Autonomous Planning for an Agile Earth-Observing Satellite

Grégory Beaumet
ONERA
Gregory.Beaumet@onera.fr

Gérard Verfaillie
ONERA
Gerard.Verfaillie@onera.fr

Marie-Claire Charmeau
CNES
Marie-Claire.Charmeau@cnes.fr

Abstract

Most of the currently active Earth-observing satellites are entirely controlled from the ground: observation plans are regularly computed on the ground, uploaded to the satellite using visibility windows and then, executed on-board as they stand. Because the possible presence of clouds is the main obstacle to optical observation, meteorological forecasts are taken into account when building these observation plans. However, this does not prevent most of the performed observations to be fruitless because of the unforeseen presence of clouds. To fix this problem, the possibility of equipping Earth-observing satellites with an extra instrument dedicated to the detection of the clouds in front of it, just before observation, is currently considered. But, in such conditions, decision upon the observations to be performed can no longer be made off-line on the ground. It must be performed on-line on-board because it must be performed at the last minute, when detection information is available and because visibility windows between Earth-observing satellites and their control centers are short and rare. In this paper we present a generic architecture made of a reactive decision making task and of a deliberative planning one. Then we show the mechanisms that can be used in both modules for the particular case of an agile Earth-observing satellite.

1. Introduction

1.1. Physical system

Earth-observing satellites are placed on heliosynchronous, low altitude, circular orbits around the Earth. Most of them are equipped with optical observation instruments, with a mass memory able to record observation data, and with a high-rate antenna able to download it towards ground mission centers.

When they are not agile that is keeping pointing to the Earth center, as the current French SPOT satellites do, they can observe specific ground areas thanks to mirrors placed in front of their instruments and mobile along the roll axis.

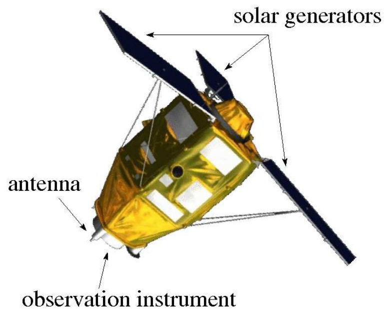
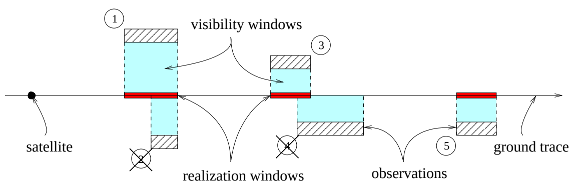


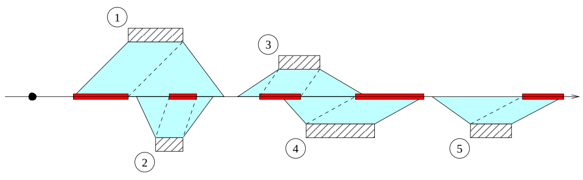
Figure 1. A PLEIADES satellite

When they are agile that is keeping controlling their attitude movement along the roll, pitch, and yaw axes thanks to reaction wheels or to gyroscopic actuators, as the future French PLEIADES satellites will do, this is the whole satellite, which orientates itself to observe specific ground areas (see Figure 1; for more details, see http://smsc.cnes.fr/PLEIADES/).

The main advantage of agility is to offer more freedom in terms of observation. With non-agile satellites, the realization window of an observation is fixed because observation is only possible when the satellite flies over the target ground area. With agile ones, this realization window can be freely chosen within a wide visibility window, because observation is now possible before, during, or after the satellite flies over the target ground area. This freedom may allow more observations to be performed because observations which conflict with each other in the context of non-agile satellites may no longer conflict in the context of agile ones. This is illustrated by Figure 2, which represents five candidate observations, from 1 to 5. For a non-agile satellite (see Figure 2(a)), observations 1 and 2 conflict with each other, because their realization windows overlap. This is also the case with observations 3 and 4. As a result, it is only possible to perform three observations, for example observations 1, 3, and 5. For an agile satellite (see Figure 2(b)), all these conflicts can be resolved and the five observations can be performed.



(a) Non-agile satellite



(b) Agile satellite

Figure 2. Agile Vs non-agile satellite

1.2. Mission objective

The satellite's global objective is to maximize the mission feedback throughout its whole life that is the observations that are performed and delivered to users, taking into account their number and their priorities.

2. Constraints on decision-making mechanisms

2.1. Decision nature

A practical way of approaching the mission objective consists in optimizing the mission feedback each time over a limited temporal horizon ahead, taking into account the available information about the satellite state (orbital position, attitude, available energy and memory levels, etc.), the candidate observations, the meteorological forecasts, and the cloud detection, and in optimizing again each time the current temporal horizon ends or changes occur.

In the particular case of an agile Earth-observing satellite, possible actions are: (1) the observation of any ground area, (2) the download of any observation data by pointing the satellite and thus its antenna to any ground station, (3) the detection of clouds in front of the satellite by pointing the satellite 30 degrees ahead, (4) the recharge of its batteries by orienting its solar generators towards the Sun, (5) a geocentric pointing when it "has nothing to do" or must remain in a safety mode, (6) an orbital maneuver when the drift of its orbital trajectory from the reference orbit becomes too important.

2.2. Global decision

Agile satellites control their attitude movements thanks to reaction wheels or to gyroscopic actuators. An attitude change movement of the considered agile satellite can be decomposed into three parallel movements, each one performed along one satellite axis (roll, pitch, and yaw). Each movement along one axis can be further decomposed into three phases. Figure 3 represents the three successive attitude movement phases along one axis. First phase is an acceleration one with a constant torque T_1. The middle phase is executed with a constant moment L_2. Finally the last phase is a deceleration phase with a constant torque T_3. Computing such a trajectory reaching as fast as possible an action start attitude within a fixed window is itself a difficult constrained continuous optimization problem (see [1] for more details). It looks like a problem of tracking a mobile target from a mobile vehicle.

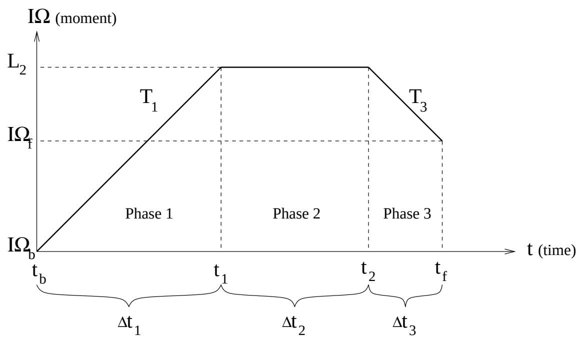


Figure 3. The tree successive attitude movement phases along one axis

For each action to execute, one needs to consider not only the attitude change trajectory necessary to reach the start attitude of the decided action, but also the attitude trajectory the satellite has to follow during the action itself. For example, a battery recharge activity compels the satellite to point its solar generators towards the Sun; a data-downloading activity compels it to point its antenna towards a ground mission center.

As a consequence, the satellite agility links all the satellite activities. The satellite can no longer reason separately on each kind of activity, as it was done in [2] with observation and download activities. But it has to make *global decisions*, reasoning on the whole satellite activity set.

2.3. On-line decision

For almost all the currently active Earth-observing satellites, the decisions are made on the ground under the supervision of human operators in centralized mission centers. Typically, an observation plan is built each day for the next day, taking into account the current set of candidate observations (see [3] for a description of the problem in the context of an agile satellite). Such a plan is uploaded to the satellite using a visibility window and executed as it stands without any execution freedom.

However, optical observation is sensitive to clouds and it is very difficult to foresee the actual presence of clouds the next day over a given ground area. Despite the use of meteorological forecasts by the planning algorithms on the ground, many observations (up to 80%) remain fruitless due to the presence of clouds. To fix this problem, engineers are currently considering the possibility of equipping Earth-observing satellites with an extra instrument dedicated to the detection of the clouds in front of it, just before observation becomes possible [4]. But if one wants to exploit information coming from the detection instrument, it is no longer possible to decide upon observations off-line each day for the next day. Decisions must be made *on-line* at the last minute, when detection information is available.

2.4. On-board decision

Moreover, because Earth-observing satellites are not often within the visibility of their control centers, decisions can no longer be made on the ground. They must be made *on-board* the satellite in order to take advantage of the available cloud detection information.

2.5. Temporally constrained decision time

The agility enables the satellite to perform observation by pointing its observation instrument (and thus the satellite itself) 47 degrees ahead. Moreover, the detection of clouds in front of the satellite is performed by pointing the satellite (and so its detection instrument) 30 degrees ahead. As a consequence, the visibility window of an observation (and thus its realization) can start immediately after (and even a few seconds before) the cloud detection information becomes available. That is why the decision-making mechanisms on-board the satellite should be based on algorithms able to provide it very quickly with a good quality solution.

The different approaches that we considered are tree search, dynamic programming, local search, and greedy search. A tree search approach seems not to be realistic because of the large size of the decision space: for each decided action we have to fix several parameters (the start time and the duration of a geocentric pointing action, the observation data to download for a download action, etc.). A dynamic programming approach should not be convenient because it needs in addition to discretize the satellite state in order to memorize its associated value. However, the satellite state is composed not only of the description of its orbital parameters and resource levels, but also of the description of its current objectives, that is the downloaded data, the performed observations, and the remaining observations. It seems to be difficult to use a local search algorithm because of the attitude constraint: in an activity plan, each activity depends on the end satellite attitude of the previous activity. Adding, deleting or moving an activity in such a plan, as a local search algorithm would do, seems to be difficult and time consuming.

Decision-making mechanisms that we consider are based on iterated stochastic greedy algorithms which build plans choosing randomly a decision with a heuristic bias at each decision time in the solution plan. The preferred decision is the first action of the best built plan (see part 3.2.).

3. Reactive – deliberative architecture

3.1. Description

To tackle this problem of control of such an agile Earth-observing satellite, we use the generic reactive-deliberative architecture defined in [5]. In this generic schema, the reactive tasks are the interface between the environment and the deliberative ones. Figure 4 represents the interactions between the environment, a reactive task, and a deliberative one.

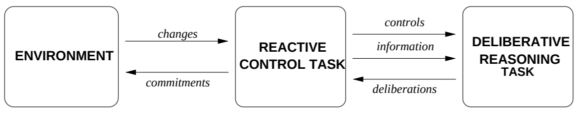


Figure 4. Global schema of the interactions between the environment, a reactive task, and a deliberative one

The reactive tasks enable the system to react to environment stimuli at the rhythm the environment dictates. They have to meet hard temporal constraints and so require only simple computations. The reactive tasks receive information about the way the environment evolves (what is referred to as changes) and emit action commitments following decisions.

The reactive tasks are in charge of the control of the deliberative ones: they trigger, interrupt, trigger again or abort them. They provide deliberative tasks with relevant information about the problem to solve, and receive deliberations resulting from reasoning.

On the contrary, the deliberative tasks need many resources in terms of computation time and memory, which are not often known a priori. Planning and scheduling tasks are typically deliberative.

It is important to stress the difference between this architecture and the classical three-layered control architecture, widely used in the robot control community (see for example [6,7]). In a few words, deliberative planning is the core of a three-layered architecture: nothing can be executed if it has not been planned before. On the contrary, reactive tasks are the core of the reactive-deliberative architecture: deliberative tasks can make proposals, but reactive tasks remain responsible for final decisions and can make decisions even when no proposal is available.

3.2. Deliberative task

The deliberative task of the reactive-deliberative architecture is based on an optimization algorithm. In this work, we consider an iterated stochastic greedy algorithm, which consists in performing a sequence of stochastic greedy searches [8]. Each greedy search performs itself a sequence of decisions in order to build an action plan over a finite planning horizon. In the Earth-observation mission context, this planning horizon may be the next observation set, whose cloud cover has just been detected. Each decision is made by stochastically choosing an action with respect to heuristic bias. This allows the same decisions not to be systematically made search after search.

Such an algorithm is naturally anytime [9]: each greedy search produces a sequence of actions; a sequence of actions is thus available from the first, that is quickly; improvement is achieved each time a greedy search produces a sequence whose associated gain is higher than the best gain produced so far. But such an algorithm offers no guarantee of optimality.

3.3. Implementation using Esterel and Java

We chose to use the synchronous *Esterel* [10] language for programming reactive tasks and the *Java* language for programming the deliberative ones and more generally for acting as a host language, and the so-called *task* mechanism offered by *Esterel* for connecting reactive and deliberative ones.

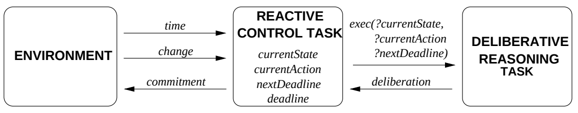


Figure 5. Main functional schema of the interactions between the environment, a reactive task and a deliberative one

Figure 5 represents the main functional schema of the interactions between the environment, a reactive task, and a deliberative one. It shows the main events that are exchanged between tasks (those that appear above arrows) or between modules inside the reactive one (those that appear in the box associated with the reactive task).

The reactive task may receive from the environment a *time* event informing it that time is going on, and a *change* event informing it of a change in the state of the environment. It may send to the environment a *commitment* event informing it of an action commitment following a decision. The reactive task may also trigger the execution of a deliberative task with information about the current state of the environment, the current executing action and the next decision-making time (*exec(?currentState, ?currentAction, ?nextDeadline)*), and may receive from the deliberative task a *deliberation* event informing it of a better available solution. The events exchanged inside the reactive task are *currentState* (informing of the updating of the current state of the environment), *currentAction* (informing of the updating of the current action executed by the satellite), *nextDeadline* (informing of the time of the next deadline), and *deadline* (informing of the occurrence of a deadline).

3.4. Temporal behavior

The temporal behavior of the reactive-deliberative architecture is the following:
The reactive task:

- receives the stimuli from the environment, corresponding to the system state changes and filters only the relevant changes (those that imply to abort and trigger again the deliberative task);
- at each relevant state change
  - computes the next deadline;
  - aborts and triggers again the deliberative task.
- at each deadline, decides on the action commitment taking into account the current system state, and either (1) the deliberations provided by the deliberative task if there is any,

or (2) a default decision computed by the reactive task itself.

The deliberative task:

– executes a reasoning task concurrently with the reactive task which controls it;
– is aborted and triggered again by the reactive task at each relevant state change;
– provides as soon as possible the reactive task with deliberations.

Figure 6 shows a typical execution example. The way time gets on is represented from top to bottom. A *change* event triggers the computation of a deadline, the emission of a *nextDeadline* event, and the triggering of the deliberative task with information about the current state, the current action, and the next deadline. Before the occurrence of the *deadline* event, the deliberative task emits two successive deliberations. Then, the *deadline* event triggers the computation of a decision, the emission of a *commitment* event, the computation of a new deadline, the emission of a new *nextDeadline* event, and again the triggering of the deliberative task.

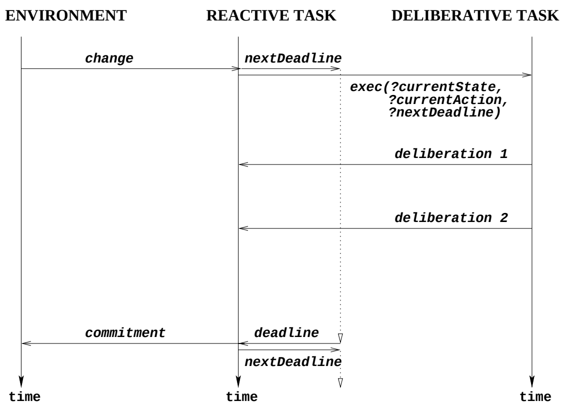


Figure 6. Temporal behavior

4. Simulation environment

4.1. Model

To evaluate the behavior of the reactive-deliberative architecture in the particular case of an Earth-observing mission by an agile satellite, we needed to simulate several scenarios. Then we developed a simulator based on a model of the real world (simulation model) (see Figure 7) comprising:

– a satellite model defined by its orbit (providing the orbital position of the satellite), its platform (essentially the Autonomous Orbit Control and the Attitude and Orbit Control System providing the satellite attitude), and its payload (*i.e.* each satellite instrument behavior);
– a model of the mission centers on the Earth surface;
– a model of the Sun kinematics around the Earth;
– and a goal model describing all the observations to perform and to download (a complex observation includes one or more basic observations which are zones to observe on the ground with several observation conditions; zones are fixed width strips defined by two points on the Earth surface).

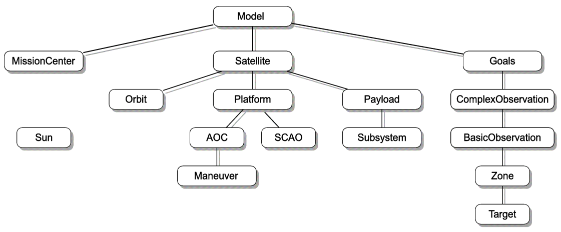


Figure 7. Environment model

On-board the satellite (*i.e.* by the on-board control software), a second environment model (reasoning model) is used which approaches the real world model and which the reactive and deliberative tasks reason on.

4.2. Micro-example

4.2.1. Scenario. We consider an Earth-observing mission by an agile satellite and a simulation horizon of 5 minutes between times 560.0 and 860.0 seconds. At the beginning of the scenario, the satellite's objective is the observation of two zones on the ground (obs 2 and obs 3). For each observation is associated a visibility window during which the satellite can perform it. A ground mission center is visible from the scenario start time until time 650.0 s. The satellite can download performed observations to the ground mission center within this visibility window. At time 565.0 s a new observation request (R(obs 1)) is sent by the ground mission center to the satellite. The Figure 8 summarizes all these elements.

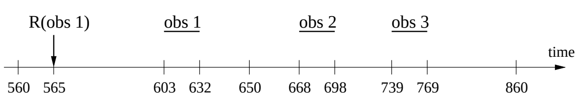


Figure 8. Example of scenario

At start time (time 560.0 s) the satellite state is the following:

- energy level: 5000 Wh
- memory level: 6.0e5 Mb
- orbital position : [-1.08e6; -6.06e6; 3.73e6] m
- orbital speed : [-1.97e3; -3.51e3; -6.26e6] m.s^{-1}
- attitude quaternion : [-0.49; -0.35; -0.35; 0.72]
- attitude speed : [9.83; -2.73; -1.56] m.s^{-1},

and the state of the goals known on-board is:

- observation 2: non performed
- observation 3: non performed.

The satellite current action is a geocentric pointing during still fifteen seconds, and a deliberative task is running to decide on the next action to perform.

The satellite aims at performing and downloading as many observations as possible within the simulation horizon.

4.2.2. Optimization algorithm. In this experiment, we suppose that the actions that can be performed by the satellite are hierarchically ordered (from the more important to the less one):

1. orbital maneuver;
2. download of data to a ground station;
3. observation;
4. detection of the cloud cover in front of the satellite;
5. recharge of the satellite batteries;
6. geocentric pointing.

For the moment, we use a simple decision rule which lists all the possible actions on a fixed planning horizon of 1000.0 seconds, among these six different actions, verifying energy and memory production and consumption. This rule chooses (1) the most important possible action and (2) the potential attitude change needed to reach the start attitude of the chosen decision from the decision-making time attitude. When two actions have the same importance, it chooses the first action which can be performed. In the future, we aim at implementing an iterated stochastic greedy algorithm. To simulate the execution of such an algorithm on more complex problems, we artificially slow down the reasoning process of the decision rule.

4.2.3. Reactive decision. Still for this experiment, the reactive decision computed by the reactive task always consists in performing a geocentric pointing action during 15 seconds in order to let more reasoning time for the deliberative task.

4.2.4. Execution. At the beginning of the scenario, the satellite is waiting for the next deadline (time 575.0 s) keeping a geocentric pointing attitude during 15 seconds.

After several reasoning seconds, the first deliberation is to commit to an attitude change in order to begin observation 2 at the start time of its visibility window.

After the arrival of the new observation 1 request, the deliberative task is triggered again and its first deliberation is to commit to an attitude change in order to begin observation 1 at the start time of its visibility window.

Using the reactive-deliberative architecture on-board the satellite leads thus to the execution described in Table 1: the satellite ends the current geocentric pointing before performing observation 1, downloading the corresponding data, then performing observations 2 and 3, and finishing with a recharge of its batteries thanks to a Sun pointing action until the end of the scenario.

On the contrary, the execution of such a scenario with just an off-line planning at the start time (560.0 s) is represented on the Table 2: the satellite ends the current geocentric pointing and then performs observation 2 and observation 3, using attitude changes to reach the needed observation start attitudes. Then it finishes with a Sun pointing action. It does not care about the new request arrival.

This experiment shows that on-board and on-line planning enables the satellite to perform and to download one more observation (observation 1).

Table 1. Execution with on-line planning

time (s)	Starting action
560.0	geocentric pointing
575.0	attitude change
603.0	observation 1
613.0	attitude change
636.0	obs 1 download
639.0	attitude change
668.0	observation 2
678.0	attitude change
739.0	observation 3
749.0	attitude change
802.0	Sun pointing

Table 2. Execution without on-line planning

time (s)	Starting action
560.0	geocentric pointing
575.0	attitude change
668.0	observation 2
678.0	attitude change
739.0	observation 3
749.0	attitude change
802.0	Sun pointing

5. Conclusion

This reactive-deliberative architecture makes possible the use of on-line decision-making mechanisms on-board an Earth-observing satellite, taking into account the current state of the system.

Executing anytime algorithms in the deliberative task allows the satellite to use at best the reasoning time it has at its disposal between a change occurrence and the next computed deadline: the deliberations get better as the reasoning time goes on.

This architecture provides a clear distinction between the respective roles of the reactive and the deliberative tasks and a clear definition of their interactions. This contrasts with numerous implementations of on-line decision-making systems where reactive and deliberative activities are mixed up into a unique code.

The last advantage of this architecture results from the use of a synchronous language like Esterel whose semantics is well-defined. Provided that it has been checked that the implementation meets properly the synchronous assumption, the execution of the program is not ambiguous. This contrasts with the previous implementation of the example of management of observations by an Earth detection and observation satellite, which used the JADE multi-agent platform (*Java Agent Development framework*) [2] and with which execution anomalies due to time management were frequent.

6. References

[1] G. Beaumet, G. Verfaillie, and M-C. Charmeau, "Estimation of the Minimal Duration of an Attitude Change for an Autonomous Agile Earth-Observing Satellite", *International Conference on Principles and Practice of Constraint Programming, CP'07*, Providence, RI, USA, 2007.

[2] S. Damiani, G. Verfaillie, and M-C. Charmeau, "Cooperating On-board and on the Ground Decision Modules for the Management of an Earth Watching Constellation", *International Symposium on Artificial Intelligence, Robotics, and Automation for Space, i-SAIRAS'05*, Munich, Germany, 2005.

[3] M. Lemaître, G. Verfaillie, F. Jouhaud, J-M. Lachiver, N. Bataille, "Selecting and scheduling observations for agile satellites", *Aerospace Sciences and Technology*, 2002, 367-381.

[4] J-M. Lachiver, J-M. Laherrère, I. Sebbag, N. Bataille, and T. Bret-Dibat, "System Feasability of On-Board Clouds Detection and Observations Scheduling", *International Symposium on Artificial Intelligence, Robotics, and Automation for Space, i-SAIRAS'01*, Montréal, Canada, 2001.

[5] M. Lemaître, and G. Verfaillie, "Interaction between reactive and deliberative tasks for on-line decision-making", *International Conference on Automated Planning and Scheduling, ICAPS'07 Workshop on Planning and Plan Execution for Real-World Systems*, Providence, RI, USA, 2007.

[6] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An Architecture for Autonomy", *The International Journal of Robotics Research*, 1998, 17(4):315-337.

[7] N. Muscettola, P. Nayak, B. Pell, and B. Williams, "Remote Agent: To Boldly Go Where No AI System Has Gone Before", *Artificial Intelligence*, 1998, 103(1-2):5-48.

[8] V. Cicirello, and S. Smith, "Enhancing Stochastic Search Performance by Value-Biased Randomization of Heuristics", *Journal of Heuristics*, 2005, 11(1):5-34.

[9] S. Zilberstein, "Using Anytime Algorithms in Intelligent Systems", *AI Magazine*, 1996, 17(3):73-83.

[10] G. Berry, and G. Gonthier, "The Esterel Synchronous Programming Language: Design, Semantics, Implementation", *Science of Computer Programming*, 1992, 19(2):87-152.