Decision-Making on-board an Autonomous Agile Earth-Observing Satellite

Grégory Beaumet and **Gérard Verfaillie**
ONERA, Toulouse, France
{Gregory.Beaumet, Gerard.Verfaillie}@onera.fr

Marie-Claire Charmeau
CNES, Toulouse, France
Marie-Claire.Charmeau@cnes.fr

Abstract

In this paper, we describe the Earth observation mission we have been working on, including its requirements in terms of autonomy. Then, we describe the decision problem to manage, the reactive/deliberative architecture we used for on-line decision-making, the decision rules used by the reactive part, and the iterated stochastic greedy search used by the deliberative part. Finally, we present results of experiments performed on off-line and on-line realistic scenarios.

The Earth observation mission

In this paper, we present the main results of a study which has been performed in the context of the French CNES-ONERA-LAAS AGATA project (Autonomy Generic Architecture: Test and Applications (Charmeau and Bensana 2008)). The AGATA project uses several scenarios as examples of application of autonomy techniques. One of these scenarios, the one presented in this paper, is a speculative project of improvement of the agile Earth-observing Pleiades satellites which will be launched in 2010: a kind of post-Pleiades project.

Agility means that these satellites are able to move freely around their inertial center along the three axes (roll, pitch, and yaw) thanks to gyroscopic actuators, while moving on their orbit. These satellites are each equipped with a high resolution optical observation instrument (see Figure 1) and observation of ground areas is performed thanks to the combined movement of the satellite on its orbit and on itself (see http://smsc.cnes.fr/PLEIADES/GP_gallery.htm Movies about Pleiades). The way of managing the Pleiades satellites that is currently considered is the following one: observation requests are sent by users to a centralized ground mission center; each day, this mission center builds a precise activity plan for each satellite, taking into account the current requests, their priorities, and the weather forecast; these plans cover all the satellite activities: orbital manoeuvres when they are necessary to maintain the satellite on its reference orbit, observation of ground areas, downloading of observation data to ground stations, pointing of the solar panels towards the Sun in order to recharge batteries, and attitude rendezvous to move from one activity to another one, for example from the end of the observation of a ground area to the beginning of the observation of another one; these plans specify the sequence of activities to be performed by the satellite with their precise temporal positions; they are executed by the satellites without any modification. However, because optical observation is sensible to the presence of clouds and because weather forecast on one day for the next one is imperfect, many observations (around 80% with the current optical Earth-observing satellites) are lost due to the presence of too many clouds.

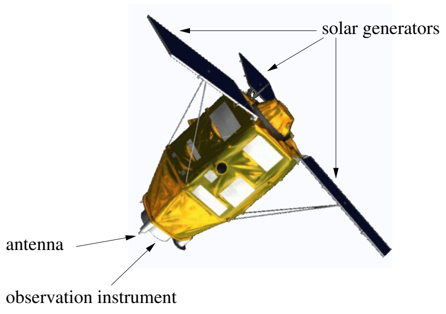


Figure 1: A Pleiades satellite.

To fix this problem, the possibility of equipping each satellite with an instrument dedicated to cloud detection in front of the satellite is currently considered. This is the scenario we studied in the context of the AGATA project and we develop in this paper.

Autonomy requirements

Cloud detection in front of the satellite is possible by pointing the satellite 30° ahead with regard to a geocentric pointing. Because observation is also possible by pointing the satellite until 30° with regard to a geocentric pointing, decision upon observation must be made very quickly after detection if we want to take into account detection information in order to avoid observing cloudy ground areas.

Because Earth-observing satellites are moving on low altitude circular orbits, they are not permanently within the visibility of a ground control station (in fact only about 10% of time). In such conditions, decision upon observation must be made on board.

This led us to consider a completely new way of managing these satellites where the ground mission center sends user observation requests to the satellites and lets them in charge of building their activity plans. In this paper, we present a technical solution of this problem of autonomous decision-making under time pressure, by considering only one satellite, and not several cooperating ones.

The decision problem

The activities the satellite can perform are the following ones: orbital manoeuvre using thrusters, attitude rendezvous using gyroscopic actuators, cloud detection, ground area observation, observation data download, sun pointing to recharge batteries, and geocentric pointing when there is nothing else to do. All these activities require a precise attitude trajectory to be properly performed, except data download which in fact does not require a precise pointing towards the ground station. However, to simplify the decision problem, we consider only data downloads with a precise pointing towards the station. As a result, the satellite is always performing one action and only one, among the seven previously listed. Thus, the decision problem on-board the satellite is to decide each time the current action ends upon the next action to trigger along with its parameters (for example, for a sun pointing its duration). To make such a decision, we must take into account feasibility constraints:

- temporal visibility windows: for example, to start the observation of a given ground area, it must be visible by the satellite; to recharge batteries, the satellite must not be in eclipse; these windows can be easily computed on-board using orbital models;
- satellite attitude conditions: for example, to start the observation of a given ground area, precise conditions must be satisfied in terms of attitude and of attitude speed; moreover, precise attitude trajectories must be followed during the whole observation in order to guarantee that the combination of the movement of the satellite on its orbit and on itself and of the movement of the Earth on itself produce a scan of the target ground area at the specified speed; the computing of these attitude trajectories taking into account orbital models and gyroscopic actuator and satellite attitude models is a difficult task (Beaumet, Verfaillie, and Charmeau 2007); it can be however performed on-board using efficient specialized algorithms;
- resource conditions: for example, to start the observation of a given ground area, enough energy and memory must be available; energy is produced by the solar panels, but this production depends on their orientation with regard to the Sun and thus on the satellite attitude trajectory (solar panels are fixed on the platform); it can be easily computed on-board using solar panel models; memory is consumed by observation and produced by data download.

We must also take into account the gain that is expected from action execution and from the actions that can be performed next. The criteria that we take into account are:

- the priorities of the observations that are performed and downloaded;
- their observation angles: because of the agility, the observation angle of a given ground area is free within some limits; the lower the angle with regard to a geocentric pointing, the better the observation quality;
- the information about the presence of clouds and the date of this information; the less expected clouds and the more recent information (for example, coming from detection), the better the expected observation quality;
- the time at which observation data is downloaded and delivered to users.

The whole decision problem has been modeled using the CNT framework (*Constraint Network on Timelines* (Verfaillie, Pralet, and Lemaître 2008; Pralet and Verfaillie 2008b)). In the CNT framework, a problem is modeled using usual static variables, but also dynamic ones, called timelines, which represent state, event, or time and whose values evolve step by step. The number of steps is itself a variable. The result is a kind of dynamic CSP (Mittal and Falkenhainer 1990). To model the decision problem, we used the following static variables:

- for each observation o, the visibility window i of the associated ground area in which o is performed and its starting and ending times in i, with values 0 when o is not performed;
- for each observation o, the window j in which o is downloaded, with a value 0 when o is not downloaded;
- for each ground station visibility window j, the starting and ending times of data download in j, with values 0 when no download is performed (several observations are usually downloaded in one shot during a ground station visibility window).

We used also the following timelines :

- one timeline to represent the current time;
- three timelines to represent the memory, energy and ergol levels, currently available;
- one composite timeline (in fact seven timelines) to represent the current satellite attitude and attitude speed;
- for each observation o, one timeline to represent the last time at which information about the cloud cover over the associated ground area has been delivered by a ground station or by the cloud detection instrument;
- one timeline to represent the satellite current action among the seven possible actions (orbital manoeuvre, attitude rendezvous ...);
- one timeline to represent the observation currently performed in case of an observation action, with a value 0 when the current action is not an observation;
- one timeline to represent the station towards data are downloaded in case of a data download action, with a value 0 when the current action is not a download;
- one timeline to represent the duration which has been decided for the current action in case of an attitude rendezvous action or of a sun or geocentric pointing action, with a value 0 when the current action is not one of these three (the duration of an orbital manoeuvre is imposed, the duration of a detection is fixed, the duration of an observation action depends on the chosen observation, and the duration of a download action depends on the choices made for static variables);

- one composite timeline to represent the target attitude and attitude speed in case of an attitude rendezvous action, with a value 0 when the current action is not a rendezvous;
- finally, one timeline to represent the precise orbital manoeuvre in case of an orbital manoeuvre action, with a value 0 when the current action is not a manoeuvre.

Considering that data downloads can be performed concurrently with other actions would be possible, because concurrent timelines can be modeled in the CNT framework, but would result in a globally more complex model.

A reactive/deliberative architecture for decision making

On-board conditions may change at any time because of the arrival of new observation requests or of new weather forecast sent by the ground mission center, because of new information about the presence of clouds provided by on-board detection, or because of unexpected levels of energy or memory. In such a quickly changing environment, this would not be a good idea to build an activity plan over a given planning horizon, to execute it and, just before the end of the planning horizon, to build a new plan over the next planning horizon, as suggested in (Muscettola et al. 1998). A more sensible option would be to build an activity plan over a given planning horizon and to execute it until it remains valid, as suggested by many authors and experimented in many applications (Chien et al. 2000). The main drawback of such an approach is that it is often difficult to assess the validity of a plan in a changing environment, especially when we are looking not only for a feasible plan, but for an optimal (or as good as possible) one. This is why we adopted a classical approach in sequential decision-making: at the end of each action, to decide upon the best action to perform next with regard to the conditions known at the decision time.

If we adopt this latter approach, the problem is now the way of making such a decision. The easiest way of doing that consists in using predefined decision rules. However, it is known that, due to their local view, even sophisticated decision rules may produce bad decisions in complex situations. Another way, used for example in the MDP framework (Puterman 1994), consists in computing off-line an optimal (or approximate) policy which associates an action with each possible situation. However, this is often impracticable due to the huge number of possible situations in realistic applications such as the one we face, in spite of the numerous techniques that have been proposed to overcome this difficulty (Bertsekas and Tsitsiklis 1996). The way we chose consists in coming back to planning, by using powerful approximate anytime planning algorithms to build activity plans from the current situation over a given planning horizon ahead and by using the first action of the best plan found as the decision to make.

In fact, it is possible to combine decision rules and planning algorithms. This is what we did with the reactive/deliberative architecture we adopted (Lemaître and Verfaillie 2007). In this architecture, which has interesting similarities with the IDEA architecture (Muscettola et al. 2002), the control of a system is divided into at least two tasks: a reactive one and a deliberative one (see Figure 2). The reactive control task is directly interfacing the environment from which it receives changes at which it reacts instantaneously possibly via action commitments. On the other hand, the reactive control task controls one or several deliberative reasoning tasks which are assumed to have an anytime behaviour and to be able to produce deliberations which can be seen as action proposals. More precisely, when changes occur, the reactive task computes a deadline for the next decision-making, which is most of the time the end of the current action, and triggers the deliberative task by providing it with information about the new situation and the deadline, everything instantaneously. After that, the deliberative reasoning task produces successive deliberations, each one assumed to be better than the previous one. When the deadline occurs, the reactive control task uses the last deliberation for decision-making. If no deliberation is available or if the last deliberation is inconsistent with the current situation (for example, if some safety constraints are violated), it uses a decision rule to make instantaneously a consistent decision. In such an architecture, the reactive control task is clearly the decision-maker and the deliberative reasoning one acts as an adviser. Because most of the actions last some tens of seconds, time is usually available during action execution to reason about the action to perform next. Anyway, when not enough time is available, the reactive decision rule applies.

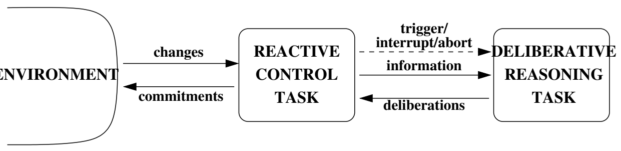


Figure 2: Global schema of the interactions between the environment, a reactive control task and a deliberative reasoning one.

We implemented this architecture by using the Esterel synchronous language (Berry and Gonthier 1992) to implement the reactive control task, the Java language to implement the deliberative reasoning task and to serve as a host language for Esterel, and the Esterel *Task* mechanism to control the deliberative reasoning task from the reactive control one. At the end, everything is compiled into a unique Java program which implements the whole control loop.

Decision rules

When designing decision rules for planning and scheduling problems, the criteria that are classically considered are: (1) the activity priorities, (2) their start time and duration, and (3) the amount of resource they consume. One wants indeed to favour activities which have the highest priority, which start as early as possible in order not to let the system idle for a too long time, and whose duration and resource consumption are as small as possible.

After some tries and experiments, we chose the following decision rule for the reactive control task. Let the seven possible activities, except attitude rendezvous, be ordered along the following decreasing priority order: (6) orbital manoeuvre, (5) observation data download, (4) cloud detection, (3) ground area observation, (2) sun pointing, and (1) geocentric pointing. Let H_d be a short decision horizon ahead the decision time. The rule consists in selecting the highest priority activity among all the ones that can be executed within H_d. Moreover, when a ground area observation is selected, the rule consists in selecting the highest priority observation, with a random tie-break. When a sun or geocentric pointing is selected, an arbitrary duration is chosen. An activity a can be executed within H_d if and only if it is possible to perform an attitude rendezvous from the current satellite attitude to the attitude required at the beginning of a, to execute a, and to end it within H_d without violating any resource constraint.

If H_d is too small, only the activities that can start and end very early are considered and high priority activities may be ignored. On the contrary, if H_d is too large, the rule may select high priority activities that start and end very late, let the satellite perform long attitude rendezvous to reach the required attitudes, and ignore lower priority activities that could have been scheduled without any problem before the higher priority ones. To set the size of H_d, we perform experiments and the best value we found for this application is 100 seconds.

An iterated stochastic greedy search algorithm

To go beyond the local view of decision rules, a sensible approach consists in searching for good activity plans from the current situation over a planning horizon H_p ahead, with H_p far larger than H_d, and in using the first action of the best plan found as the decision to make. Systematically, we set H_p such that it goes until the end of the next eclipse period k (at most 100 minutes away) in order to be sure that the satellite will be able to go out of k with enough energy.

To compare two activity plans, we use a two level hierarchical criterion. The first level criterion is an evaluation of the gain that will be obtained at the end of the plan thanks to the execution of the planned activities. This evaluation is itself a combination of all the criteria listed in the section "The decision problem". The second level criterion is an evaluation of the gain that could be obtained after the end of the plan, taking into account the state of the system at that time. Because this evaluation over an infinite horizon with a lot of uncertainties about the future observation requests and weather conditions is very difficult, the only criterion we consider is the level of energy, in order to favour plans that let the satellite with a high level of energy.

To search for good activity plans, we have the choice between several search mechanisms, mainly tree search, dynamic programming, local search, and greedy search. The use of a restricted form of tree search has been explored for similar applications (Khatib et al. 2003). However, because of the huge number of alternatives to consider (choice of the activity type and of its parameters: for example, the target ground area in case of observation or the duration in case of a sun pointing), systematic tree search does not seem to be a good candidate. Moreover, depth or best first searches are not known to have a good anytime behaviour, a very important feature in this application. Dynamic programming may improve on tree search thanks to its caching mechanism. It has been successfully used on a far simpler problem: the building of observation plans for a non agile satellite (Damiani, Verfaillie, and Charmeau 2004). However, it requires a memory linear in the size of the state space which prevented us from using it in this application. Local search has been successfully applied in similar applications (Chien et al. 2004). However, implementing local moves would be difficult in this application. Indeed, let us assume that we want to replace an activity c by an activity d between activities a and b in the current plan. This would require to check whether or not activity d can follow activity a, but also whether or not activity b and all the following ones in the current plan can follow activity d i.e., to compute a completely new satellite attitude trajectory from the end of activity a: something which is computationally costly and may in addition lead to the removal of activities that were previously planned after a, but may be no longer possible due to the new attitude trajectory. This complexity of a local move would limit the number of moves and thus the power of local search. Greedy search is a simple powerful mechanism when heuristic and random choices are combined as in (Bresina 1996; Pralet and Verfaillie 2008a). Moreover it allows activity plans to be built in an incremental way forwards from the current state: something which simplifies the computing of the satellite attitude trajectory and of the resulting energy production. As a consequence, the search algorithm we implemented is an iterated heuristic-biased stochastic greedy search.

More precisely, the heuristic-biased stochastic greedy search we implemented is a stochastic version of the decision rule presented in the previous section. Roughly speaking, at the decision time, the highest priority activity among all the ones that can be executed within the decision horizon H_d is selected with a probability p. With a probability $1-p$, a lower priority activity will be selected. The stochastic decision diagram used is shown in Figure 3. After decision-making, the end of the selected activity becomes the new decision time and the selection process is repeated until the end of the planning horizon H_p is reached. The quality of the resulting plan is evaluated and compared with the quality of the best plan found so far. Then, the planning process is started again from the initial decision time until it is interrupted by the reactive control task. Obviously, the setting of parameters such as p is crucial in such an algorithm. We perform systematic experiments on realistic scenarios with various parameter settings to determine a good setting. For example, we set $p = 0.9$.

For the moment, the algorithm we implemented is very simple. The first search strictly applies the decision rules. The second one follows the best plan previously found (before the deliberative task was triggered), from which the first action has been removed when it has been triggered. Then, the following stochastic searches try and improve on these first two searches. However, the algorithm involves nor look-ahead (to be able to stop the search when it is clear that no plan better than the best found so far can result from

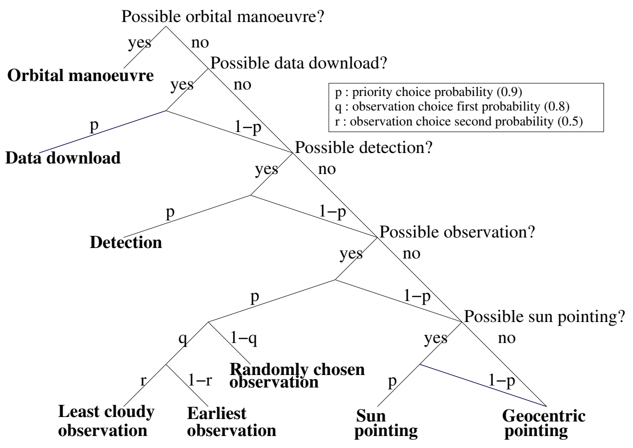


Figure 3: Stochastic decision diagram.

the current search), neither learning (to be able to learn high quality sub-plans from previous searches, as done with Ant Colony optimization (Dorigo, Maniezzo, and Colorni 1996)) capability. So, it could be improved along several directions.

Experimental framework

To perform experiments, we built a complete realistic framework which integrates models of the movement of the relevant bodies (satellite, Earth, and Sun), of the platform (gyroscopic actuators, solar panels, batteries), of the payload (detection instrument, observation instrument, mass memory, data download antenna), of the ground stations, of the cloud cover (actual and forecast), and of the user observation requests.

This framework allows us to build scenarios and to perform off-line and on-line experiments with various algorithms and various configurations of these algorithms. Figure 5 shows the graphical interface of this framework in the context of an on-line real-time simulation, where we can observe how evolve the state of the satellite and of its environment (as it is actually and as it is viewed by the satellite software), the current action, the current deadline (at the end of the current action), the currently best plan found so far (from the current deadline ahead), and the gain accumulated by execution.

Off-line experiments

We first carried out off-line experiments which aim at evaluating the ability of the iterated heuristic-biased stochastic greedy search to produce quickly activity plans whose quality is higher than the quality of the plan which would result from a simple greedy search which would apply strictly the decision rule.

Figure 4 shows typical results over a planning horizon H_p of 60 minutes, with 30 candidate observations. The iterated heuristic-biased stochastic greedy search has been run 10 times, each time with a time limit of 100 seconds. A dot is plotted each time an iterated heuristic-biased stochastic greedy search produces a plan whose quality is better than the one of the best plan found so far.

We observe that the stochastic greedy search takes only some seconds to perform better than the strict greedy search. Such a result is clearly compatible with the time available for reasoning in this application: typically some tens of seconds.

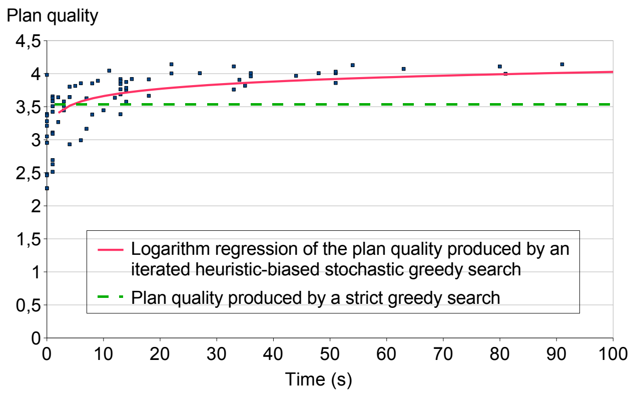


Figure 4: Comparison between an iterated heuristic-biased stochastic greedy search and a strict greedy search.

On-line experiments

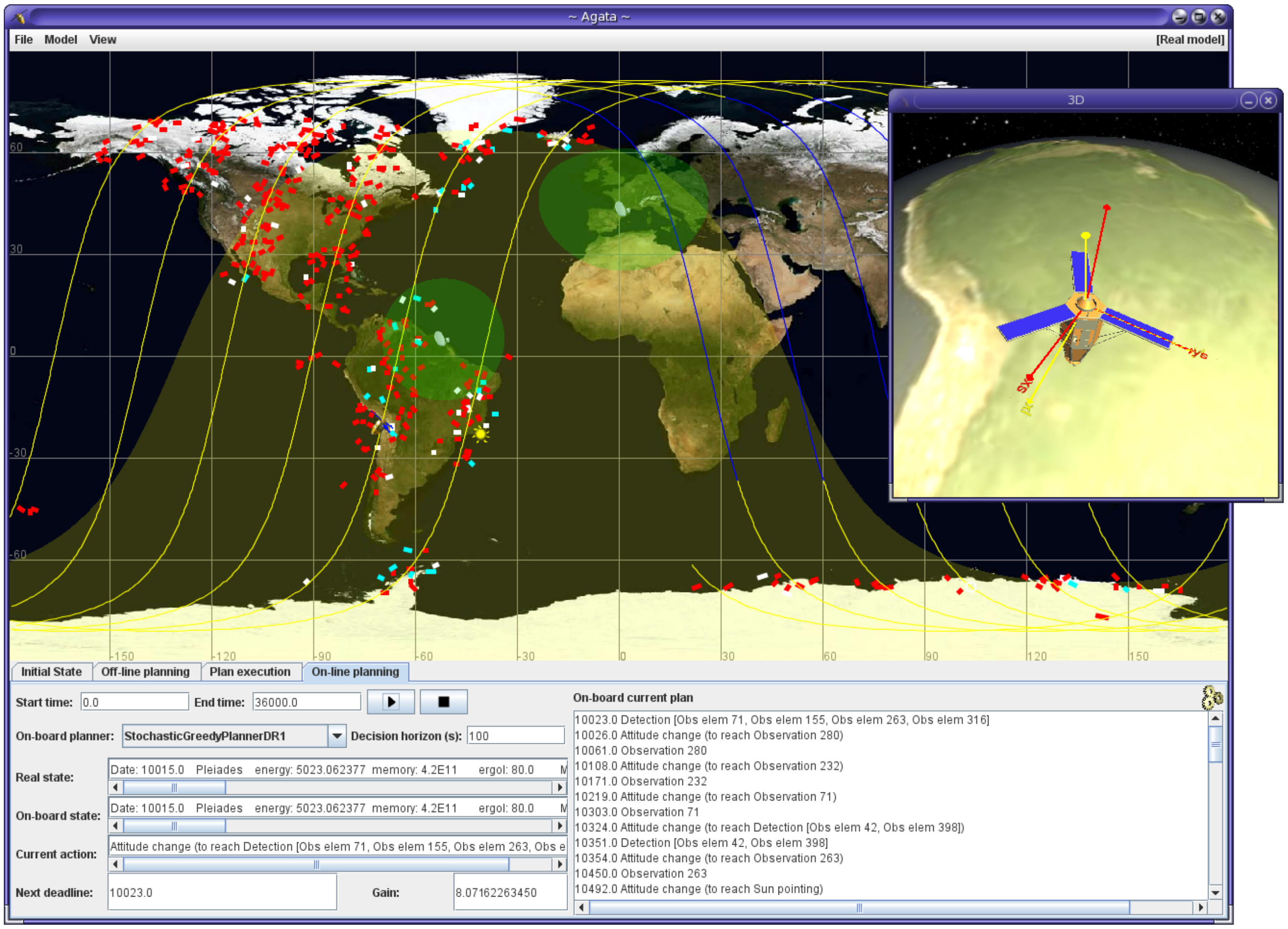
Then, we carried out on-line real-time experiments with the reactive/deliberative architecture, with the decision rule used by the reactive control task, and with the iterated heuristic-biased stochastic greedy search used by the deliberative reasoning one. In such a context, what is evaluated is not the speculative gain computed by the planning algorithm when it is activated, but the actual gain resulting from the execution of the actions to which the reactive control task commits, using advice from the deliberative reasoning task.

Table 1 shows the results of simulations over a horizon of one hour, with 60 candidate observations, with and without the deliberative task. no is the number of performed observations, no_1 (resp. no_2 and no_3) is the number of performed observations of priority 1 (resp. 2 and 3, with 3 being the highest priority level), no_{fc} (resp. no_{mc}) is the number of performed observations such that the actual cloud cover is smaller (resp. higher) than 50%, mc is the mean cloud cover over all the performed observations, and g is the gain resulting from execution.

We observe that the results are systematically better, according to all the criteria, with the deliberative task acting as an adviser than without it.

Table 2 shows the results of simulations performed on the same scenario, with and without detection, but in both cases with the deliberative task.

We observe the positive impact of detection: detection takes time and thus allows less observations to be performed; but the performed observations are less cloudy, resulting in a smaller mean cloud cover and globally in a higher gain.



Start time: 0.0 End time: 36000.0

On-board planner: StochasticGreedyPlannerDR1 Decision horizon (s): 100

Real state: Date: 10015.0 Pleiades energy: 5023.062377 memory: 4.2E11 ergol: 80.0

On-board state: Date: 10015.0 Pleiades energy: 5023.062377 memory: 4.2E11 ergol: 80.0

Current action: Attitude change (to reach Detection [Obs elem 71, Obs elem 155, Obs elem 263, Obs e

Next deadline: 10023.0 Gain: 8.07162263450

On-board current plan

```
10023.0 Detection [Obs elem 71, Obs elem 155, Obs elem 263, Obs elem 316]
10026.0 Attitude change (to reach Observation 280)
10061.0 Observation 280
10108.0 Attitude change (to reach Observation 232)
10171.0 Observation 232
10219.0 Attitude change (to reach Observation 71)
10303.0 Observation 71
10324.0 Attitude change (to reach Detection [Obs elem 42, Obs elem 398])
10351.0 Detection [Obs elem 42, Obs elem 398]
10354.0 Attitude change (to reach Observation 263)
10450.0 Observation 263
10492.0 Attitude change (to reach Sun pointing)
```

Figure 5: Graphical interface of the experimental framework.

These experiments have been performed using a Sun station whose memory and computing power are far higher than the ones of computers currently embedded in Earth-observing satellites. First, we can hope that the memory and computing power available on board satellites will increase significantly in the next years. Second, we must emphasize that an iterated stochastic greedy search consumes a very small amount of memory, because it only needs to record the best plan found so far and the one currently built. As for the computing time, we observed that the mean CPU time to perform one stochastic greedy search over a planning horizon H_p of 100 minutes is of about 243 milliseconds, which is very small with regard to the time usually available for reasoning, which is of some tens of seconds. For example, situations where the deliberative task does not have enough time to produce any plan are very rare. They are in fact limited to the ones where changes from the environment, on the one hand, and a deadline, on the other hand, occur at the same time. Moreover, we must stress that it is always possible to adapt the size of the planning horizon H_p to the available computing power. Nevertheless, we plan to perform experiments in which the deliberative task is artificially slowed down in order to simulate and evaluate the impact of a smaller computing power.

	without the deliberative task	with the deliberative task
no	10	15
no_1	1	3
no_2	1	3
no_3	8	9
no_{fc}	6	12
no_{mc}	4	3
mc	48%	34%
g	2.70	4.98

Table 1: Results of on-line real-time simulations with and without the deliberative task

	without detection	with detection
no	18	15
no_1	3	3
no_2	4	3
no_3	11	9
no_{fc}	9	12
no_{mc}	9	3
mc	55%	34%
g	2.32	4.98

Table 2: Results of on-line real-time simulations with and without detection

Discussion

This work shows how reactive control and deliberative reasoning can be integrated into a reactive/deliberative architecture for the high level mission management of an autonomous Earth-observing satellite. It shows how "intelligent" search mechanisms can be controlled in a real-time context, can propose better decisions to the satellite reactive control, and may result in a globally better satellite behaviour and finally in higher user satisfaction.

References

[Beaumet, Verfaillie, and Charmeau 2007] Beaumet, G.; Verfaillie, G.; and Charmeau, M. 2007. Estimation of the Minimal Duration of an Attitude Change for an Autonomous Agile Earth-observing Satellite. In *Proc. of the 13th International Conference on Principles and Practice of Constraint Programming (CP-07)*.

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

[Bertsekas and Tsitsiklis 1996] Bertsekas, D., and Tsitsiklis, J. 1996. *Neuro Dynamic Programming*. Athena Scientific.

[Bresina 1996] Bresina, J. 1996. Heuristic-Biased Stochastic Sampling. In *Proc. of the 13th National Conference on Artificial Intelligence (AAAI-96)*, 271–278.

[Charmeau and Bensana 2008] Charmeau, M.-C., and Bensana, E. 2008. Testing Spacecraft Autonomy with AGATA. In *Proc. of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-08)*.

[Chien et al. 2000] Chien, S.; Knight, R.; Stechert, A.; R.Sherwood; and Rabideau, G. 2000. Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling. In *Proc. of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-00)*.

[Chien et al. 2004] Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davies, A.; Lee, R.; Mandl, D.; Frye, S.; Trout, B.; Hengemihle, J.; D'Agostino, J.; Shulman, S.; Ungar, S.; Brakke, T.; Boyer, D.; VanGaasbeck, J.; Greeley, R.; Doggett, T.; Baker, V.; Dohm, J.; and Ip, F. 2004. The EO-1 Autonomous Science Agent. In *Proc. of the 3rd Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*.

[Damiani, Verfaillie, and Charmeau 2004] Damiani, S.; Verfaillie, G.; and Charmeau, M.-C. 2004. An Anytime Planning Approach for the Management of an Earth Watching Satellite. In *Proc. of the 4th International Workshop on Planning and Scheduling for Space (IWPSS-04)*.

[Dorigo, Maniezzo, and Colorni 1996] Dorigo, M.; Maniezzo, V.; and Colorni, A. 1996. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics* 26(1):29–41.

[Khatib et al. 2003] Khatib, L.; Frank, J.; Smith, D.; Morris, R.; and Dungan, J. 2003. Interleaved Observation Execution and Rescheduling on Earth Observing Systems. In *Proc. of the ICAPS-03 Workshop on "Plan Execution"*.

[Lemaître and Verfaillie 2007] Lemaître, M., and Verfaillie, G. 2007. Interaction between reactive and deliberative tasks for on-line decision-making. In *Proc. of the ICAPS-07 Worshop on "Planning and Plan Execution for Real-world Systems"*.

[Mittal and Falkenhainer 1990] Mittal, S., and Falkenhainer, B. 1990. Dynamic Constraint Satisfaction Problems. In *Proc. of the 8th National Conference on Artificial Intelligence (AAAI-90)*, 25–32.

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

[Muscettola et al. 2002] Muscettola, N.; Dorais, G.; Fry, C.; Levinson, R.; and Plaunt, C. 2002. IDEA: Planning at the Core of Autonomous Reactive Agents. In *Proc. of the 3rd NASA International Workshop on Planning and Scheduling for Space*.

[Pralet and Verfaillie 2008a] Pralet, C., and Verfaillie, G. 2008a. Decision upon Observations and Data Downloads by an Autonomous Earth Surveillance Satellite. In *Proc. of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-08)*.

[Pralet and Verfaillie 2008b] Pralet, C., and Verfaillie, G. 2008b. Using Constraint Networks on Timelines to Model and Solve Planning and Scheduling Problems. In *Proc. of the 18th International Conference on Artificial Intelligence Planning and Scheduling (ICAPS-08)*.

[Puterman 1994] Puterman, M. 1994. *Markov Decision Processes, Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

[Verfaillie, Pralet, and Lemaître 2008] Verfaillie, G.; Pralet, C.; and Lemaître, M. 2008. Constraint-based Modeling of Discrete Event Dynamic Systems. *Journal of Intelligent Manufacturing, Special Issue on "Planning, Scheduling, and Constraint Satisfaction"*. To appear.