure. They all have an acquisition identifier `acquisitionId` $\in [0, 49]$ as a parameter. Additionally, the acquisition telecommand has a specific `duration` parameter, and the download telecommand defines a ground station identifier (`stationId`). All payload telecommands have a fixed execution time. In the following, we consider that data can be systematically erased after they have been downloaded.

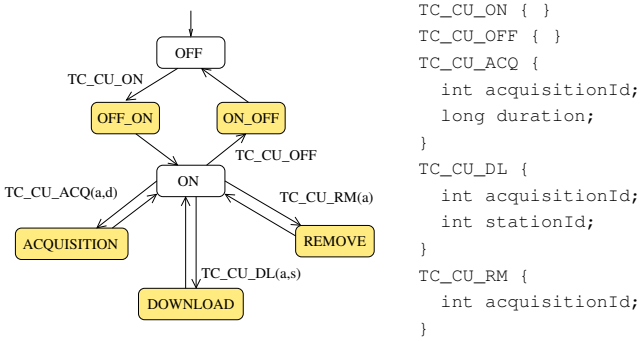

Figure 3: Payload features

On-board resources Mission data is collected in mass memory. The latter has a fixed capacity and at most 50 files can be stored at any time (50 file identifiers available). Moreover, the capacity of the batteries is limited. A complex energy model is used to check whether a given plan is feasible. It takes into account the power consumed in each payload mode and the power produced depending on the orientation of the solar panels, the onboard temperature, or the state of charge of the batteries itself. In this model, margins are used to guarantee the feasibility of the plan, such as a 15% margin on the actual orientation of the solar panels.

Existing ground planning algorithm Plans are typically produced on the ground for a one-day time frame. To build each plan, the basic planning algorithm works as follows.

- Initially, the algorithm starts from an empty plan and computes all candidate acquisitions. The latter all have fixed start and end times, and they are ranked based on a specific ranking policy that we do not detail here.
- At each step, the candidate acquisition a that has the highest rank is inserted into the current plan. Acquisition a is possibly merged with the acquisition b that precedes a and/or the acquisition c that follows a in the plan, depending on the temporal distance between b and a and between a and c; this acquisition merging step allows to limit the number of mass memory file identifiers that are used.
- Following each acquisition insertion, a download plan is rebuilt based on a fixed "first acquired - first downloaded" decision rule. The pointing plan of the satellite is rebuilt as well, based on the following rules: (1) the satellite must have a *Mission* pointing during acquisitions and a *Downlink* pointing during data download; (2) a *Standby* pointing is introduced between two *Mission* pointings or two *Downlink* pointings if and only if the duration of this *Standby* pointing is not less than a minimum fixed duration $DuMinStandby$; (3) the *Standby* pointing periods must be as long as possible. Similar decision rules are used to determine when the payload must be switched on or off, in particular based on a parameter $DuMinOff$ defining the minimum duration required for an off period.
- At any step, if one of the system constraints is violated (number of file identifiers exceeded, mass memory capacity exceeded, energy level lower than a given limit according to the energy propagation model), acquisition a is rejected, the plan before the insertion of a is restored, and the algorithm goes to the next step.

The algorithm stops when there are no more candidate acquisitions. The plan obtained is then transformed into a sequence of time-tagged telecommands.

3 Ground computation of recharge-aware energy thresholds

We now introduce the ground part of the collaborative ground-onboard planning process proposed to replace the current full ground planning approach. More specifically, we detail the computation of the energy activation thresholds given the autonomous pointing adaptation strategy. In the following, we consider that the energy level can vary within interval $[Emin, Emax]$, with $Emin$ the minimum energy level acceptable and $Emax$ the energy level that corresponds to a maximum state of charge of the batteries.

3.1 Forward energy propagation model

To check whether a given plan π defined over time frame $[Start_\pi, End_\pi]$ is feasible starting from initial energy level $E_0 \in [Emin, Emax]$ at time $Start_\pi$, we dispose of a function Φ that gives the energy evolution profile obtained over $[Start_\pi, End_\pi]$. More formally, Φ_{π,E_0} takes as an input a time $t \in [Start_\pi, End_\pi]$ and returns a value $\Phi_{\pi,E_0}(t) \in [0, Emax]$ standing for the energy level obtained at time t when applying π starting from initial energy level E_0. Plan π is said to be consistent if $\Phi_{\pi,E_0}(t) \geq Emin$ holds for every time $t \in [Start_\pi, End_\pi]$. The content of function Φ_{π,E_0} can be based on any more or less complex physical model. In the current basic ground planning system, it takes into account parameters like the power produced through the solar panels, the power consumed at any time by the platform and the payload given the time-tagged acquisition and download activities in π, and the saturation of the level of energy when it reaches maximum level $Emax$. For these parameters, some margins can be considered. For example, as mentioned before, the basic ground planning algorithm uses a margin of 15% on the orientation of the solar panels when evaluating power production at a given step.

3.2 Backward energy propagation

On the other way around, assume that we are given a minimum energy level E required at time End_π. The set $\Theta_{\pi,E}$ of initial energy levels that allow to meet this requirement can be defined as:

$$\Theta_{\pi,E} = \{e \in [Emin, Emax] \,|\, (\Phi_{\pi,e}(End_\pi) \geq E) \quad (1)$$
$$\wedge (\forall t \in [Start_\pi, End_\pi],\ \Phi_{\pi,e}(t) \geq Emin)$$

Let us assume that $\Theta_{\pi,E}$ is not empty, or equivalently that the requirements can be met when the initial energy level equals $Emax$. Then, the minimum energy level $\Psi_{\pi,E}$ needed at $Start_\pi$ to meet the requirement is defined as:

$$\Psi_{\pi,E} = \min(\Theta_{\pi,E}) \quad (2)$$

If $Emin \in \Theta_{\pi,E}$, then we have $\Psi_{\pi,E} = Emin$, otherwise quantity $\Psi_{\pi,E}$ can be computed by an iterative method that starts from extreme energy levels $e_1 = Emin$ and $e_2 = Emax$ and tries to converge to the minimum value of $\Theta_{\pi,E}$ either through dichotomic search or through interpolation methods for finding the minimum zero of function f defined by:

$$f(e) = \min(\ \Phi_{\pi,e}(End_\pi) - E, \quad (3)$$
$$\min_{t \in [Start_\pi, End_\pi]}(\Phi_{\pi,e}(t) - Emin))$$

Alternatively, by partitioning interval $[Start_\pi, End_\pi[$ into small intervals $[t_0, t_1[, \ldots, [t_{K-1}, t_K[$ and assuming that the power produced over $[t_k, t_{k+1}[$ is a constant μ_k, it can be shown that the minimum level of energy β_k required at time t_k can be obtained through a backward recursion:

$$\beta_k \leftarrow \max(Emin, \beta_{k+1} - \mu_k(t_{k+1} - t_k)) \quad (4)$$

starting from $\beta_K = E$, and then we can take $\Psi_{\pi,E} = \beta_0$.

3.3 Computation of energy activation thresholds

The plan π considered is actually composed of a sequence of acquisition and download activities $[a_1, \ldots, a_n, a_{n+1}]$ where each activity a_i has a fixed start time referred to as $start(a_i)$ and a fixed end time referred to as $end(a_i)$. As shown later, activities in π are either mandatory or optional. Activity a_{n+1} is a fictitious mandatory activity that has a null duration and is placed at the end of the time frame covered by π. Our goal is then to determine, for each optional activity a_i, the minimum level of energy $\theta(a_i)$ required at time $start(a_i)$ to execute a_i and all mandatory activities that follow a_i in π, and end up with a final energy level greater than or equal to a lower bound E.

To do this, we can start from $\theta(a_{n+1}) = E$ and use a backward propagation process to compute step-by-step $\theta(a_n), \theta(a_{n-1}), \ldots, \theta(a_1)$. More precisely, given an activity a_i, let a_j be the next mandatory activity placed after a_i. Let $\pi[a_i \rightarrow a_j]$ be the fully instantiated plan obtained over interval $[start(a_i), start(a_j)]$ if all optional activities placed between a_i and a_j are canceled. This complete plan defines all switch on/off telecommands sent to the payload, as well as a pointing strategy that depends on whether the time gap between a_i and a_j can contain a *Standby* pointing of minimum duration $DuMinStandby$. The activation threshold $\theta(a_i)$ associated with a_i is then the minimum level of energy that the satellite must have at time $start(a_i)$ to reach time $start(a_j)$ with an energy level greater than or equal to $\theta(a_j)$, that is:

$$\theta(a_i) = \Psi_{\pi[a_i \rightarrow a_j], \theta(a_j)} \quad (5)$$

with Ψ the backward propagation term introduced before. Note that while resource propagation is often used in timeline-based planning systems (Chien et al. 2012), using backward propagation for extending the conditions under which a plan remains applicable is less common, one exception being the DLR BIROS approach mentioned before.

3.4 Updated ground planning algorithm

We now detail the way the ground mission planner can plan optional activities and compute energy activation thresholds that are aware of the autonomous pointing adaptation strategy used onboard.

For this, we consider on one hand the default *conservative energy model* $\mathcal{M}_1$ used by the current ground planner, and on the other hand an *expected energy model* $\mathcal{M}_2$. The latter can be obtained by reducing the margins taken with regards to the precise orientation of the solar panels or by increasing the maximum power that can be produced through the solar panels. It can also correspond to an expected model whose parameters are learned from actual energy evolution profiles observed in telemetry measures.

Then, the new ground planning algorithm proposed works as follows. At phase 1, the ground planner computes a plan π_1 based on the standard ground planning algorithm that uses energy model $\mathcal{M}_1$. Acquisition and download activities in π_1 are called the *mandatory activities*. Once plan π_1 is obtained, the minimum energy level $\theta(a)$ associated with each acquisition or download activity a involved in π_1 is computed, based on the techniques introduced before.

At phase 2, energy model $\mathcal{M}_1$ is replaced by $\mathcal{M}_2$, hence there may be opportunities for inserting new activities into the plan. For this, all acquisitions rejected during phase 1 are considered again one by one, from highest to lowest rank. During phase 2, the ground planner is forced to keep all mandatory acquisition and download activities selected during the first phase as well as the precise dates at which they are scheduled. For each acquisition a that we try to insert during phase 2, we use the standard ground planning algorithm with the three slight modifications listed below.

- First, a cannot be merged with acquisitions selected during phase 1, since merging optional activities with mandatory ones based on energy model $\mathcal{M}_2$ could jeopardize the feasibility of the mandatory acquisitions.
- Second, we forbid acquisition merging operations that would lead to acquisitions whose duration is greater than a maximum duration $DuMaxOptional$. The goal here is to have a granularity that allows the system to add optional activities anytime it is possible.
- Third, as in the standard algorithm, a download plan and a pointing strategy are built after each acquisition insertion. This allows to determine whether the memory capacity constraint, the identifier capacity constraint, and the energy constraint (based on model $\mathcal{M}_2$) are satisfied. If no, the insertion of a is rejected. If yes, one additional check is performed: for each acquisition or download activity a introduced so far during phase 2, we check that if the level of energy is equal to $Emax$ at the start time of a, it is still possible, *according to conservative model* $\mathcal{M}_1$, to reach the next mandatory activity b that follows a with a level of energy greater than or equal to energy threshold $\theta(b)$. For this computation, we remove all activities placed between a and b and we adapt the pointing strategy accordingly. The objective here is to ensure that triggering a does not threaten future mandatory activities.

At the end of phase 2, we obtain a new plan π_2. All activities added to this plan during phase 2 are called *optional*. For each of them, we compute the energy threshold $\theta(a)$ required at the start time of a to perform a and all future mandatory activities. By construction, it is sure that at least $\theta(a) = Emax$ is consistent, but the backward propagation algorithm can be used to get a less restrictive bound.

Figure 4 illustrates the energy activation thresholds computed for a specific plan that contains two acquisition activities (A1, A2), two download activities (D1, D2), *Standby* pointings, and a night period for the satellite (null power production from the solar panels). Figure 4b gives the energy profile obtained from conservative model $\mathcal{M}_1$. This profile is inconsistent since the level of energy is strictly less than $Emin$ at a given step. However, by considering that activities A2/D2 are optional, it is possible to send this plan to the satellite while specifying the minimum level of energy $\theta(a)$ required to execute each optional activity a and the mandatory activities that follow. Figures 4c to 4f illustrate these minimum energy levels and the corresponding energy profiles obtained over the rest of the planning horizon. Thresholds $\theta(D2), \theta(D1), \theta(A2), \theta(A1)$ are computed based on the backward propagation process presented before.

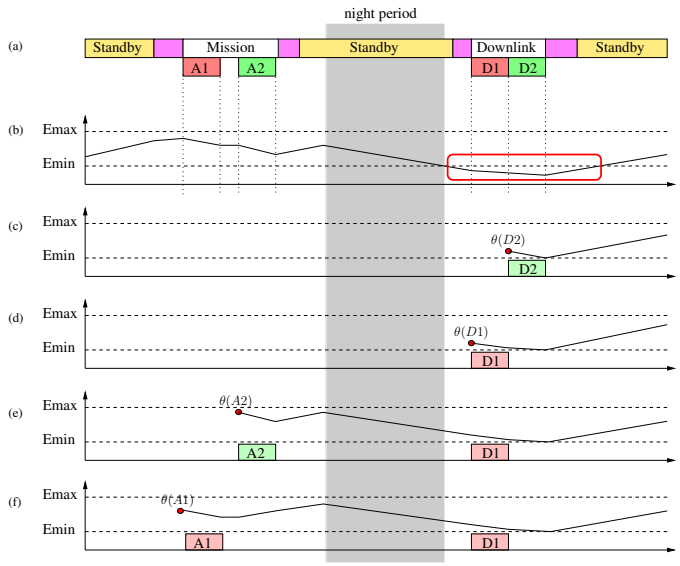

Figure 4: Energy thresholds computed on the ground

4 Lightweight onboard plan adaptation

We now detail the onboard planner developed to trigger or not the acquisition and download activities, and to adapt the pointing strategy in the exact same way as predicted by the ground mission center.

4.1 Extended TC format

As a first step, the basic telecommands TC_CU_ACQ and TC_CU_DL presented before are slightly extended to cover optional activities. This extension leads to telecommands TC_CU_ACQ+ and TC_CU_DL+ given in Figure 5. For both of these telecommands, three parameters are added compared to the standard ones:

- `energyThreshold`: a minimum energy level required at the start time of the corresponding acquisition or download activity (value -1 if the activity is mandatory);
- `maxDuFromStandby`: a maximum duration required by the platform to make a transition from the *Standby* pointing to the *Mission* (resp. *Downlink*) pointing so as to reach this pointing at the beginning of the acquisition (resp. download); for an acquisition or download activity a, this parameter is used to determine, without any maneuver duration estimation onboard, the latest time at which the *Standby* pointing can be left before performing a;
- `maxDuToStandby`: a maximum duration required by the platform to make a transition from the *Mission* (resp; *Downlink*) pointing to the *Standby* pointing just after the corresponding activity; this parameter is used to determine, without any maneuver duration estimation onboard, the earliest time at which the *Standby* pointing can be reached after performing the activity.

4.2 System state evolution model

To apply decision rules, the onboard planner uses a representation of the state of the satellite based on the attributes given in Table 1, together with functions written in C that describe how each telecommand makes the system state evolve over time and how the system state evolves as time flows. The descriptions used within these functions are close to *timed automata* (Alur and Dill 1994), but some points such as the management of the level of energy are closer to *hybrid automata* (Henzinger 1996).

```
TC_CU_ACQ+ {
  int acquisitionId;
  long duration;
  double energyThreshold;
  long maxDuFromStandby;
  long maxDuToStandby;
}
TC_CU_DL+ {
  int acquisitionId;
  int stationId;
  double energyThreshold;
  long maxDuFromStandby;
  long maxDuToStandby;
}
```

Figure 5: Extended TC format

Having an explicit representation of the system state and of the impact of the telecommands on this state allows to check that the detailed TC plans produced onboard are consistent. If this is not the case, for instance if the current level of energy is too low, state attribute `valid` is set to "false". Said differently, the onboard deliberation engine embeds a model of a discrete event dynamic system to verify plans. Such an approach has been recently proposed in a work related to the verification of TC plans produced onboard an autonomous satellite (Mussot et al. 2020). In this work, a component named *telecommand verifier* checks the validity of TC plans produced onboard with regards to a formal model of the system expressed in a specific language called CSM (*Compact Satellite Model*), from which Lustre code and then C code can be automatically generated. The CSM language however seems to be focused on logical state attributes, which does cover numerical attributes such as the amount of memory or energy available at a given step. Also, the CSM language does not seem to deal with table data structures, that are for instance required in our case to indicate whether acquisition data is recorded and whether it has already been downloaded (see Table 1).

4.3 Planning loop

The onboard deliberation engine receives as an input a plan defined by a sequence of high level acquisition and download telecommands (sequence of TC_CU_ACQ+ and TC_CU_DL+ telecommands). At execution time, it must both filter these telecommands to remove optional activities that should not be triggered due to an insufficient level of energy, and detail the remaining telecommands so as to produce fully instantiated plans containing only standard telecommands (TC_CU_ACQ, TC_CU_DL, TC_CU_ON, TC_CU_OFF, TC_CU_RM, TC_PF_PTG), as in the plans sent by the ground in the autonomy-free system. Note that expanding so-called *synthetic telecommands* into

Attribute	Type	Semantics
`time`	`long`	current time
`energyLowerBound`	`double`	lower bound on the energy level at the current time
`memoryAvailable`	`long`	memory size currently available
`recorded`	`bool[N_IDS]`	acquisition data recorded or not, for each acquisition identifier
`downloaded`	`bool[N_IDS]`	acquisition data already downloaded or not, for each acquisition identifier
`volume`	`long[N_IDS]`	memory size occupied, for each acquisition identifier
`pfMode`	`PF_Mode`	current pointing mode of the platform
`pointedStationId`	`int`	identifier of the station pointed at (useful only for the *Downlink* pointing mode)
`activePfClock`	`bool`	ongoing maneuver to a stable pointing mode
`pfClockNextTime`	`long`	time at which the ongoing maneuver will be over (if any)
`maxDuToStandby`	`long`	maximum duration required to come back to a *Standby* pointing at the end of the ongoing acquisition or download activity (if any)
`cuMode`	`CU_Mode`	current mode of the payload
`ongoingAcqId`	`int`	identifier of the file concerned by the ongoing acquisition or download (if any)
`activeCuClock`	`bool`	ongoing transition to a stable state for the payload
`cuClockNextTime`	`long`	time at which the ongoing state transition will be over (if any)
`valid`	`bool`	validity of the current state with regards to the state evolution model

Table 1: System state attributes manipulated in the onboard state evolution model

detailed telecommands for Earth observation satellites is not a new concept (Pouly, Jouanneau, and Olhagaray 2014).

To achieve this task, the onboard planner is regularly called. When it is called at time T_0, some telecommands are already committed (transmitted to the executive layer) up to a given time $T_1 = T_0 + \Delta$, where Δ stands for the *latency* of the planning loop, that is the time required to produce detailed telecommands at each onboard planning episode. The planning loop is then responsible for producing a plan over time frame $[T_1, T_2[$, with $T_2 \geq T_1 + \Delta$ (see Figure 6). As shown thereafter, the default setting is $T_2 = T_1 + \Delta$ but to deal with activities placed at the end of the current decision horizon, time T_2 must sometimes be extended a bit. Telecommands computed over $[T_1, T_2[$ are then committed, and the planning loop is called again at time $T_2 - \Delta$. In practice, latency Δ can be low (a few seconds), therefore time interval $[T_1, T_2[$ will involve only a few candidate synthetic telecommands.

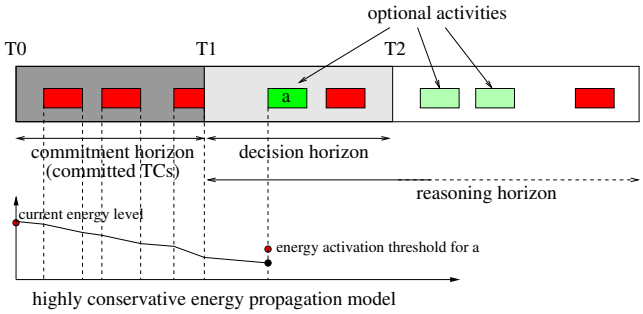

Figure 6: Horizons manipulated by the onboard planner

To make decisions over $[T_1, T_2[$, each planning episode uses the following steps.

- (*Observation step*) From the onboard system, get information about the actual state of the satellite at T_0, especially concerning the current energy level E_0 at that time.
- (*State projection step*) From the sequence of TCs already committed over $[T_0, T_1[$, estimate the system state obtained at T_1, especially the energy level E_1 at that time; in the prototype developed, E_1 is obtained based on a very conservative energy model that only takes into account the maximum power consumed by the payload at each time in $[T_0, T_1[$ (null power production assumption).
- (*Synthetic TC filtering/expansion*) Consider synthetic TCs involved in $[T_1, T_2[$ chronologically; for each of them:
  - determine the (unique) sequence of basic TCs $TcSeq$ that must be used to put the platform and the payload in a configuration where the acquisition or download TC can be triggered; sequence $TcSeq$ is obtained from the fixed decision rules that detail when the satellite should come back to a *Standby* pointing and when it should switch the payload on or off;
  - simulate the successive application of all TCs in $TcSeq$, starting from the current state s of the system; this simulation leads to a new state s' at the end time of the acquisition or download activity considered; for this simulation, use the highly conservative energy model that ignores power production;
  - if the synthetic TC a considered is optional and state s' is not valid (*e.g.* if the level of energy at the beginning of a is less than threshold $\theta(a)$ computed on the ground, or if the data to download is not stored in the mass memory), then a is rejected, otherwise it is accepted, the detailed TCs in $TcSeq$ are added to the output sequence of telecommands, and the new current state becomes s'; on the example provided in Figure 6, the optional activity involved in $[T_1, T_2[$ would be rejected because the (pessimistic) state projection at the beginning of this activity leads to an energy level that is strictly less than the threshold computed on the ground. A data deletion telecommand is also systematically added at the end of the download activities, even if the download activity is rejected.

- (*Management of the end of the decision horizon*) Make decisions on the platform pointing strategy and the payload on/off strategy to use just after the last activity considered (more details in Section 4.4).

The planning loop defined above is inspired by the NASA JPL CASPER planning system (Chien et al. 2000; Knight et al. 2001). The latter has been validated in over more than ten years of flight on EO-1 (Chien et al. 2005). It involves a state projection phase and a flaw resolution phase during which the flaws identified in the current plan are dealt with one by one. In our case, each synthetic TC can be seen as a flaw resolved through TC expansion or TC canceling. One difference is that in our case, state projection and flaw resolution are performed step-by-step in window $[T_1, T_2[$.

4.4 Management of the decision horizon end

In real-time planning systems, the management of the end of the decision horizon is always a tricky part, since commands might be needed to prepare the system for activities that are placed *after* the end of the current decision horizon. In the following, we only discuss the pointing strategy, but the management of the payload switch on / switch off decisions can be dealt with similarly.

Let us for instance assume that in Figure 7a, acquisition A1 that is involved in $[T_1, T_2[$ is accepted. Even if there are no more synthetic TCs over $[T_1, T_2[$, the onboard autonomy module must choose a pointing strategy between the end of A1 and time T2. To do this, the next candidate acquisition A2 is considered. If it is optional and the duration between the end of A1 and the start of A2 does not suffice to come back to a *Standby* pointing between the two acquisitions, the onboard planner simulates the execution of A2 and evaluates whether the level of energy obtained at the beginning of A2 is necessarily greater than or equal to threshold $\theta(A2)$ computed on the ground. If yes (Figure 7b), acquisition A2 is accepted, the *Mission* pointing is maintained until the end of A2, all telecommands associated with the expansion of A2 are committed, and time T_2 is extended until the end of A2. Otherwise, A2 is rejected and the next candidate acquisition (A3) is considered. On this example, A3 is sufficiently far to come back to a *Standby* pointing after A1. This *Standby* pointing is maintained for the minimum duration required, and time T_2 is updated accordingly (Figure 7c). It is not necessary to decide at this step whether A3 should be performed: it is indeed better to wait for information about the real level of energy available shortly before A3. The case of the *Downlink* pointing is dealt with similarly.

More generally, if the satellite does not have a *Standby* pointing after processing the last synthetic TC involved in $[T_1, T_2[$, then the algorithm considers the next candidate TC after T_2. If this next TC requires a pointing change or is placed sufficiently far according to the $DuMinStandby$ parameter, then a *Standby* pointing telecommand is added as early as possible after the detailed TCs already produced over $[T_1, T_2[$. Otherwise, the algorithm analyzes whether the next TC is feasible according to the energy thresholds. If yes, this TC is accepted and time T_2 is delayed, otherwise it is rejected and the algorithm considers the next TC provided by the ground if there exists one.

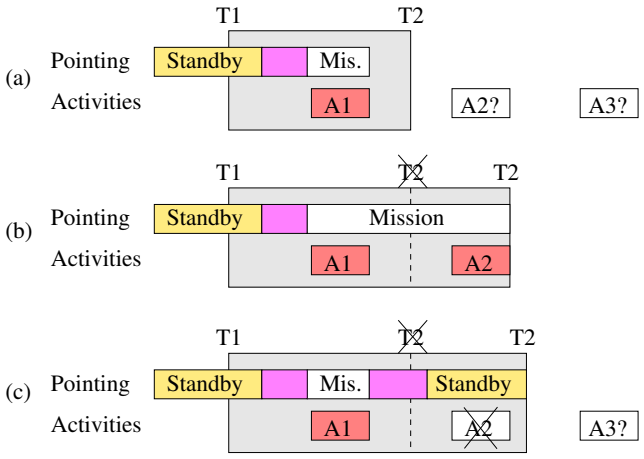

Figure 7: Management of the end time of the planning horizon from the *Mission* or *Download* pointing mode

We now consider the case depicted in Figure 8a, where the satellite has a *Standby* pointing after processing the candidate acquisition and download activities involved in $[T_1, T_2[$. In this case, we must take into account the first activity placed after T_2, namely A1. Let us assume that A1 corresponds to an acquisition activity. If the realization of A1 requires triggering a maneuver to the *Mission* pointing before T_2, then the algorithm determines whether A1 should actually be kept, by simulating the application of A1 and by comparing the energy level obtained at the beginning of A1 with the energy threshold $\theta(A1)$ computed on the ground. If acquisition A1 is accepted, the transition to the *Mission* pointing is triggered as late as possible before A1, the *Mission* pointing is committed from the start of A1 to the end of A1, and time T_2 is set to the end of A1 (Figure 8b). Otherwise, A1 is rejected and the next activity, A2, is considered. If A2 is sufficiently far from T_2, the *Standby* pointing is maintained until T_2 and it is not necessary to make an immediate decision on the execution of A2 (Figure 8c).

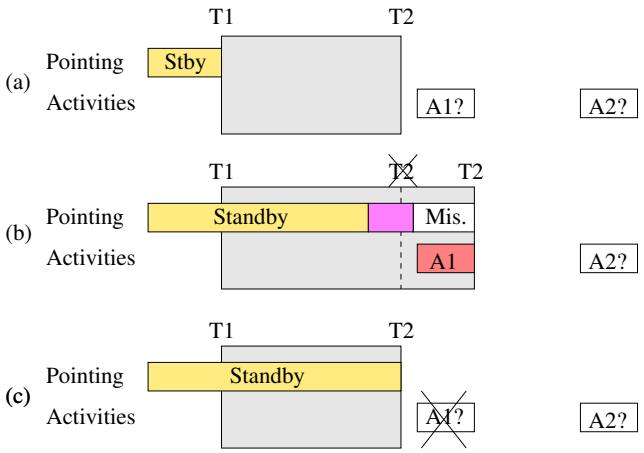

Figure 8: Management of the end time of the planning horizon from the *Standby* pointing mode

More generally, if the platform has a *Standby* pointing after the last TC processed in $[T_1, T_2[$, the algorithm considers the first candidate activity a placed after T_2. If this next TC is placed sufficiently close to T_2, that is if it requires to trigger a maneuver before T_2, then the algorithm analyzes the feasibility of this TC according to the energy threshold $\theta(a)$ computed on the ground. If it is feasible, then it is accepted and time T_2 is extended until the end time of a. Otherwise, the TC is rejected and the next TC, if any, is analyzed.

4.5 Execution trace

The lightweight planning loop proposed has been implemented in a prototype written in C. We provide hereafter an execution trace obtained from the ground plan given in Figure 9a. This plan is composed of synthetic TCs that are handled by the autonomy module. Figure 9b gives the expanded version of it obtained when all activities are activated. On the ground, the synthetic plan is computed over time interval $[0, 1000]$, and we consider a planning loop latency Δ equal to 30 time units for the onboard system. The first parameter of each TC is the execution time of the required activity, and the other parameters are given in the same order as in Figures 2, 3, and 5. Hence, in the complete version of the plan, the payload must be switch on at time $t = 100$. Then, at $t = 150$, a maneuver to the *Mission* pointing must be triggered (mode 1 of the platform). At $t = 170$, acquisition of identifier 2 must start, for 10 time units. A "-1" energy threshold indicates that there is no threshold (mandatory activity). At $t = 180$, acquisition number 3 must start, for 20 time-units. At $t = 200$, a maneuver to the *Standby* pointing must be triggered (mode 0 of the platform). At $t = 600$, the satellite must trigger a maneuver to the *Downlink* pointing (mode 2 of the platform), the goal being to point to station number 0. At $t = 680$, a download activity must be triggered for acquisition number 2, and the corresponding data is deleted at $t = 730$. At $t = 900$, acquisition 3 is downloaded and the corresponding data is deleted at $t = 950$. At $t = 960$, the payload is switched off and the satellite comes back to a *Standby* pointing.

```
TC_CU_ACQ+(170,2,10,-1,10,10)
TC_CU_ACQ+(180,3,20,5.2,10,10)
TC_CU_DL+(680,2,0,-1,10,10)
TC_CU_DL+(900,3,0,0.0,10,10)
```

(a)

```
TC_CU_ON(100)
TC_PF_PTG(150,1,-1)
TC_CU_ACQ(170,2,10)
TC_CU_ACQ(180,3,20)
TC_PF_PTG(200,0,-1)
TC_PF_PTG(600,2,0)
TC_CU_DL(680,2,0)
TC_CU_RM(730,2)
TC_CU_DL(900,3,0)
TC_CU_RM(950,3)
TC_CU_OFF(960)
TC_PF_PTG(960,0,-1)
```

(b)

Figure 9: Plans computed on the ground: (a) synthetic TC plan; (b) fully instantiated plan

Figure 10 gives the execution trace obtained when the onboard planner is provided with the synthetic plan of Figure 9a. It is possible to see that acquisition number 3 is canceled, and after that the satellite comes back earlier to the *Standby* pointing (maneuver to *Standby* initiated at $t = 180$ instead of waiting for $t = 200$). A similar remark holds for data download for acquisition 3 that is also canceled.

```
Plan [curTime=0, commitEndTime=40, decisionEndTime=70]
Plan [curTime=40, commitEndTime=70, decisionEndTime=100]
Plan [curTime=70, commitEndTime=100, decisionEndTime=130]
Plan [curTime=100, commitEndTime=130, decisionEndTime=160]
Plan [curTime=130, commitEndTime=160, decisionEndTime=190]
ACCEPT synthetic TC: TC_CU_ACQ+(170,2,10,-1,10,10)
REJECT synthetic TC: TC_CU_ACQ+(180,3,20,5.2,10,10)
SEND TC_PF_PTG(160,1,-1)
SEND TC_CU_ON(165)
SEND TC_CU_ACQ(170,2,10)
SEND TC_CU_OFF(180)
SEND TC_PF_PTG(180,0,-1)
Plan [curTime=160, commitEndTime=480, decisionEndTime=510]
Plan [curTime=480, commitEndTime=510, decisionEndTime=540]
Plan [curTime=510, commitEndTime=540, decisionEndTime=570]
Plan [curTime=540, commitEndTime=570, decisionEndTime=600]
Plan [curTime=570, commitEndTime=600, decisionEndTime=630]
Plan [curTime=600, commitEndTime=630, decisionEndTime=660]
Plan [curTime=630, commitEndTime=660, decisionEndTime=690]
ACCEPT synthetic TC: TC_CU_DL+(680,2,0,-1,10,10)
SEND TC_PF_PTG(670,2,0)
SEND TC_CU_ON(675)
SEND TC_CU_DL(680,2,0)
Plan [curTime=660, commitEndTime=730, decisionEndTime=760]
SEND TC_PF_PTG(730,0,-1)
Plan [curTime=730, commitEndTime=760, decisionEndTime=790]
Plan [curTime=760, commitEndTime=790, decisionEndTime=820]
Plan [curTime=790, commitEndTime=820, decisionEndTime=850]
Plan [curTime=820, commitEndTime=850, decisionEndTime=880]
Plan [curTime=850, commitEndTime=880, decisionEndTime=910]
REJECT synthetic TC: TC_CU_DL+(900,3,1,0,10,10)
SEND TC_CU_OFF(880)
Plan [curTime=880, commitEndTime=910, decisionEndTime=1000]
```

Figure 10: Execution trace

5 Conclusion and perspectives

This paper presented a hybrid architecture composed of a ground planning system that computes plans whose activities can be conditioned by energy resource thresholds, and a light onboard planner that is able to perform opportunistic energy recharge activities to improve the performance. With regards to previous works, one originality is that the plan is not decomposed into predefined fragments, and the ground planning system exploits the fact that the onboard planning system can adapt its pointing strategy when activities are canceled. As other authors (Filippo, Lombardi, and Milano 2021), we believe that such hybrid online/offline architectures can be highly relevant to increase system performance. Some efforts were also performed to guarantee that the plans reconstructed onboard are consistent with regards to a discrete event dynamic model of the system. As mentioned initially, there is an opportunity for an in-flight experiment onboard a nanosatellite, but before that tests should be performed and validation should be pushed one step further.

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):322332.

Alur, R., and Dill, D. 1994. A theory of timed automata. *Theoretical Computer Science* 126:183–235.

Chien, S.; Knight, R.; Stechert, A.; Sherwood, R.; and Rabideau, G. 2000. Using iterative repair to improve responsiveness of planning and scheduling. In *International Conference on Artificial Intelligence Planning Systems (AIPS-00)*.

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davies, A.; Mandl, D.; Frye, S.; Trout, B.; Shulman, S.; and Boyer, D. 2005. Using Autonomy Flight Software to Improve Science Return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication (JACIC)* 196–216.

Chien, S.; Johnston, M.; Policella, N.; Frank, J.; Lenzen, C.; Giuliano, M.; and Kavelaars, A. 2012. A generalized timeline representation, services, and interface for automating space mission operations. In *International Conference On Space Operations (SpaceOps-12)*.

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.

Henzinger, T. 1996. The theory of hybrid automata. In *Proc. of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS-96)*, 278–292.

Knight, R.; Rabideau, G.; Chien, S.; Engelhardt, B.; and Sherwood, R. 2001. CASPER: Space exploration through continuous planning. *Intelligent Systems IEEE* 16(5):70–75.

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 *IWPSS'15*.

Mussot, V.; Zilio, S. D.; Correnson, L.; Rainjonneau, S.; Bardout, Y.; and Scano, G. 2020. Formal Approach for the Verification of Onboard Autonomous Functions in Observation Satellites. In *Proceedings of the 10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*.

Pouly, J.; Jouanneau, S.; and Olhagaray, P. 2014. Autonomous mission planning in space : Mission benefits and real-time performances. In *ERTS*^{2}.

Wörle, M., and Lenzen, C. 2013. Ground assisted onboard planning autonomy with VAMOS. In *IWPSS'13*.