Université Fédérale

Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace

Présentée et soutenue par :
Adrien MAILLARD

le lundi 9 novembre 2015

Titre :

Production au sol de plans flexibles pour des satellites agiles d'observation de la terre

Flexible Scheduling for Agile Earth Observing Satellites

École doctorale et discipline ou spécialité :

EDSYS : Systèmes embarqués et Robotique

Unité de recherche :

Équipe d'accueil ISAE-ONERA CSDV

Directeur(s) de Thèse :

M. Gérard VERFAILLIE (directeur de thèse)
M. Cédric PRALET (co-directeur de thèse)

Jury :

M. Gilles TROMBETTONI Université de Montpellier, France - Président
M. Roman BARTAK Charles University, Czech Republic - Rapporteur
M. Frédéric FONTANARI Airbus D&S, France
M. Jean JAUBERT CNES, France
M. Derek LONG King's College, United Kingdom
M. Cédric PRALET ONERA, France - Co-directeur de thèse
Mme Christine SOLNON INSA Lyon, France - Rapporteur
M. Gérard VERFAILLIE ONERA, France - Directeur de thèse

Remerciements

> « On a à peine vu clair que c'est déjà la fin. Batailler, s'imaginer qu'on va bouleverser le monde pour avoir torché quelques milliers de pages et raconté, et décanté sa petite tranche de vie en long et en large. La belle affaire ! Contente-toi de manger ta soupe en regardant les étoiles. Toujours semblables à elles-mêmes dans le soir azuré. Depuis le vieil Adam. Et avant le vieil Adam. Et avant ce qui était avant qu'il n'y eût rien. Splendides et immuables, nos petites frangines les étoiles. Ont présidé ta naissance. Présideront à ta mort. T'ont vu vagissant dans les langes, laid comme un ouistiti. Te verront chenu, planté sur deux cannes, cadavérique, figé, couleur de suif, empaqueté dans ta caisse, aspergé d'eau bénite. »
> Louis Calaferte, Septentrion.

J'aimerais rendre hommage aux personnes que j'ai pu croiser pendant ces trois années. D'abord, je remercie chaleureusement Gérard et Cédric qui ont eu la difficile tache d'encadrer ce travail à l'ONERA et de me gérer au quotidien. Vous avez été de vrais modèles, à la fois doués et modestes, comme on s'attend à avoir en thèse.

Merci à Thierry Desmousceaux qui a supervisé ce travail pour Airbus lors de la première année et demi. Frédéric Fontanari a pris la relève et a tout de suite contribué à notre travail. Merci à Jean Jaubert qui a suivi mon travail pour le CNES avec bienveillance et qui m'a beaucoup aidé.

Merci à Marie-Jo Huguet et Thierry Vidal d'avoir suivi mes travaux au travers de deux fructueux comités de thèse.

Merci à Christine Solnon et Roman Bartak d'avoir accepté de rapporter cette thèse et d'avoir contribué à la réflexion sur mon travail.

Je remercie tous les ingénieurs de l'unité Conduite et Décision de m'avoir accueilli dans de bonnes conditions. Merci à Françoise, Serge, et Valérie d'avoir été patients lorsqu'il s'agissait des missions et de l'organisation de la soutenance. Merci à Pascale Snini et Joelle Guinle pour ces aspect, au CNES.

Mais il a bien fallu se détendre un peu après le travail. Je remercie tous les membres de la section théâtre de l'ONERA. J'ai beaucoup aimé jouer avec vous dans *Les Justes* de Camus et dans *L'équarrissage pour tous* de Vian. "La révolution, bien sûr! mais la révolution pour la vie, tu comprends?", vous comprenez ?

Merci à tous les doctorants du département sans qui la vie au laboratoire n'aurait pas été aussi animée. Merci à Sergio, je souhaite vivement que notre club de cinéma franco-italien perdure dans le temps; Pierre et Simon, combien d'heures avons-nous passé en pause à discuter joyeusement; Henri, nos discussions et débats politiques resteront un temps fort de ma thèse, à bientôt en France, en Hongrie, à Cuba, ou bien en Équateur; Jérémy, je garde un excellent souvenir de notre déprime rédactionnelle; Nicolas, tes barbecues ont certainement contribué à la bonne humeur générale quand l'ONERA était vide en été, continue à être swag et à m'envoyer du rap à l'ancienne; Jacques, je sais qui contacter lorsqu'il sera l'Heure, fais attention à ne pas sauter en attendant; Patrick, merci pour l'animation, les jeux, les points de vue contradictoires dans les débats; Francis, je ne t'ai pas beaucoup vu ces derniers mois mais je te souhaite une bonne continuation; Emmanuel, hasta la victoria siempre, merci pour le cake aux fruits, continue tes marches constructives ça a l'air de fonctionner, à très bientôt; Alvaro, comment vais-je faire pour choisir les films que je ne vais pas voir ? Igor, toujours fourré dans un livre de mathématiques; Mathieu, je sais qui contacter si je veux construire un avion un de ces jours; Jorrit, a bientôt pour une pause et une promenade

avec ton pote à 4 pattes; Hélène, passionnée par le spatial, tu m'as aussi montré que le sport est plein de politique;

Merci à Rémi et Maxime, mes deux colocataires qui me supportent depuis pratiquement 4 ans. Rémi, je me souviens de nos trajets quotidiens à vélo, à toute allure, le long du canal, à en perdre le souffle. Allez, remplissons les sacoches et partons sur les chemins. Maxime, que de bons souvenirs dans les dunes illuminées du coucher de soleil marocain, j'espère qu'on aura l'occasion de repartir. Et qu'un jour j'entendrai ton duo avec Hélène. Souvenez-vous, tous les deux, des repas du dimanche, de cette horloge atroce que vous avez lâchement acheté dans mon dos, des réveillons festifs (et de la voisine), des mornes dimanches qu'il fallait occuper, des lacs italiens au petit matin... Sachez que grâce à vous, je garde un souvenir très agréable de ces années. Tout cela s'en ira bien trop vite.

Des remerciements spéciaux vont à ma famille qui m'a toujours soutenu moralement et encouragé à poursuivre mes études.

Un dernier remerciement, particulièrement affectueux, à toi. Par chance, je t'ai rencontré ici à l'ONERA, il y a quelques mois seulement et j'espère continuer ma route avec toi encore longtemps. *Nous étions faits pour être libres\Nous étions faits pour être heureux*.

Pour terminer ces remerciements et comme je sais que c'est souvent la seule partie lue dans une thèse, je veux maintenant rappeler les mots d'un homme qui a pensé le *système technicien* et qui a beaucoup à apporter aux techniciens que nous sommes.

> «De plus en plus des techniciens prétendent formuler des problèmes de la société comme des problèmes exacts et en des termes qui permettent une solution. Le mythe croissant de la solution, évacue progressivement de nos consciences le sens du relatif, c'est-à-dire de l'humilité du politique vrai.»
>
> Jacques Ellul, L'illusion politique.

Pour que nous n'oubliions jamais que derrière les modèles, les équations, et les algorithmes, il y a souvent des hommes. Dans cette thèse nous modélisons un problème d'optimisation sous contraintes. J'encourage vivement mes contemporains à ne pas considérer tous les aspects de la vie comme un tel problème, à la manière de certains économistes et scientifiques, et à rejeter l'idée d'une soi-disante neutralité axiologique de la science.

Comme disait Kessel, *je voulais tant dire et j'ai dit si peu*, merci à tous.

Adrien Maillard
Novembre 2015, Toulouse

Contents

Introduction 1

"Tu sais, je voudrais ne jamais descendre."

—Jean Mermoz

Earth observation from space is useful in many domains such as meteorology, geodesy, climate modelling, natural disaster management, or military reconnaissance. It allows us to better understand natural phenomenas such as marine currents, to prevent or follow natural disasters, to follow climate evolution, and many other things. The space system we consider is made of Earth observation satellites orbiting around the Earth. The latter realize acquisition of images for civil or military users. They are equipped with high-resolution optical instruments and communicate with a large network of ground stations. They acquire data, compress and record it on board, and then download it to the ground. Their usage for any acquisition is a complex process. In this process, users first submit observation requests to mission centres. The latter build activity plans which are sent to the satellites. These plans contain several types of actions such as orbital maneuvers, acquisition realizations, and acquisition downloads.

A key issue when scheduling activities of observation satellites is that they evolve in a dynamic environment where unexpected events occur, such as meteorological changes or new urgent observation requests. Several parameters are uncertain when planning on the ground such as cloud cover or available power on board. Until now, plans are not to be modified on board. It means that in face of uncertainty, current plans need to be robust. It implies that on one hand, all the decisions are made offline on the ground and the satellite is a simple executive which neither makes, nor changes any decision, and on the other hand, worst-case assumptions are made about uncertain parameters. This makes planning very pessimistic and plans suboptimal. These uncertainties make planning and scheduling satellites activities offline on the ground more and more arguable. The objective of this work is to improve performances by giving more autonomy to the satellite for decision-making without compromising the predictability that is needed for some activities. The main idea is to share decision-making between ground and board to take advantage of the high computing power on the ground and of the low uncertainty on board. First we apply this idea to download scheduling which consists in scheduling file downloads during ground station visibility windows. Second, we apply this idea to observation planning.

Industrial context: the OTOS program

During 2011 and 2012, two agile Earth-observing satellites, Pléiades-1A and Pléiades-1B, were launched by Soyuz rockets at the Guiana Space Centre. It is the result of a French-Italian cooperation program under management of the Centre National d'Etudes Spatiales (CNES, the French space agency). Many European countries contribute to this program such as Sweden (onboard computers) or Spain (S-band antenna).

OTOS is a technological program whose aim is to increase the performance of such systems and to decrease usage costs. Parameters to improve include image resolution (the number of meters per pixel on an image, the current resolution can be as

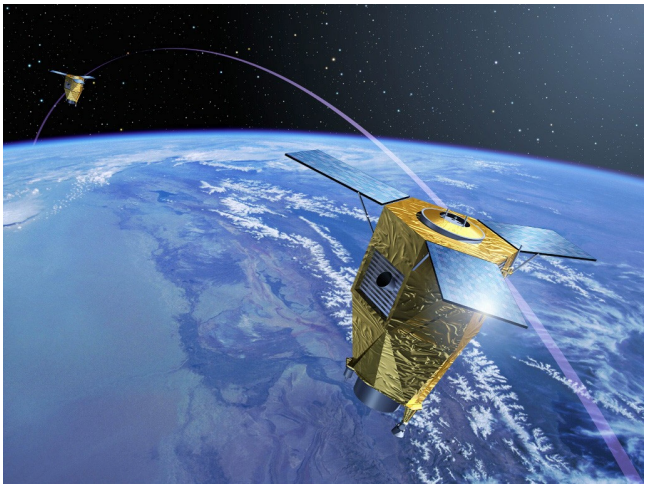


Figure 1.1: Artist view of a Pléiades satellite orbiting around Earth (CNES, 2003).

high as 20cm per pixel), altimetric precision when measuring reliefs, and the number of spectral bands (number of frequencies at which the instrument is able to collect data). Also, an improvement is wanted in mission-related parameters such as *reactivity* (delay between an observation request and its realization) which is important with regard to situations such as forest fires or floods, and *visiting frequency* over the same geographical zone; which can be useful for zone surveillance.

Contributions

Data download problem

Our first contribution concerns the data download problem which consists in scheduling data downloads during communication windows with ground stations. Sophisticated onboard algorithms compress low-interest zones of acquired images such as cloudy zones or low-interest zones such as uniform terrain. Then, the amount of data that results from an acquisition and thus is recorded on board and must be downloaded to the ground is more and more unpredictable. It depends on the data that has been acquired.

As satellites are not continuously accessible by a ground control station, generated volumes are known first on board before being communicated to the ground. This leads to think that decisions about downloads should be made on board just before any reception station visibility window with the exact knowledge of the volumes of the already recorded data. However, planning onboard is time-consuming and requires computing resources. Moreover, the resulting plans are unpredictable. This is problematic especially for high-priority acquisitions, because users who request an image may want to know when data will be downloaded. In such conditions, it is still possible to build data download plans on the ground with the assumption of maximum volumes (minimum default compression rate). The resulting plans are

then always executable on board, but suboptimal due to an under-use of the station visibility windows.

In this work, we introduce more decision-making autonomy on board satellites while keeping some predictability for operators on the ground. The data download problem is a complex scheduling problem with temporal and non-temporal constraints, close to *RCPSP/max* or *flexible job-shop* problems. However, no existing technique can be directly applied to this problem because of specific features such as time-dependent temporal constraints or the objective of resource fair sharing in the optimization criterion. To solve this problem, the idea is to build adaptable download schedules on the ground and to use the latest information about real volumes on board to adapt them. More precisely, a schedule is built on the ground with maximum volume assumptions for high-priority downloads and expected volume assumptions for low-priority downloads. Ground planning over a large horizon with sufficient computing power and time allows to produce a plan with a good quality and to take into account several criteria such as fair sharing between users or the age of information, which could not be possible on board. For that, we use a Squeaky Wheel Optimization scheme built upon a non-chronological greedy algorithm. The numerous problem constraints are checked with the constraint library InCELL. Because of the use of expected volumes for low-priority acquisitions, the plan is not directly executable onboard. Some volumes may be higher than expected and then the plan needs to be repaired. That is why onboard plan adaptation is performed before each group of sufficiently close reception station visibility windows. This adaptation is performed by a fast greedy repair procedure which builds a consistent plan from the ground plan by removing low-priority acquisitions when they endanger future high-priority ones. Bounds on resource availability computed on the ground allow to quickly check whether or not the real volumes of low-priority downloads endanger next high-priority downloads. With this look-ahead like mechanism, we ensure that all high-priority downloads will be done. When real volumes are lower than expected, the onboard repairing procedure tries to insert downloads that are not in the ground plan or to move forward future downloads. Onboard repairing allows to take advantage of real volumes and to update the plan in a limited time (compared to pure onboard planning). The combination of these two planning phases into a flexible scheme makes this approach a very competitive one, even better in some cases than pure onboard planning in terms of criterion and much more predictable for high-priority downloads.

Acquisition problem

We also investigate the acquisition problem which consists in selecting and scheduling observations. Nowadays, observation plans are built on the ground. A download plan is then built, and the whole plan is simulated. In this procedure, energy, temperature and memory constraints are checked. If they are violated, some of the low-priority acquisitions are removed and the constraint-checking process is started again. Because simulating the evolution of onboard energy and temperature is computationally expensive, it cannot be done finely during the acquisition planning process. It is replaced by high-level constraints with safety margins during planning. The parameters affected by these margins energy consumption, energy production, temperature evolution and observation volumes. Actual energy consumption is often lower than maximum and actual energy production is often higher than minimum, and so on. The real energy profile is then always higher than predicted and a lot of acquisitions that could have been done are eliminated when planning because of

the high-level constraints. To increase the system capability, the idea is to remove such constraint checks on energy from the ground planning process for low-priority acquisitions. However, contrarily to the data download problem, it is not possible to adapt the plans on board because of limited computing capabilities and because this adaptation is far more complex than the adaptation of data download schedules. Then, the onboard software can only execute plans. In a new approach, observation plans produced on the ground are conditional plans involving conditions for triggering low-priority acquisitions. Building these conditional plans involves replacing high-level constraints by the determination of minimum levels of energy that must be present on board at the beginning of each low-priority acquisition to ensure that, even if the low-priority acquisition is performed, future high-priority acquisitions can be done. These levels are produced using a backward simulation scheme that starts from the end of the horizon and computes realization bounds on the levels of energy. Once on board, before each low-priority acquisition, if the actual level of energy is higher than the required level computed on the ground, the acquisition is performed. If not, the observation instrument is not turned on, and energy is then saved. Compared with the current approach, this approach avoids wastage of resource and allows more acquisitions to be executed. Remaining work includes computing alternative satellite movements to save more energy in case of acquisition cancelling (attitude movements needed for the acquisition are still performed), extending this approach to other resources such as temperature, and generalizing to more generic problems.

Document structure

This document is structured as follows:

- Part I is about the context of this study. Chapter 2 describes a typical Earth-observation mission. Chapter 3 points out related work in decision-making under uncertainty.
- Part II is about the download planning problem. Chapter 4 proposes a mathematical model for this problem. Chapter 5 presents algorithms to solve the deterministic version of the problem. Chapter 6 presents our flexible approach and experiments.
- Part III details the work done on the observation planning problem. In Chapter 7, we describe the problem and the current approach to solve it, our flexible approach for this problem, and the experiments.

We then conclude in Chapter 8, contributions are summed up, and a critical analysis is formulated. Possible future works are pointed out.

Part I

Context and related works

A typical Earth-observation mission 2

In this chapter, we describe the components of a typical Earth-observation mission, the different actors it involves, and the decision-making process leading to activity plans.

The space system we consider is made of an agile Earth-observation satellite (see Sect. 2.2.2), several ground satellite control stations, several ground data reception stations, and several data processing centers, each one associated with a satellite user. In the following paragraphs, we describe several parts of the system (see Fig. 2.1).

2.1 System architecture

The system is composed of both a ground segment (described below) and an Earth-observation satellite (described in Sect. 2.2).

2.1.1 Mission center

The mission center receives observation requests from system users and produces acquisition and download plans. These plans are then transmitted to ground control stations. The planning process is complex and several exchanges of messages between involved entities may be necessary to validate activity plans.

2.1.2 Ground control stations

Control stations receive activity plans produced by the mission center and upload them onto the satellite. They also receive the so-called health telemetry. The latter is a set of data sent from the satellite to the ground stations which allows to check whether all embedded systems are working properly. They also receive the list of observations executed, the list of observations deleted, and the list of observations downloaded. Control stations are present accross the world. For example, the CNES has access to the following network of control stations : Kerguelen in the Kerguelen Islands; Kiruna in Sweden; Aussaguel in France; Kourou in French Guiana; Hartebeesthoek in South Africa; Perth in Australia. All stations are connected with the mission center with a high-rate link.

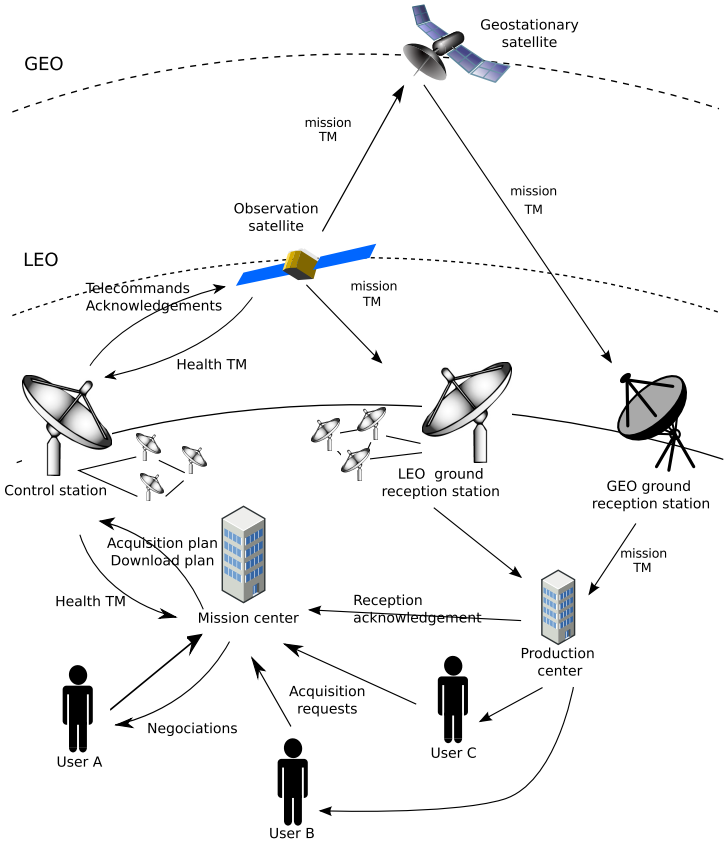


Figure 2.1: System architecture.

2.1.3 Ground reception stations

Ground reception stations are exclusively used for receiving mission telemetry which corresponds to data collected by the satellite instruments (images in an Earth-observation mission). Such data can be received when the satellite is in visibility, that is when its telemetry antenna is able to communicate with the station antenna. Each user has one or several allowed stations for downloading data. Acquisitions received in a ground reception station are then sent to the user's production center. The list of acquisitions downloaded and the list of acquisitions stored on board the satellite are sent to the mission center. It is important to maintain consistency between ground stations and the satellite so that algorithms producing plans can rely on the most recent information. Because it is not always possible to maintain such a consistency, because of data transfer latency for example, it is also necessary to think about plan repair at execution when the plan built on the ground is not fully consistent with the real state of the satellite.

2.1.4 Users and observation requests

Observation requests are emitted by users of the system. Each observation is defined by:

- a geographical zone to observe which is a polygon split into strips (see Fig.2.2); each strip is referenced by its geographical coordinates, an observation duration, a data volume; and sometimes weather forecast; the observation must be carried out during one of the visibility window of the strip;
- a priority level; priorities are said to be *tight* because the satisfaction of any request of the high-priority is always preferred to the satisfaction of any set of requests of lower priorities;
- a weight, allowing to sort acquisitions which have the same priority level;
- maximum observation angles: the larger the observation angle, the worst the quality; the best quality is given with a nadir pointing , that is when the satellite is pointing towards the center of the Earth.

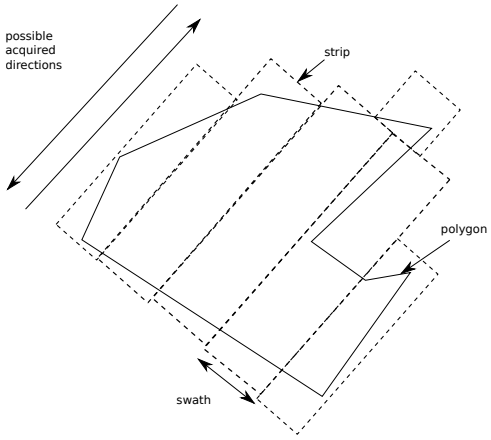


FIGURE 2.2: A geographical zone split into strips.

Once observation requests have been performed, associated data are sent to production centers which transform this raw mission telemetry into an image product. Each user of the system has its preferred production center.

2.1.5 Geostationnary satellite

The network of ground reception stations for low Earth orbit satellites is not large enough to cover the surface of the Earth. Geostationary satellites have a high altitude of around 36000 kilometers which gives them a large coverage (1/3 of the Earth surface). Thanks to their orbit, their pointing is stationary above a given point. When a low-orbit satellite is flying over a zone without much ground reception stations, the only way of immediately downloading data to the ground is to transmit data to geostationary satellites with an optical or a radio link. The geostationary satellite, with its much larger covering, will be able to download this data onto a ground reception station. This kind of satellite is frequently used in telecommunications and meteorology. In some missions, it is possible to make use of a network of geostationary

satellites as a relay for downloading data to ground reception stations (such as the *European Data Relay System*).

2.2 Characteristics of Earth-observation satellites

In the following, we review some of the characteristics of a typical Earth-observation satellite. We speak of *attitude* to designate the orientation of the satellite with respect to an inertial frame of reference or another entity such as a planet. A satellite is able to :

- perform attitude change maneuvers thanks to gyroscopic actuators. See Sect. 2.2.2 for more details;
- perform orbital maneuvers thanks to engines with liquid propellants;
- perform observations thanks to a telescope;
- download observations thanks to a telemetry antenna;
- perform heliocentric pointings, towards the sun, to fill up its batteries with solar energy, and near-heliocentric pointings to favor download efficiency while refilling the batteries;
- perform geocentric pointings towards the center of the Earth, to make communications with ground reception stations easier.

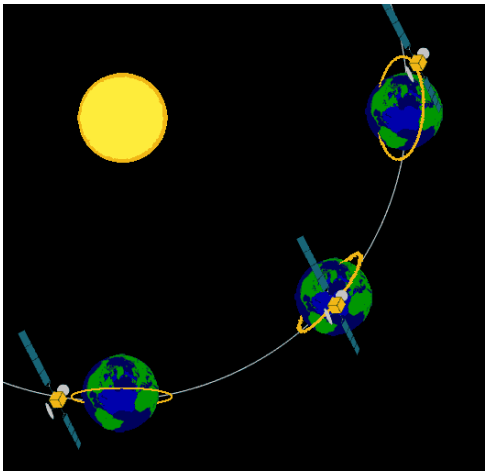


Figure 2.3: View of an heliosynchronous orbit with respect to the Sun.

2.2.1 Orbit

The satellite is placed on a low geocentric orbit at an altitude between 700 and 1000 kilometers. This orbit is:

- **heliosynchronous**: the angle between the orbit plan and the sun direction is more or less constant (see Fig. 2.3) though the Earth equatorial bulges causes a precession on this orbit;

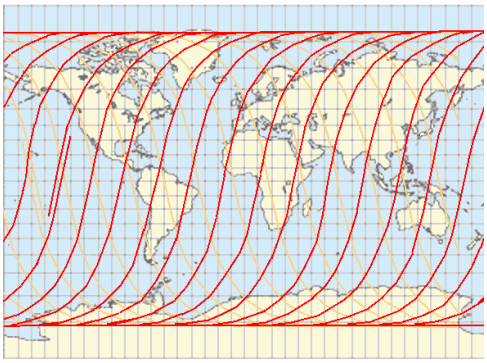


FIGURE 2.4: Track of an heliosynchronous orbit.

- **near-polar**, the orbit is inclined compared to a line going from the North pole to the South pole but the inclination is very small, hence the satellite trajectory is almost above the poles;
- **of short periodicity**; the satellite comes back frequently above a point at the same solar hour (2 to 3 days).

Heliosynchronism combined to Earth rotation allows to cover almost entirely the Earth surface in a given period (see Fig. 2.4). These orbital characteristics favor similar sunshine conditions at each come back above a point which is interesting when one wishes to follow the evolution of a phenomena on the ground.

2.2.2 Platform

The satellite considered in this manuscript must be compatible with the VEGA European launcher, which is specialized in placing payload at low-orbit (see Fig. 2.5). The satellite mass will be less than 1500 kilograms, its power consumption will be about 2 kilowatts, its lifetime will be 7 to 10 years, and it will be electrically or chemically propelled.

Agility Non-agile Earth-observing satellites have only one degree of freedom thanks to an orientable mirror put in front of each instrument. This mirror is actionned on the roll axis between acquisitions. It allows the satellite to take image at the right or left side of its ground track. An image is then produced by the movement of the satellite on its track on the ground (see Fig. 2.6). Time windows during which image acquisition is possible can be pre-computed for each observation request. Indeed, there is only one opportunity in each orbit to perform a given observation (see the left of Fig. 2.7). When two opportunity windows overlap or are too close to each other, it means that the two associated observations are not compatible. A choice has then to be done. Therefore, the resulting observation planning problem includes a selection problem.

In this study, we consider planning and scheduling for an *agile* Earth-observing satellite (such as Pléiades). With this type of satellite, instruments and solar panels are body-mounted (to avoid low-frequency attitude perturbations) and the platform is able to move quickly around its gravity center, along the roll, pitch and yaw axes,

Figure 2.5: A Vega launcher. (Source: ESA)

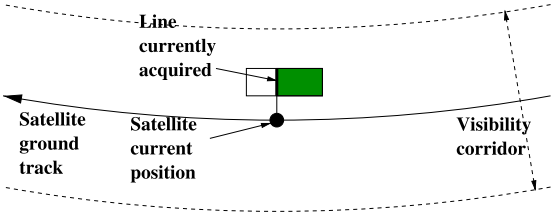


Figure 2.6: Earth-observation with a non-agile satellite.

thanks to gyroscopic actuators, while moving along its orbit [Chrétien et al., 2004]. The two additional agility axis, compared to the non-agile case, allow to greatly increase the number of opportunities for image acquisitions. Indeed, the satellite can point ahead of its position or backwards with regard to nadir (see Fig. 2.8), the choice of the start of acquisition becomes free in a time window that can be pre-computed (see Fig. 2.7). Such time windows are longer than when the satellite is not agile, and some acquisitions that were not compatible become compatible thanks to this agility (see the right of Fig. 2.7). The resulting planning problem includes a selection and a scheduling problem, and it becomes more difficult.

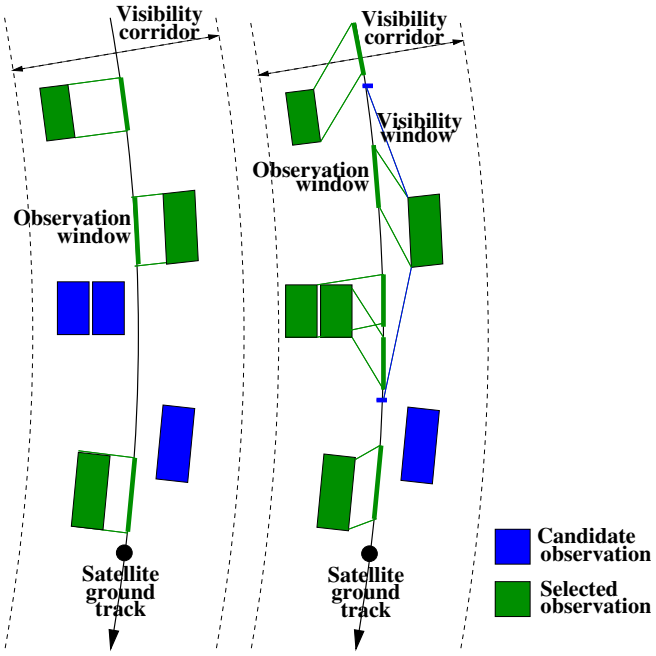


FIGURE 2.7: Comparison of observation capabilities between a non-agile satellite (left) and an agile satellite (right).

Moreover, for agile satellites, minimum transition durations between two successive observations depend on the start date of the transition (see Fig. 2.8). Computing these minimum durations is a continuous optimization problem under constraints [Beaumet et al., 2007], which can be solved using solvers such as RealPaver [Granvilliers and Benhamou, 2006] or IBEX [Chabert and Jaulin, 2009], or approximate solution techniques (see Sect. 4.2.3).

The ability of the satellite to do attitude movements combined to its orbital trajectory allows to perform acquisitions by scanning the zone to acquire. *Observation profiles*, that is attitude movements done by the satellite while performing an observation, are difficult to compute, and they depend on the start date of the observation.

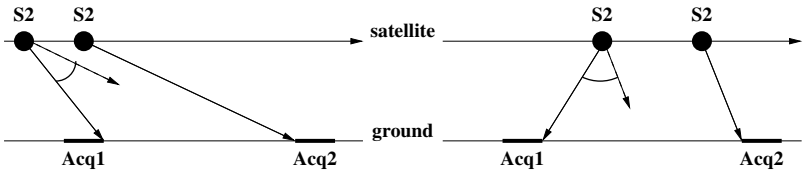


Figure 2.8: The duration of transition between two acquisitions depends on the angle to travel and then on the start date of the transition.

Orbital maneuver An orbital maneuver allows to change the orbit of a spacecraft thanks to a propeller system. For instance, orbital maneuvers are necessary to place space probes into orbit around planets. Regularly, Earth-observation satellites deviate from their orbit. It is then required to perform orbital maneuvers to rectify their trajectories. These maneuvers, which are consuming propellant, are done above zones that are not frequently observed such as poles.

2.2.3 Image acquisition chain

We now describe the onboard image acquisition chain that is visible on Fig. 2.9.

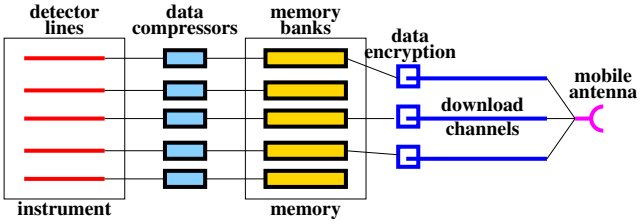


Figure 2.9: How data is acquired, recorded, and downloaded.

2.2.3.1 Telescope

To realize observations, the satellite is equipped with a telescope. Its large primary mirror has a resolution of 20 to 30 centimeters and a swath of 15 to 20 kilometers. It is possible to take a minimum of 250 images per day. The focal plane is made of several detector lines, each of which is dedicated to a frequency range. Several detector lines make an image channel. The following constraints must be taken into account :

- a minimum pre-heating time is required before using the telescope;
- the temperature increases linearly when the instrument is activated and has an exponential decrease when the instrument is put off;
- the temperature of the focal plane must not exceed a value which limits the continuous period of use;
- there must be a limited amount of ON/OFF cycles because they degrade the instrument reliability;

(a) High-density image (b) Cloudy image (c) Low-density image

FIGURE 2.10: Three observations with different settings. (a) is a high-density image that will almost not be compressed. (b) includes clouds that will be compressed. (c) has no cloud but has a low-density; these zones will be compressed on board.

– sun dazzling is prohibited: the angle between the satellite axis and the sun direction must always be greater than or equal to a minimum value.

2.2.3.2 Onboard mass memory

The satellite has a mass memory, managed by a standard file system, designed to store roughly 2 days of acquisition. Several instrument modes may be activated for performing an observation. Each mode activates some of the telescope image channels. Each image channel is linked to a memory bank in the mass memory (see Fig. 2.9). An observation is then divided into several files, one for each image channel. A memory bank can only store a limited amount of data. Deleting an observation from the mass memory is decided by the ground after the observation has been validated. It is also possible to delete a low-priority observation to free space for a high-priority observation or when an observation is too old.

After acquisition, *selective compression* is used on the image, to compress parts of the image such as low-density zones or clouds (see the example on Fig. 2.10). The number of bits necessary to encode pixels of these zones may typically be reduced from 12 to 3. The volume of data generated by an acquisition is then very variable (varying from 1 to 4 times the minimum size). It can only be estimated before acquisition and observed after acquisition. Until now, because plans must be executable onboard, maximum volumes (no selective compression) are taken into account during the planning phase. Download plans are then sub-optimal because real volumes are very often lower than maximum (compression is frequent). This variability is one of the motivation for giving more decision-making autonomy to the satellite.

2.2.3.3 Telemetry antenna

To realize downloads, the payload telemetry includes an orientable antenna mounted on the Earth side of the satellite. It allows to cover almost the whole half space. It has a rotation speed of 5° per second and its rate is 2 Gbits per second. It has an emission cone of only a few degrees, meaning that it cannot point towards several stations at a time.

Once the antenna pointing direction has been decided, the satellite-station tracking is automatically carried on, taking into account current attitude movements. The ground reception station and the satellite use ephemerides to track themselves mutually. It is also possible to have dynamical tracking by signal servoing between the satellite and the station.

When planning downloads, one must take into account the rotation speed of the antenna, the satellite's attitude and the station coordinates. A minimal transition duration (that we will also call *antenna transition*) is required before starting communication with a ground station even if the satellite is in the reception visibility cone of the station (*pull-in time*).

The antenna rate depends on the satellite-station distance. When the satellite is far from the station, signal dispersion induces a greater error rate. In addition, at this distance, the satellite is low-angled, meaning that the signal is traversing more atmospherical layers than when the satellite is at the zenith of the station (see Fig. 2.11). Until now, the link rate was fixed at the worst possible rate, obtained when the satellite-station distance is maximum. In the future system, the rate is said to be *smart* because it varies during each communication window, depending on the error rate. There is a higher rate when the satellite is at the zenith of the station (see Fig. 2.12).

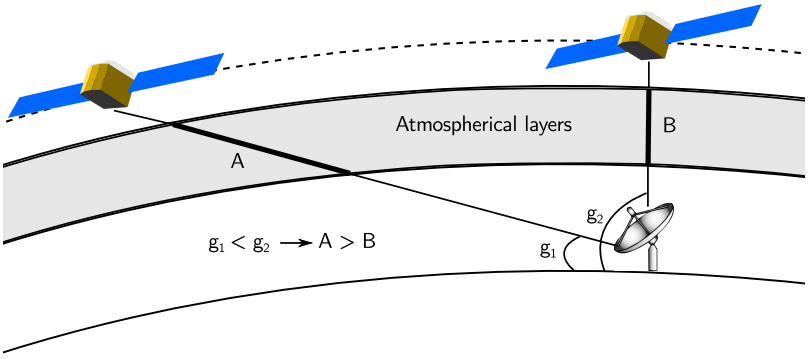


Figure 2.11: The more the satellite is low-angled with regard to the surface, the more atmospherical layers the signal has to traverse.

2.2.3.4 Data download

The satellite can download observations when it is visible from a ground reception station (when the satellite is in a *visibility window*), when its telemetry antenna is oriented towards the station, and when the ground station antenna is oriented towards the satellite. Then, effective communication becomes possible for a certain duration that we call *download window*. Observations are downloaded according to a specific order determined by the download plan.

Data downloading can use several concurrent emission *channels* (see Fig. 2.9). Each file resulting from an observation can be downloaded using one channel. Interrupting a file download is not acceptable. Due to the movement of the satellite on its orbit and of the Earth on itself, the satellite-station distance evolves and hence

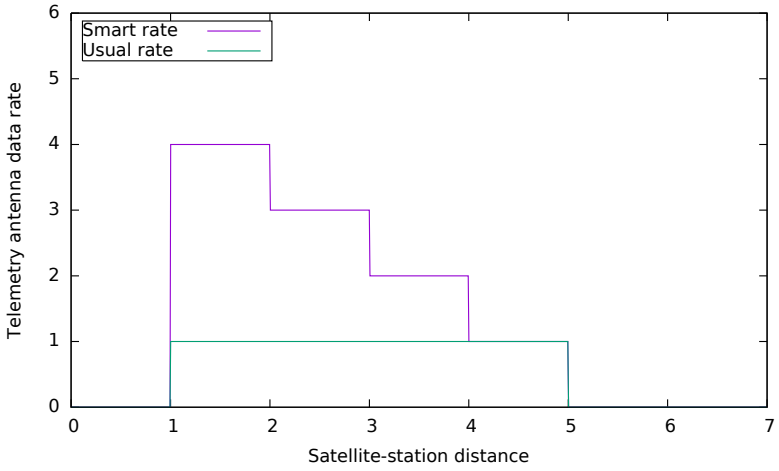


FIGURE 2.12: Shape of the telemetry antenna link rate according to satellite-station distance.

the download duration of a file depends on the time at which download starts. Files resulting from an acquisition can be downloaded in any order using any channels, but must be all downloaded within a single visibility window. Channels and memory banks are unsharable resources. This means that, if two files are recorded on the same memory bank or use the same channel, their downloads cannot overlap.

Data is also encrypted before download. One encryption key is associated with each entity (e.g. a country, a group of countries, or a company). There is one physical encryption component for each download channel, allowing data associated with several users to be downloaded concurrently. The operation of changing a key on a component is immediate but needs to be written as an entry in a global key change table. The number of entries it is possible to record in this table is limited. Making changes in this table takes a small amount of time, typically 2 seconds; and it requires stopping all download activities.

2.2.3.5 Instrument switching

To save energy, to keep their temperature below a given threshold, and to maximize their lifetime, the acquisition and download instruments (optical observation instrument and data emission antenna) are not maintained permanently active. Each of them has three modes: ON, standby (SDBY), and OFF, in a decreasing order of energy consumption. Each instrument is maintained ON only when necessary: effective acquisition or download. Between acquisitions or downloads, it is maintained in the SDBY mode if and only if the duration between the two successive acquisitions or downloads is less than or equal to a given threshold. Otherwise, it is maintained OFF.

2.2.3.6 Instrument temperature

If we consider an acquisition and download schedule, including the associated instrument switching plan, we can build an approximation of the temperature profile of each instrument and check whether or not temperature is always lower than or equal to a maximum acceptable level. Important margins are used in that domain because it is very difficult to model thermal interactions between the platform and the instruments.

2.2.3.7 Onboard energy

Energy is consumed by the satellite platform, and by the acquisition and download instruments according to their current mode (ON, SDBY, or OFF). It is produced by solar panels which are body-mounted on the satellite. Because the satellite orbit is quasi-polar, low-altitude, and helio-synchronous, it alternates day and night periods (eclipse of the Sun by the Earth during night periods). Energy production is only possible during day periods and, within these periods, it depends on the satellite attitude trajectory. Produced energy is stored in batteries whose capacity is limited. See Fig. 2.13 for a typical energy profile. $Emax$ is the maximum energy level, equal to the battery capacity: the energy level cannot be greater than $Emax$. $Emin$ is the minimum acceptable energy level: the energy level must not be lower than $Emin$. If we consider an acquisition and download schedule, including the associated instrument switching plan, we can build the associated energy profile and check whether or not the energy level is always greater than or equal to $Emin$.

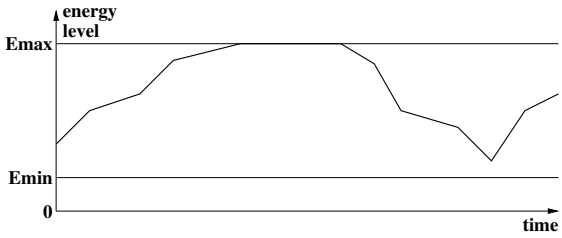


Figure 2.13: Typical energy profile.

2.3 Decision-making process and uncertainties

The satellite evolves in a dynamic environment and it must face several sources of uncertainty :

- the cloud coverage, conditioning the success of observations; weather forecasts are useful but not always accurate;
- the volume of files after selective compression; because the cloud coverage is not exactly known in advance, the compression rate is not known either;
- the reliability of the onboard/ground data link, which is influenced by interferences; the actual quality of the data link can alter download durations;
- the energy available on board, hard to model because it varies according to the production by the solar panels and the consumption by the satellite systems;

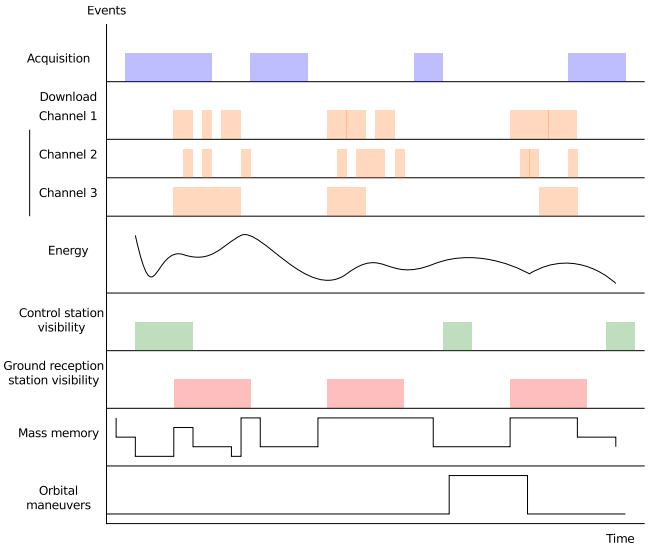


FIGURE 2.14: Timelines representing a plan execution.

– the arrival of new urgent observation requests.

These uncertainties influence activity planning. We say that the environment and the resulting planning problem are *dynamic* because problem parameters are quickly evolving. Planning and scheduling problems with uncertainties results in large search spaces (as it includes hundreds of acquisitions per day in our case). Plans not taking these uncertainties into account are destined to fail [Bresina et al., 2002b]. The current approach consists in building plans offline on the ground (typically every 8 hours with a 24-hour gliding planning horizon), where acquisition and download schedules are produced with precise start and end times for activities. These schedules verify all the constraints related to satellite attitude, to acquisition and download, and to onboard energy, memory, and temperature. They are uploaded to the satellite during ground control station visibility windows and executed by the satellite without any change (see typical timelines on Fig. 2.14).

The existing acquisition and download schedule building process is illustrated in Fig. 2.15. At the highest level, an acquisition schedule and then a download schedule are proposed, taking into account user request priorities and high-level aggregated constraints such as no more acquisition time than a given threshold over each satellite revolution or each set of consecutive satellite revolutions. The proposed schedules are then checked according to all the constraints. In case of any constraint violation, they are modified by removing for example some acquisitions or downloads. The process continues until a schedule that satisfies all the constraints is found. This process always terminates because an empty schedule (no acquisition and no download) is always feasible if we assume a nominal satellite state and behavior.

To be sure that the proposed schedule is really executable by the satellite in spite of the numerous uncertainties, safety margins are used when checking constraints. For example, the expected energy production and consumption rates are replaced

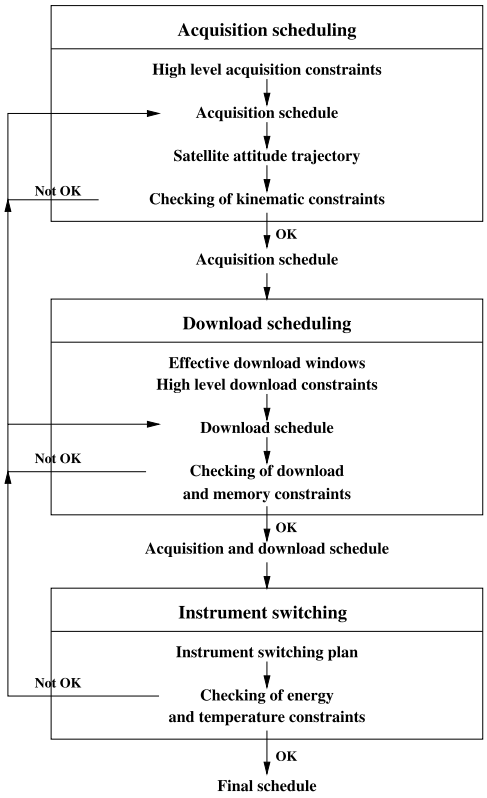


Figure 2.15: High-level view of the acquisition and download schedule building process.

by minimum production and maximum consumption rates, resulting in harder constraints.

The goal of this thesis is to go beyond this existing scheme, and to improve system performance through a better management of uncertainty. In the next chapter, we will see how it is possible to deal with uncertainties in combinatorial problems such as planning and scheduling.

Combinatorial optimization under uncertainty 3

"Une fois pris dans l'événement, les hommes ne s'en effraient plus. Seul l'inconnu épouvante les hommes."

— Antoine de Saint-Exupéry, Terre des hommes

Combinatorial optimization is a branch of mathematical optimization. The goal, in a combinatorial optimization problem, is to find the best element in a finite set of possible elements. Generally, this set is very large and complete exploration is impossible. Some methods have been developed to solve specific problems such as planning or scheduling. In this chapter, we briefly review techniques developed for tackling these problems in uncertain environments.

3.1 General considerations

First, we define some useful notions.

Determinism Determinism is a simplification of the world where actions have only one possible known outcome. In a deterministic world, results of successive dice rollings are known because all factors having an impact are known as well as the evolution of these factors.

Non-determinism is more realistic and proposes to model all the possible outcomes of an action. For example, an action can either succeed or fail. Taking this into

account may be of importance for critical systems. Planning in non-deterministic domains features several difficulties compared to deterministic planning. On one hand, problem modeling is harder in terms of granularity and comprehensiveness because additional variables have to be added to represent uncertainty. On the other hand, the search space is much larger and then longer to explore. When the different outcomes of an action can be associated with a probability, we speak of a *stochastic* action (see Fig. 3.1).

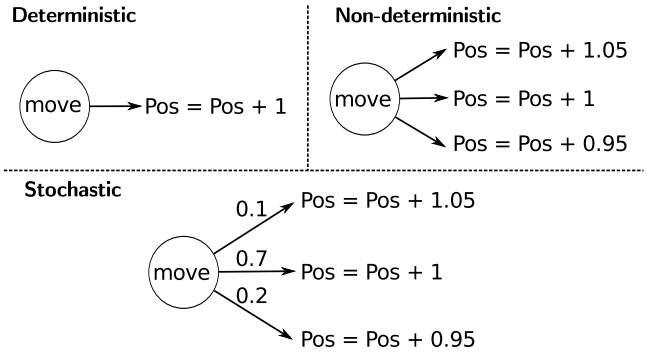


Figure 3.1: Difference between a deterministic action, a non-deterministic action, and a stochastic action. The action in the upper-left part is deterministic, it has only one outcome. The action in the upper-right part is non-deterministic, it has several outcomes. The action in the bottom part is stochastic, it has several outcomes, weighted by a probability of occurrence.

Observability Once sources of uncertainty have been sorted out, it is possible to characterize them. The *state* of the system is considered to be the set of variables needed to make a decision involving the system. For example, it may be the position of a robot or the available power in its batteries. A state is said to be *totally observable* if the agent knows the values of all variables at every time. It is *partially observable* when the value of some of the variables is not accessible. In this case, the planning system may have to make assumptions about these variables.

Robust planning and scheduling Several definitions are possible for robust planning and scheduling but it is an important subject with regard to uncertainty. A robust plan can be a plan that maximizes an expected reward but it can also be a plan built offline with the aim of *resisting* or *absorbing* the exogenous events that could disturb the execution [Cichy et al., 2005] (meaning that uncertain parameters revealed during the online phase should not impact the plan built during the offline phase), a plan ensuring that the solution quality will not deteriorate [Sevaux and Sörensen, 2002], or a plan which can be quickly adapted in face of the events encountered at execution [Ginsberg et al., 1998].

Globally, the principle of robust planning and scheduling is to minimize the number of changes at execution but it is not practicable to consider only an offline phase unless the environment is assumed to be deterministic [Rasconi et al., 2010]. Some authors are proposing methods to evaluate the *robustness* of such robust plans [Fox

et al., 2006b] [Rasconi et al., 2010]. For example, in our case, a satellite activity plan is built to satisfy several users. The level of satisfaction is measurable for each user. One of the goal can be to stabilize this level of satisfaction when replanning compared to a previous plan. These levels of satisfaction depend on agreements between users before sending the plan on board. If the system is supposed to be reliable, it should automatically take these levels into account when planning.

Phases of decision-making In deterministic decision-making, available actions are deterministic. When planning, data is supposed to be available and a plan can be computed. It is often easier to compute a plan of good quality in this setting although it may be inconsistent with the environment when it is executed. In decision-making under uncertainty, data is not supposed to be complete when planning and/or it is updated during execution. We then distinguish two phases in decision-making:

- the *offline* phase which is the phase before the execution of the plan; during this phase, only a part of the data is known;
- the *online* phase which is the phase during the execution of the plan; during this phase, uncertainties are progressively cleared up.

For example, if we look at a scheduling problem, we can define two subproblems : the static subproblem and the dynamic subproblem [Rasconi et al., 2010]. The static subproblem (offline phase) consists in assigning start dates and end dates to activities while ensuring constraints and optimizing a criterion. The dynamic subproblem (online phase) consists in monitoring the execution and in repairing the plan when necessary. In our satellite case, we have three phases : offline phases on the ground, offline phases on board and online phases on board (see Sect. 6.1). We will see that the planning and scheduling frameworks presented in this chapter can also be distributed among three categories: *proactive* approaches, *reactive* approaches, and *flexible* approaches. Each category address more or less a specific decicion-making phase.

Proactive approaches A way of dealing with problems that include uncertainty is to adopt a proactive approach, that is to take uncertainty into account mainly during the offline phase. It can be done by modeling the missing or uncertain information and by building a plan or a family of plans which will be the most efficient in the light of the information that is available when planning. In this setting, we consider that it is possible to anticipate uncertain events and that the resulting plan will only need minor changes at execution. Most of the approaches presented from Sect. 3.2 can be used as proactive planning schemes.

Reactive approaches Another way of planning in a dynamic and uncertain context is to build a plan during the *online* phase, that is when most uncertainties have been cleared up. This approach often needs a fast decision-making paired with an immediate execution. Decisions may be locally optimal but are seldom globally optimal. Local repair is a reactive approach that is applied when a baseline plan becomes inconsistent because of exogenous events. The objective is to make the plan consistent again by making a relatively small number of changes. Compared to replanning, local repair does not start from scratch. It can be an interesting approach when plan stability is wanted, that is when the number of changes has to be minimized.

When a baseline plan is computed in the offline phase, reactive approaches can be local [Smith, 1995] if a relatively small set of activities is updated, or

global if the whole plan becomes inconsistent [Sakkout and Wallace, 2000]. Reactive planning approaches have been experimentally used in the space domain with CASPER [Knight et al., 2001] [Chien et al., 2004b] on the EO-1 satellite, RAX [Bernard et al., 1998], PROBA [Teston et al., 1999] and RASSO [de Novaes Kucinskis et al., 2007] to deal with real-time events such as natural phenomenons. In [Beaumet et al., 2011], authors have investigated a reactive onboard planning scheme for an agile Earth-observing satellite. In [Fox et al., 2006a], replanning and local repair are compared and it is shown that the latter allows a greater plan stability. Plan repair is also used in IxTeT-Exec, an extension of the temporal planner IxTeT which is dedicated to plan execution [Lemai and Ingrand, 2004]. In IxTeT-Exec, replanning is used when it is impossible to repair the current plan. In ASPEN, which is used in space missions, an iterative conflict repair is continually updating the plan according to the events [Rabideau et al., 1999]. More generally, these approaches are exploring the links between planning and execution in critical environments. Several methods may be considered for handling uncertainty about the future in a reactive planning framework:

- the future is *determinized*, meaning that one outcome is chosen for each action among the set of possible outcomes;
- the future is stochastic and sampling is performed to make the problem tractable; we will see several methods such as *Hindsight Optimization*, Monte-Carlo based methods, and *Online Stochastic Combinatorial Optimization* which can be used in a reactive framework.

Flexible approaches Flexible planning and scheduling is a combination of proactive and reactive approaches. Over the whole set of decisions to be made, some are made offline and some are made online or closer to execution. This allows to divide the workload between the two phases. Computing resources are often more important during the offline phase but uncertainties are large. On the contrary, during the online phase, computing resources and uncertainties are low. Flexible approaches produce plans which are designed to be adapted during the online phase. Note that though we present these methods as *flexible*, it is true that boundaries between flexible, reactive, and proactive approaches are sometimes a bit fuzzy.

In scheduling, there exist several forms of flexibility : temporal flexibility, flexibility on resource access order and flexibility on resource assignment [Billaut et al., 2013]. In [Wu et al., 1999], partial-order plans are built, providing temporal flexibility. Highly constrained subsets of activities are partially ordered. Temporal aspects such as exact start dates are set only at execution. This approach requires high computing resources both when planning and when executing to transform the offline order into a executable plan. *Partial-Order Schedules* (POS) synthesis [Policella et al., 2004b] also provides temporal flexibility. A POS is a graph where nodes represent activities and edges represent precedence constraints between pairs of activities such that any temporal solution (start and end dates) enforcing these constraints between activities is valid. The approach follows a least commitment strategy in which tasks are ordered according to their needs in resources. For that, precedence constraints are introduced for tasks accessing the same resource. Almost no computation is required online because all conflicts have been solved contrarily to the previous approach. In [Policella et al., 2009], robust and flexible plans are built from completely instantiated plans. To do that, a high-quality plan is first built. Then, temporal flexibility is added by transforming the plan structure into a *Partial Order Schedule*. In this method, the transformation of an instantiated plan into a flexible plan does

not diminish the plan quality in terms of criterion. On the other hand, a very high-quality plan can be difficult to transform because it is highly constrained and adjusted. In [Orlandini et al., 2011], temporally flexible plans are represented as automatons (*Timed Game Automaton*) which makes them verifiable (using a *reachability game*) with a formal verification tool (UPPAAL-TIGA [Behrmann et al., 2007]). In the parametric dispatching approach [Gerber et al., 1995], the goal is to schedule tasks in a real time system. These tasks are ordered in a list and have temporal constraints. In this approach, there are two components : an offline component that computes lower and upper parametric bounds for each task such that each bound is a function of the actual realization date of the predecessors of the task. Only the first task has non-parametric bounds. Parametric bounds allow to have flexibility when executing the schedule. The online component updates these bounds and launches the tasks: when the actual bounds of a task are computed, the online component may dispatch the task during the time window defined by the bounds. Another form of flexibility consists in optimizing not only several temporal or resource instantiations given a unique baseline plan but how plans can be quickly changed at execution. For example, in [Jensen, 2003], the objective is to build a plan that is easily adaptable on board when unexpected events occur. For that, the optimization focuses not only on a specific plan but on a neighborhood of plans. In case of replanning, a high-quality plan is easily found in the immediate neighborhood.

3.2 Planning under uncertainty

We now give a closer look to a classical combinatorial optimization problem, planning, as it has many common features with our data download and observation problems. A classical planning model Q is defined as a tuple $Q = \langle S, s_0, S_G, A, f \rangle$ where:

- S is a finite set of states;
- $s_0 \in S$ is the initial state,
- $S_G \subseteq S$ is the set of goal states,
- A is the set of operators (the actions),
- a state transition function $f : S \times A \to S : (s, a) \to f(s, a) = s'$.

More precisely, given a description of the initial state of the world, a description of the desired goals, and a description of a set of possible actions, the solution to a *planning* problem is a plan, that is a sequence of actions, that leads to one of the goal states. Several strategies exist for problem involving uncertainty about the *effects*, that is the result of the state transition function, or durations of actions.

3.2.1 Conformant planning

One of the most important assumption when planning concerns state *observability* and several planning frameworks are developed to address each possible assumption. Conformant planning is a subfield of classical planning where the aim is to find a plan, in a non-deterministic domain, guaranteeing that the goal will be reached with no *observability*, that is when the result of actions is not known at execution. In a *classical planning* problem, there is an initial state, a goal and a set of actions that modify the state. A solution to such a problem is a sequence of actions that leads from the initial state to the goal. In conformant planning, the goal must be reached whatever the effects of the actions and whatever the initial state among a set of possible initial states [Hoffmann and Brafman, 2006]. This problem is then

much more complex than classical planning. Verifying a conformant plan is a hard problem in itself because one has to explore all the possible outcomes of the actions. To solve this problem, it is possible to search in the *space of belief states*, that is the space where the elements are the *possible* states of the world given actions previously executed. Unfortunately, the number of elements in this space is exponential with regards to the state space. Authors are then trying to represent these elements in the most compact way and in developing efficient heuristics to quickly move into this space [Palacios and Geffner, 2009]. See Fig. 3.2 for an illustration of the space of belief states and an example of conformant plan.

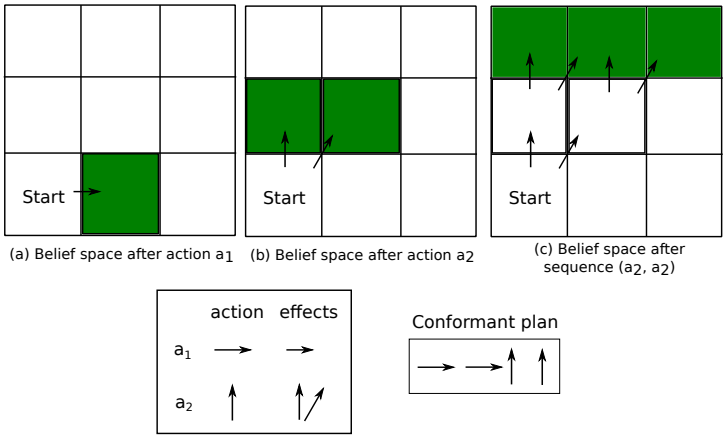


Figure 3.2: A conformant plan for a robot moving on a grid. Action a_1 is deterministic, action a_2 is non-deterministic. Whatever the initial position of the robot on the grid and the action effects, the execution of the conformant plan will succeed in reaching the goal. Belief spaces in different cases is also shown.

Although the idea of computing a plan or a schedule that will always remain consistent is interesting, modeling our problems as conformant planning problems would not be satisfactory. Indeed, our problems do not need the assumption of no observability to be solved as uncertain parameters are observable on board the satellite. In addition, building a conformant plan for the data download problem would consists in assuming maximum volumes for all observations as it is already the case in the current approach and is too pessimistic.

3.2.2 Contingent planning

Conformant planning is adapted when the environment is not observable at all. When observations are available, other frameworks are used. Recall that when the actions are non-deterministic, several outcomes are possible for each action. With conditional planning, the plan is adapted to observed outcomes of actions and to observed exogenous events. A conditional plan has several *execution branches* depending on the current state of the system. The plan produced is not a sequence of actions but a directed root tree or a directed acyclic graph where each path from the root to a leaf is a complex execution sequence (see Fig. 3.3). This type of plan is tailored

to observable and partially observable environment because branching is based on perception, contrarily to conformant planning.

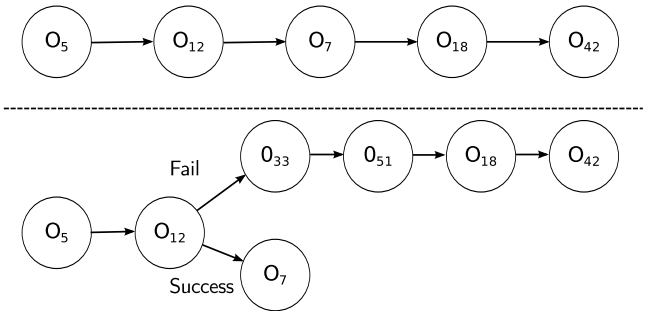


FIGURE 3.3: Above, a plan with five deterministic observation activities. Below, a conditional plan with two execution branches, allowing to respond in case of failure of O_{12} by branching to O_{33}. Here, only action O_{12} is non-deterministic.

There exist numerous planners able to build conditionnal plans such as MBP [Bertoli et al., 2001]. A classification of such planners can be found in [Dearden et al., 2003]. Contingent planning is hard in theory and in practice [Littman et al., 1998], and when solving real-world problems, it is often not possible to explore all the outcomes of all actions. This is why there exist several derived approaches.

When the potential number of branches is too important, it is possible to focus only on the most important ones. In the *Just-In-Case* approach [Drummond et al., 1994], developed for planning the observations of a telescope, a baseline plan is built and then analyzed. *Hot spots* are identified, that is steps in the plan where the risk of failure is the highest. Conditional branches are then inserted at these steps. In the same manner, [Dearden et al., 2003] tries to build contingent plans for a martian rover. In this context, conditional planning is considered because sequential plans often become inconsistent once onboard and the rover wastes time waiting for a new plan sent by the mission center. In addition, replanning onboard is not practicable because of the limited computing power available. Moreover, because of the criticity of the actions, plans have to be approved by human operators before being sent to the rover. To come up with uncertainty at execution time, a baseline plan is build and contingent branches are incrementally added based on expected gain of branching.

Another strategy is to build a catalog of static plans and to switch between plans according to observations made [Fohler, 1993]. Each plan is computed to fit into a specific situation called *mode*. Events triggering switches between plans are explicitly defined. In *conditional scheduling* [Greenwald and Dean, 1998], a model of the environment is also used to prove that produced plans are safe at execution.

Pure contingent planning would not be suitable for the data download problem. As all tasks are stochastic, computing conditional branches for each case would clearly be intractable. However, it is possible to consider a partial contingent planning approach for the observation problem, as we will see in Chap. 7.

3.2.3 Markov Decision Processes

When uncertain outcomes can be associated with probabilities, the planning problem may be thought of as a Markov Decision Process [Puterman, 2014].

A Markov Decision Process (MDP) represents a planning problem as an optimization problem in which the expected reward depends on the sequence of actions performed. Actions and outcomes are modeled as a state/transition system with probabilities for each transition. States and transitions can be associated with rewards which are then used to compute the total reward. When a state or an action is desirable, a positive reward is attached to it. Otherwise, a negative or less important reward is defined. An example of MDP can be seen on Fig. 3.4. More formally, an MDP is a stochastic automata $M = \langle S, A, T, I, R, \gamma \rangle$ with:

- S a finite set of possible states;
- A a finite set of possible actions;
- $T : S \times A \times S \rightarrow [0; 1]$ a probability distribution over transitions between states; $T(s, a, s')$ is the probability for the system to reach state s' when action a is applied in state s;
- $I : S \rightarrow [0; 1]$ a probability distribution over the initial state;
- $R : S \times A \times S \rightarrow \mathbb{R}$ a reward function for transitions; $R(s, a, s')$ is the reward when the system is going from state s to s' after action a;
- γ a discount factor (optional).

The objective is to compute a *policy* $\pi : S \rightarrow A$ specifying, for each state of the system, the action to choose.

To compute a policy maximizing the total reward, one can solve the Bellman equations. For each state s, let $V^*(s)$ be the maximum expected reward that can be obtained from s. This equation is recursive:

$$V^*(s) = \max_{a \in A} \left(\sum_{s' \in S} T(s, a, s') \cdot \left[R(s, a, s') + \gamma \cdot V^*(s') \right] \right)$$

The optimal policy $\pi^*(s)$ in state s, that is the action a to choose in this state can be obtained with dynamic programming and is given by :

$$\pi^*(s) = \arg\max_{a \in A} \left(\sum_{s' \in S} T(s, a, s') \cdot \left[R(s, a, s') + \gamma \cdot V^*(s') \right] \right)$$

When the environment is partially observable, that is when it is not always possible to observe the current state, *Partially-Observable Markov Decision Process* (POMDP) are used to model the problem [Kaelbling et al., 1998]. In this framework, two items are added :

- Ω a set of observations;
- $O : \Omega \times S \times A \rightarrow [0, 1]$ a probability distribution representing the probability of receiving an observation at a given state after performing an action.

The search is then performed in the space of belief states (as in the conformant planning framework). Thanks to Bayes rules, it is then possible to update the current belief state based on the last observation. Unfortunately, the sizes of both the data download problem and the observation scheduling problem are too important to be tackled by MDP and POMDP algorithms.

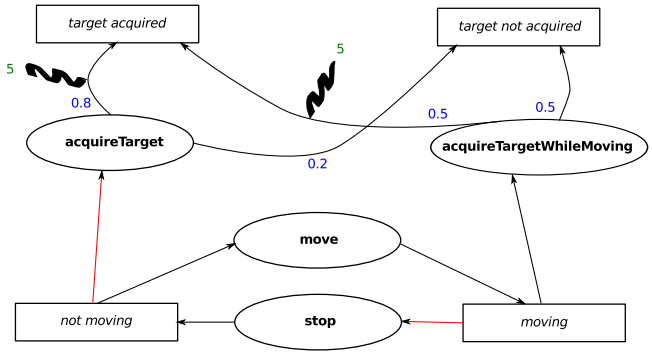


FIGURE 3.4: Example of an MDP representing a planning problem involving a robot equipped with a sensor. Rectangles are states, actions are circles. The robot can move around and try to acquire an image. A reward is obtained if the target is acquired. There are deterministic actions such as Move or Stop and stochastic actions. The sensor error is modeled as a failure probability. Here the optimal policy is to get to state *not moving* and then to try to acquire the target. The expected gain is then maximized.

3.2.4 Hindsight optimization

Hindsight Optimization (HOP) is a manner of using a deterministic planner in a stochastic environment to evaluate available actions in each state. In other words, it allows to compute a heuristic value for each action in each state. This heuristic value may be an estimation of the distance to the goal from the current state, for example. Fig. 3.5 presents an example of descent in a search tree used in hindsight optimization. In HOP, the idea is to build several possible futures, that is several deterministic subproblems for each action that has to be evaluated. In each deterministic subproblem, non-deterministic outcomes are removed for all actions by randomly choosing one outcome, transforming actions into deterministic ones. This process is done for each new deterministic subproblem in a non-stationary manner. It means that the actions may not have the same effect in all the problems. This allows to vary the determinized outcome of an action in a given state. Once determinized subproblems have been built, they are solved using a deterministic planner. The deterministic subproblems are much easier to solve than their stochastic counterpart. Once several plans have been produced, a criterion is computed for each of these plans. The criteria obtained are then aggregated to get the heuristic value of the currently evaluated action. Temporal and spatial complexity of this approach depends on the depth of exploration and on the number of deterministic problems that are solved per action. HOP is a technique originally developed for packet scheduling in communication networks [Wu et al., 2002]. It has recently been used in probabilistic classical planning [Yoon et al., 2008] [Kiesel and Ruml, 2014].

The idea of determinizing a problem with uncertainties to compute an estimate of the solution is relevant to our problems. In the data download problem, all file durations are uncertain but bounded by minimum and maximum values. Computing a download plan with minimum or maximum volume may allow to know whether some tasks are possible or not, and to compute higher and lower bounds for the

optimization criterion.

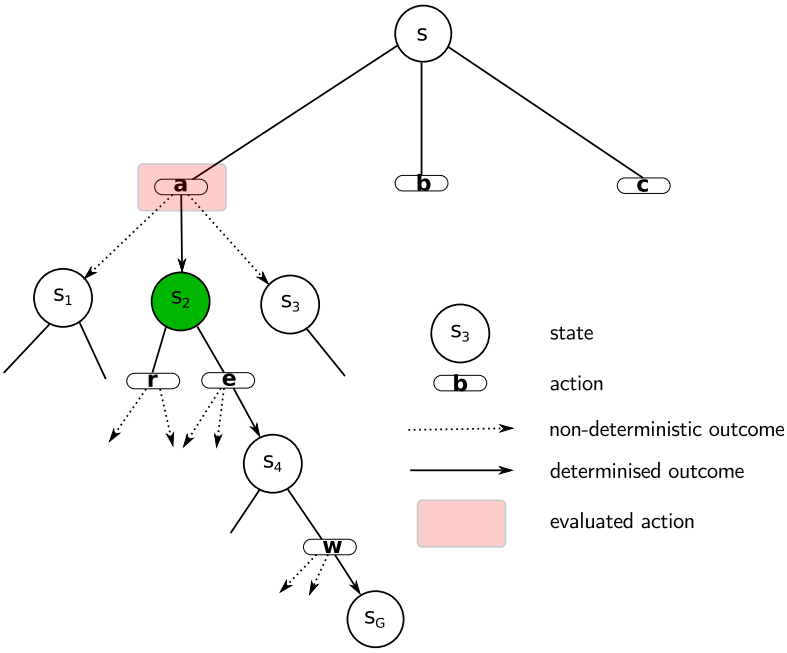


Figure 3.5: Example of one descent in a search tree used in hindsight optimization. The considered planning problem has actions with non-deterministic outcomes. The current state is state s and an action has to be chosen among the set of available actions $\{a, b, c\}$. The search tree shows the evaluation of action a. This action can lead to three state $\{s_1, s_2, s_3\}$. From state s until a goal state s_G has been reached, action outcomes are determinized. a is then determinized and leads to s_2. Action e is chosen randomly and determinized. It leads to s_4. Action w is chosen randomly and determinized. The distance between s_G and s can be used in the evaluation of action a. This process is repeated several times for each action to obtain robust heuristics. The action which leads to s_G in the smallest expected number of steps is chosen as the next action to apply.

3.2.5 Monte Carlo Tree Search

The problem of choosing an action in a stochastic environment may be modeled as an MDP with a lot of states. To select the best next action to do at a given time, a possible idea is to look ahead at a certain depth, that is to explore the future, and to estimate a reward for each action. Because this method is very costly if performed in a complete manner, sampling-based Monte Carlo methods have became popular (see the survey [Browne et al., 2012]).

Pure Monte-carlo Tree Search (MCTS) is a heuristic search algorithm that explores the search tree and tries to focus on parts of the tree such that it samples

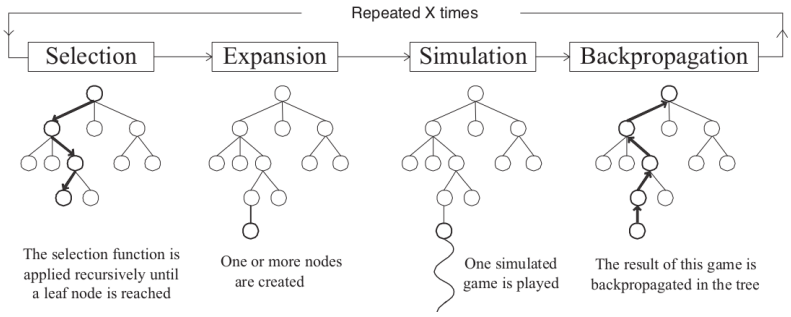


FIGURE 3.6: The four steps of Monte Carlo Tree Search [Chaslot et al., 2008]. First, the selection function is applied recursively until a leaf node is reached. Second, one or several nodes are created. Third, one simulated game is played until a winner is designed. Fourth, the result (win or lose) is backpropagated to node weights.

efficiently the reality (see Fig. 3.6). Sampling is a statistical method that consists in taking only a subset of a population to characterize this population. Sampling is efficient in games such as Backgammon, Poker, Scrabble, Go and even for real-time strategy games. At each iteration, it apply a selection function until a leaf node (which represents an action) is reached. One or several successor actions of this leaf node are added to the tree for future evaluation. Then it improves the current evaluation of the selected node by developing a complete playout (in games, a *playout* is a sequence of moves from an initial state to a final state in which one of the players *wins* the game). To develop a playout, the actions and effects at each stage are randomly choosen. When a playout is finished, the result is backpropagated. For example, a possible heuristic is to choose the action that has the maximum number of won developed playouts.

The dilemna, when exploring a Monte-Carlo tree is to balance exploration of the most promising actions, to gain reliable estimations of their expected gain (*exploitation*), and exploration of sub-optimal states that can be more successful in the long run. It is the *exploration-exploitation dilemna*. Upper-confidence bounds on trees (UCT) [Auer, 2003] are used to balance exploration and exploitation. In the selection phase, a is selected in state s if it maximizes

$$Q^{\otimes}_{UCT}(s,a) = Q_{UCT}(s,a) + c\sqrt{\frac{\log n(s)}{n(s,a)}}$$

with

- $Q_{UCT}(s,a)$ the mean utility yet obtained by selecting a in state s;
- c a parameter, enabling to balance exploration and exploitation;
- $n(s)$ the number of visits to state s in the previous iterations;
- $n(s,a)$ the number of selections of action a in state s in the previous iterations.

The first term favors promising actions and the second term favors actions that are not yet explored. In [Bjarnason et al., 2009], authors combine HOP with UCT and obtain great performances on a card game called *Klondike Solitaire*. Contrarily to MCTS, HOP does not maintain a tree and only tries to evaluate the next action to perform.

The idea of sampling may be interesting for our problems as their search spaces are very large. It would allow to reduce the number of search steps and still to take uncertainty into account.

3.3 Managing time and resources under uncertainty

We have seen that several techniques allow planning under uncertainty. Our problems have planning features but also scheduling features. In this section, we address scheduling under uncertainty. The computational effort for solving scheduling problems focuses on allocating time and resources to tasks. Managing uncertainty in scheduling problems is very important in applications because, for example, tasks such as manufacturing have rarely fixed and known durations, and external events may disturb the initial schedule. It is then necessary to have strategies to ensure that execution will not fail.

3.3.1 Job-Shop scheduling

In this section, we look at pure scheduling problems. Details on these problems can be found in [Pinedo, 2012]. The Job-Shop Scheduling Problem (JSSP) consists in scheduling each job J_i in a set $J = \{J_1, ..., J_n\}$ onto $M = \{M_1, ..., M_m\}$ identical machines while minimizing the total *makespan*, that is the total length of the schedule (see an example on Fig. 3.7). Each job is made of an ordered sequence of tasks. Each task must be processed during an uninterrupted period on a machine. The original problem is an online problem where jobs arrive one at a time and must be dispatched in an industrial manufacturing context. In the JSSP the *route* of job is fixed. It means that an order for tasks to be processed on machines is given as input. This order is not given in the *Open-Shop* version of the problem. The *Flexible Job-Shop* (FJSSP) version is even less restrictive as it allows a job to be processed by any machine of a given set. FJSSP is close to our data download problem where files from observations are stored onto memory banks and have to be scheduled onto identical communication channels. Finally, note that there exists a stochastic version of the problem [Mahdavi et al., 2010] which can take into account machine failures and stochastic task durations.

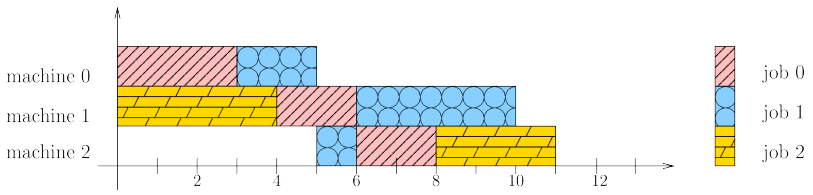


Figure 3.7: Solution to a Job-Shop problem with 3 machines and 3 jobs.

3.3.2 Resource-Constrained Project Scheduling Problem

The Resource-Constrained Project Scheduling Problem (RCPSP/max) [Bartusch et al., 1988] is a classic NP-hard problem that has several real-world applications in logistics and project scheduling. It is more general than the job-shop scheduling

problems. The deterministic version of RCPSP/max consists in scheduling n activities $\{a_1, a_2, ..., a_n\}$ where each activity a_i has a duration d_i. Tasks are executed without preemption, that is a task cannot be interrupted.

In RCPSP/max, there are two types of constraints :

- temporal constraints between task start dates : there are minimum and maximum time lags between two tasks a_i and a_j, that is constraints such as $st(a_j) - st(a_i) \geq T_{ij}^{min}$ and $st(a_j) - st(a_i) \leq T_{ij}^{max}$; a schedule is temporal-feasible if temporal constraints are enforced;
- resource constraints: resources $k \in \{1, ..., K\}$ are *renewable* and have a capacity C_k. Each task a_i requires r_{ik} units of resource of type k. If $A(t) = \{i \in \{1, 2...n\} | st(a_i) \leq t < et(a_i)\}$ is the set of active tasks at time t, it is said that a solution is resource-feasible if the consumption of any resource k does not exceeds its capacity C_k, that is if $\sum_{i \in A(t)} r_{ik} \leq C_k$.

A solution to an RCPSP/max is an assignment of start dates for all activities, that is a vector $ss = (st(a_1), ..., st(a_n))$ where $st(a_i)$ (resp. $et(a_i)$) represents the start date of task a_i (resp. the end date of task a_i). Because durations are deterministic, we have $st(a_i) + d_i = et(a_i)$. The schedule *makespan* is $st(a_{n+1}) = \max_{i=1,...,n} et(a_i)$. A solution is valid if it is resource-feasible and temporal-feasible. The usual objective in RCPSP/max is to find a valid solution with a minimal makespan. An example can be seen on Fig. 3.8.

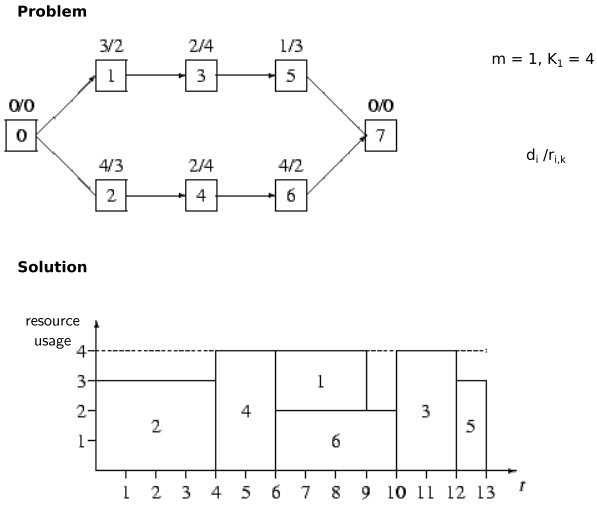


FIGURE 3.8: Example of an RCPSP/max with 6 tasks and 1 resource (top) and a solution (bottom). With each task i are associated a duration d_i and the number of required resource r_{ik} . Arrows are precedence constraints between tasks. A solution is represented as a Gantt diagram in which each task is a rectangle (duration $\times$ quantity of resource used).

Coming back to uncertainty, there exists a variant where task durations are stochastic (RCPSP/max with durational uncertainty). These task durations can be expressed as the sum of their mean value and deviation (or with an upper-bound and

lower-bound [Sotskov et al., 1998]): $\tilde{d}_i = d_i^0 + \tilde{z}_i$ with d_i^0 the mean of d_i and $\tilde{z}_i$ the deviation with an expected value of 0 and a standard deviation of ω. It is assumed that all variables $\tilde{z}_i$ are independent. Although *start-to-start* constraints are still deterministic, other constraints become stochastic. For example the *start-to-end* constraint $st(a_j) - et(a_i) \leq \Delta$ that constrains the start of an activity regarding the end of another activity becomes $st(a_j) - st(a_i) \leq \Delta + \tilde{d}_i$. The resulting makespan is stochastic as well. One solution to this variant is a policy defining tasks to launch regarding the actual durations. Such policies are compared in [Stork, 2001]. In general, the objective is to build high-quality robust plans with respect to a performance criterion such as maximizing *stability* or minimizing *failures* at execution.

In [Fu et al., 2012], the objective is to build a *Partial Order Schedule* policy while minimizing a *robust makespan* using local search. Makespan d_{rm} is said to be robust if for a certain acceptable level of risk $0 < \epsilon \leq 1$, all plans built with the policy have a $1 - \epsilon$ probability of ending before d_{rm}.

The RCPSP/max problem, together with the job-shop scheduling problem, is close to a part of our data download problem where files have to be scheduled onto channels and memory banks. Indeed, observations may be seen as jobs as they are made of several files. But this framework does not include all constraints of the problem, such as assignments restrictions (some stations are not allowed to some of the users), or time-dependent aspects. Also the makespan criterion is not sufficient.

3.3.3 Simple Temporal Networks

Although there are several forms of flexibility, temporal flexibility is particularly important when, in planning and scheduling, durations of tasks are uncertain. Temporal flexibility may provide the necessary adjustments for a plan or a schedule to remain feasible. In this section, we describe a classical way of representing temporal flexibility. Before speaking about *Simple Temporal Networks*, let first describe the *Simple Temporal Problem* (STP). The latter is a constraint satisfaction problem which contains a set of temporal variables $\{X_1, ..., X_n\}$ with domains $[lb_i, up_i]$ and a set of constraints $a_{ij} \leq X_j - X_i \leq b_{ij}$. A solution is a tuple $(x_1, ..., x_n)$ such that $x_i \in [lb_i, up_i]$ and all constraints $a_{ij} \leq x_j - x_i \leq b_{ij}$ are satisfied. Constraint $a_{ij} \leq X_j - X_i \leq b_{ij}$ may be rewritten as: $\{X_j - X_i \leq b_{ij}\}$ and $\{X_i - X_j \leq -a_{ij}\}$. Solving an STP then amounts to solving a set of linear inequalities, which is a well known problem in operations research (solvable with the simplex method for example). An STP can also be transformed into a shortest-path problem in a graph and shortest path algorithms can be applied to solve it in a much more efficient way.

An STN can represent temporal constraints of an STP as a distance graph $G(V, E)$ where nodes in V represents temporal variables $\{X_1, ..., X_n\}$ (to which we add an origin X_0) and edges in E represent distances between these variables. Then for each inequality $\{X_i - X_j \leq w_{ij}\}$, there is an edge (i, j) with a weight w_{ij}. This distance graph has interesting properties. The STN is *consistent* if its associated distance graph does not have any negative cycle. Moreover, if d_{0i} is the length of the shortest path from origin X_0 to node i and d_{i0} is the length of the shortest path from node i to origin, then the set of possible values for X_i is $[-d_{i0}, d_{0i}]$. When a constraint is updated or added, modifications can be efficiently propagated through the graph. An STN allows to represent temporally flexible plans, that is a set of plans not completely instantiated (flexibility measures have been proposed in [Wilson et al., 2014] for example).

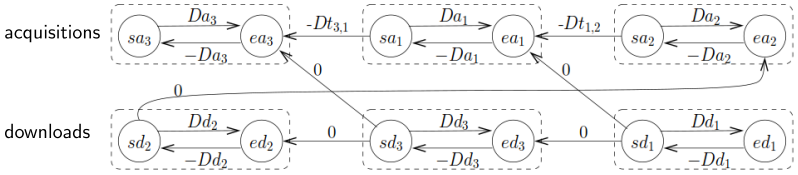


FIGURE 3.9: An example of distance graph associated to an STN.

To illustrate STNs; we consider the example which is depicted in Fig.3.9 (from [Pralet and Verfaillie, 2012]). In this example, there are 3 acquisitions a_1, a_2, a_3 that must be realized in the following order $a_3 \rightarrow a_2 \rightarrow a_1$. For each $i \in [\![1;3]\!]$, $Tmin_i$ and $Tmax_i$ are earliest start dates and latest end dates of a_i and Da_i is the duration of acquisition a_i. The minimum transition duration between the end of a_3 and the start of a_1, and between the end of a_1 and the start of a_2 are denoted $Dt_{3,1}$ and $Dt_{1,2}$ respectively. In the real problem, these durations are not constant and vary with start/end times of activities. We consider two reception station visibility windows $w_1 = [Ts_1, Te_1]$ and $w2 = [Ts_2, Te_2]$. The satellite must download a_2 then a_3 in window w_1, then a_1 in w_2. Download duration of a_i is denoted Dd_i.

This problem can be modeled as an STN including for each acquisition $a_i (i \in [\![1;3]\!])$, (a) two variables sa_i and ea_i, the start and end of acquisition a_i, with domains $d(sa_i) = d(ea_i) = [Tmin_i, Tmax_i]$; (b) two variables sd_i and ed_i, the start and end dates of an acquisition download, with domains $[Ts_1, Te_1]$ for $i = 2, 3$ and $[Ts_2, Te_2]$ for $i = 1$. This STN contains constraints given in Eq. 3.1 to 3.4.

Equation 3.1 define the duration of observations and downloads, Equation 3.2 imposes minimum transition times between acquisitions, Equation 3.3 is a non-overlapping constraint between downloads, and Equation 3.4 enforces that an acquisition cannot be downloaded before being finished.

$$\forall i \in [\![1;3]\!], (ea_i - sa_i = Da_i) \wedge (ed_i - sd_i = Dd_i) \tag{3.1}$$
$$(sa_1 - ea_3 \geq Dt_{3,1}) \wedge (sa_2 - ea_1 \geq Dt_{1,2}) \tag{3.2}$$
$$(sd_3 - ed_2 \geq 0) \wedge (sd_1 - ed_3 \geq 0) \tag{3.3}$$
$$\forall i \in [\![1;3]\!], sd_i - ea_i \geq 0 \tag{3.4}$$

There exist many variations of STN, for instance to deal with disjunctive constraints [Stergiou and Koubarakis, 2000]. Considerable work has been done to allow STNs and its variants to be used as a support for planning and execution. Simple temporal networks with uncertainty (STNU) [Vidal and Ghallab, 1996] allow to deal with uncontrollable durations between events. For that, a set of contingent links is added to the STN definition. A contingent link (x_i, l, u, x_j) induces an uncontrollable bounded duration in $[l, u]$ between the two temporal variables x_i and x_j. The problem is to figure out if an STNU can be executed without becoming inconsistent. In other words, one has to find how to execute the non-contingent variables, which are not affected by a contingent link, such that temporal constraints are satisfied. Several levels of controllability have been developed. *Dynamic controllability* (DC) is certainly the most important one. An STNU is said to be dynamically controllable

if there is a *strategy* for setting the non-contingent temporal variables such that all of the constraints in the network will be satisfied, no matter how the durations of the contingent links turn out. This property can be checked in polynomial time [Hunsberger, 2014].

A solution to our data download problem may be modeled as an STNU with file download durations being the contingent links. Enforcing DC would mean setting maximum durations (and then maximum volumes) for all activities as it is already done in the current pure ground approach. Also, it would lack onboard flexibility as all resource assignments would have to be done on the ground. Executing the STNU may move forward downloads if volumes are lower than maximum but this strategy would be equivalent to a Partial Order Schedule as it would have the form of a directed acyclic graph. This is why STNUs would not be interesting for the data download problem.

But, we can look at another aspect of the data download problem. Because data link rate is not constant, the duration of a download depends on its starting time. In the same manner, because the satellite perform attitudes moves, the duration of an antenna transition between two stations depends on its starting time. These two aspects may be represented in a *Time-dependent Simple Temporal Network* (TSTN) [Pralet and Verfaillie, 2013b]. The latter framework proposes to deal with time-dependent constraints. In this framework, constraints between temporal variables are not necessarily constant anymore but may depend on their assignments. In Chap. 6, we will use TSTN to deal with several aspects of the data download problem. Finally, we note that there is no TSTNU framework which supports both time-dependent and uncontrollable durations.

3.4 Combinatorial optimization under uncertainty

We now look at more general frameworks to tackle combinatorial optimization problems in an uncertain and dynamic environment.

3.4.1 Uncertainty in constraint programming

Constraint Programming (CP) is a declarative programming paradigm [Rossi et al., 2006]. In CP, variables and constraints linking these variables define the properties of a solution. Several types of constraints (linear, non-linear...) and domains (such as boolean, integer) are authorized, which allows to model a great number of problems. Decision variables define what can be set. A criterion may be defined and the problem goes from a satisfaction problem to an optimization problem. Then, generic methods are used to search in the space of assignments. One interesting idea in constraint programming is to specialize the processing of some constraints based on efficient methods to prune more efficiently the search space. For example, there exists a global constraint $\texttt{allDifferent}(x_1, ..., x_n)$ that takes n variables as an input and enforces that $x_1, ..., x_n$ take different values. If $n = 3$ and if the domains of x_1, x_2, x_3 are equal to $\{0, 1, 2\}$, this constraint forbids assignments $(1, 1, 1)$ or $(2, 2, 1)$ and allows assignments $(2, 1, 3)$ or $(1, 2, 3)$. Pruning domains with this constraint is modeled as a matching in a bipartite graph. Properties of the latter problem is then used to derive an efficient *filtering algorithm* for the constraint. An example of constraint satisfaction problem is given in Fig. 3.10.

Extensions of CP have been designed to integrate uncertainty. When some variables follow a probability distribution, we speak of a *Stochastic Constraint Satisfaction*

$$var = [S, E, N, D, M, O, R, Y] \tag{3.5}$$
$$\forall i, var(i) \in [\![0; 9]\!] \tag{3.6}$$
$$S \neq 0 \tag{3.7}$$
$$M \neq 0 \tag{3.8}$$
$$\text{allDifferent}(var) \tag{3.9}$$
$$\begin{aligned} &1000 * S + 100 * E + 10 * N + D \\ &+1000 * M + 100 * O + 10 * R + E \\ &\neq 10000 * M + 1000 * O + 100 * N + 10 * E + Y \end{aligned} \tag{3.10}$$

FIGURE 3.10: Example of constraint satisfaction problem. The SEND+MORE=MONEY problem consists in finding distinct digit values for letters $\{S, E, N, D, M, O, R, Y\}$ such that the equation holds.

Problem (SCSP) [Walsh, 2002]. Then, several decision stages may be considered. In the one-stage version, decision variables are assigned before the realization of some of stochastic variables. This first version is rather close to the *Probabilistic Constraint Satisfaction* framework [Fargier and Lang, 1993] where some variables are stochastic. In the two-stage version of SCSP, there is a set of *recourse* variables, that is a set of variables that are not assigned in the first phase but only after some stochastic variables are realized and observed, and before some other stochastic variables are realized. The solution to such an SCSP defines how to set second-stage variables according to the realization of stochastic variables in the first stage. It is a *policy*. The objective is to assign values to decision variables such that constraints are enforced with a probability greater than or equal to θ.

In *Mixed Constraint Satisfaction* (MCSP) [Fargier et al., 1995], variables are partitioned between controllable and uncontrollable variables. Uncontrollable variables have each a set of possible values and are not associated with probabilities. A solution to the MCSP is a conditional strategy for each possible assignment of uncontrollable variables in case of full observability, and a unique assignment of controllable variables covering the maximum number of worlds in the case of partial observability.

Multi-stage stochastic constraint programming may be an adapted framework for the data download problem where data volumes are stochastic. In this problem, there are several phases of decision-making, on the ground and on board. Between these phases, volumes are realized and observed. Then, there may be one decision stage for each decision-making phase. In addition, all kinds of non-linear constraints may be modeled by constraint programming such as assignment, or planning and scheduling constraints [Barták and Salido, 2011], even if there are not all yet implemented in the solvers (such as a recent global constraint for managing time-dependent transitions [Aguiar Melgarejo et al., 2015] that we need for managing satellite attitude transitions). We use CP to solve a deterministic and simplified version of the data download problem in Chap. 5.

3.4.2 Online stochastic combinatorial optimization

Online Stochastic Combinatorial Optimization (OSCO) is an heuristic online decision-making framework for problems where data is not available a priori but revealed during execution [Hentenryck and Bent, 2009]. Targeted problems are for example those with *requests* that must be dispatched. The problem consists in choosing which request to deal with to maximize utility. OSCO inherits of both online methods, because the objective is to make a decision in a limited time and because the uncertainty lies only on the future, and stochastic optimization because sampling is used for decision-making.

Generic offline problem The framework assumes a discrete model of time and considers a time horizon $H = [1, h]$. The input is a sequence $\langle R_1, \ldots, R_h \rangle$ where each R_i is a set of requests. A solution to the offline problem serves a request $r \in R_1 \cup \ldots \cup R_t$ at each time $t \in H$ and is represented as a sequence $S = \langle r_1, \ldots, r_h \rangle$. This sequence must satisfy the problem-specific constraints C ($C(S)$ is true in this case) and as each request $r \in R$ is associated with a reward $w(r) \geq 0$, $w(S)$ is the reward of the sequence. It is assumed that an algorithm $\mathcal{O}$ able to optimally solve this offline problem is available.

Generic online problem In the online version of the problem, the set of requests $\langle R_1, \ldots, R_h \rangle$ are not available initially but are revealed progressively at each time step t. At step t, the online algorithm has at its disposal the sequences $\langle R_1, \ldots, R_t \rangle$ and the already served requests $\langle s_1, \ldots, s_{t-1} \rangle$. The goal is to decide which request to serve. In this online version, the sets of requests $\langle R_1, \ldots, R_h \rangle$ are the realization of random variables $\langle \xi_1, \ldots, \xi_h \rangle$ whose distribution is specified by $\mathcal{I}$.

To solve the online stochastic problem, the online algorithms has two back-boxes:

1. an offline optimization algorithm $\mathcal{O}$ which, given a set of past decisions and sequence of requests, returns an optimal solution to the problem;
2. a conditional sampling procedure Sample which given a time t and a sampling horizon Δ, returns a sequence of

$$\textsc{Sample}(t, \Delta) = \langle R_{t+1}, \ldots, R_{t+\Delta} \rangle \tag{3.11}$$

realization of the random variables obtained by sampling distribution $\mathcal{I}$.

Algorithm 1: OnlineOptimization(H)

```
S_0 = ⟨⟩;
for t ∈ H do
    do s_t = ChooseRequest(S_{t-1}, ⟨R_1, ..., R_t⟩);
    S_t → S_{t-1} : s_t;
```

The generic online algorithm is shown in Algorithm 1. Different versions of ChooseRequest exist as maximizing expectation or hedging against the worst-case scenario. In the following, we only present one version of this procedure in Algorithm 2, the Online Expectation (ChooseRequest-E) version, which is appropriate when time constraints are loose. At each time step, it evaluates all possible decisions

Algorithm 2: ChooseRequest-E($S_{t-1}, \langle R_1, \ldots, R_t \rangle$)

```
1 F ←Feasible(S_{t-1}, ⟨R_1, ..., R_t⟩);
2 for r ∈ F do
3 |   f(r) ← 0;
4 for i ← 1..T/|F| do
5 |   A ← ⟨R_1, ..., R_t⟩ :: Sample(t, Δ);
6 |   for r ∈ F do
7 |   |   f(r) ← f(r) + w(O(S_{t-1} : r, A));
8 return arg max(r ∈ F) f(r)
```

with respect to all scenarios and thus the number of optimizations $\mathcal{T}$ should be large enough in order to produce high-quality results.

Line 1 computes the requests which can be served at time t . Lines 2–3 initialize the evaluation function $f(r)$ for each feasible request r. The algorithm then generates a number of samples for future requests (lines 5 and 6). For each such sample, it computes a scenario A consisting of all available and sampled requests (line 6). The algorithm then considers each feasible request r at time t (line 7) and applies the optimal offline algorithm with the concatenation $S_{t-1} : r$ indicating that request r is scheduled at time t (line 8). The evaluation of request r is updated in line 8 by incrementing it with its reward and the value of the optimal offline solution. All scenarios are evaluated for all available requests and the algorithm then returns the request $r \in F$ with the highest evaluation (line 11).

This algorithm takes a sequence of decisions which can be thought of as an approximation to a multi-stage stochastic combinatorial program where non-anticipatory constraints are relaxed. This relaxation can be viewed as a two-stage stochastic optimization approximated by an exterior sampling method that evaluates each decision with respect to all generated scenarios. This evaluation uses the offline optimization algorithm as a black-box. This framework provides interesting theoretical properties (under some assumptions) such as an expected constant loss compared to the offline optimal solution.

OSCO has been applied in [Awasthi and Sandholm, 2009] in which authors solve the problem of finding compatible pairs of donors and patients for kidney transplant in the United States. This high-dimensional NP-hard problem is then solved by transforming it into a online problem such as presented earlier.

In the data download problem, volumes of observations are revealed during execution but unfortunately, downloading one observation may have an impact on future possible downloads because there are assignment constraints that prevent some observations to be downloaded onto some ground stations. In the observation planning problem, observations may be failed because of clouds obstructing the target. Weather forecasts may be available for estimating the probability of this failure in the future. OSCO may then be an interesting framework for online decision-making about observation planning. Unfortunately, onboard sampling for this problem is not considered because of onboard computing limitations.

3.5 Planning and scheduling in the space domain

We have seen how to handle uncertainty for planning and scheduling in general and now we focus on how these problems are solved in the space domain.

Scheduling observations for an Earth-observing satellite is a complex problem : scientists from several domains need a lot of acquisitions; possible windows to acquire data are small; ground reception station must communicate with a lot of satellite at the same time. In that case, the planner may not be able to make the best decision and human operators often need to validate the automatically generated plans. Indeed, while used by the planning system, information from different sources are aggregated and decision-making may sometimes be hardly understandable by humans. Bringing automated tools to operators is difficult and interactive software allow the users to modify information after visualizing a plan as in the *Mixed-Initiative Planning* approach [Ai-Chang et al., 2004] which is used for martians rovers Spirit and Opportunity.

Problems from the space domain feature specific constraints and large search spaces. Solving these problems often involves new methods. In [Woods et al., 2009b], planning and scheduling techniques are implemented onto a ExoMars prototype so it is able to perform autonomous science on Mars. In [Frank et al., 2001], authors present the satellite domain and implement an algorithm based on a heuristic tree exploring mechanism called *Heuristic Biased Stochastic Sampling* [Bresina, 1996]. This technique has originally been developed for scheduling observations on telescopes. Pemberton [Pemberton and Greenwald, 2002] proposes an algorithm combining separation and evaluation with a greedy search for a acquisition scheduling problem in which acquisitions have priorities and fixed durations. Wolfe and Sorensen [Wolfe and Sorensen, 2000] solve a harder problem in which each activity has an earliest start date, a latest end date, a minimum duration and a maximum duration (*window-constrained packing problem*). They compare two greedy algorithms with a genetic algorithm, which is proven to be the best.

Authors of [Lemaître et al., 2002] and [Verfaillie and Lemaître, 2001] are interested in scheduling observations on an agile Earth-observing satellite with constraints close to ours. They compare four approaches : a greedy algorithm, a dynamic programming strategy, constraint programming and local search. The latter presents the best performance.

In [Pralet and Verfaillie, 2008], the satellite Frankie of the Hotspot mission must detect ground phenomena such as volcanic eruptions or forest fires. Onboard planning is done with an anytime iterated stochastic greedy algorithm.

Space systems are expensive and critical resources that are used at their full potential. Users send a lot more observation requests than the system is able to process. That is why the resulting planning and scheduling resulting problems are said to be *oversubscribed*. Oversubscribed scheduling (OS) is a particular case of scheduling. In that class of problem, there are not enough resources to process all the tasks. The problem then becomes a lot more difficult since it is not only a scheduling problem, in which we must decide *when* tasks are processed and on which resources, but also a selection problem, in which we must select first the best subset of tasks to be processed. The problem of scheduling for the Air Force communication satellites is studied in [Barbulescu et al., 2006]. Notably, an analysis shows that search spaces of OS problems seem to have large plateaus. This result leads to design algorithms making large leaps in the search space to escape these plateaus. In [Globus et al., 2004b], authors had already glimpsed this result by comparing algorithms on a 2-satellite problem where hill climbing was much worse

than simulated annealing. Learning from this fact, we use a large neighborhood search to compute data download plans in Sect. 5.3.

We have seen that planning and scheduling under uncertainty can be tackled with various frameworks and methods. The challenge, in our case, is the size of the instances, and many of the methods are not scalable to the thousands of tasks that acquisition planning or data download scheduling problems include. In the next part, we will focus on the data download problem and provide a model, algorithms, and a flexible decision-making approach for this problem.

Part II

Building adaptable data download schedules

Introduction

Because of the use of more and more sophisticated compression algorithms, the amount of data that results from an observation and thus is recorded on board and must be downloaded to the ground is more and more unpredictable. It depends on the data that has been acquired. In the case of optical instruments, the presence of clouds over the observed area allows high compression rates and results in a low amount of data to be recorded and downloaded.

As satellites are not continuously accessible by a ground control station, generated volumes which are directly known on board are available on the ground only with some delay. This leads to think that decisions about downloads should be made on board just before any reception station visibility window with the exact knowledge of the volumes of the already recorded data. This approach has been explored in [Pralet et al., 2014c]. However, planning onboard is time-consuming. Moreover, the resulting plans are unpredictable. This is problematic especially for high-priority observations, because users who request an image may want to know when data will be downloaded. In such conditions, it is still possible to build data download plans on the ground with the assumption of maximum volumes (minimum default compression rate). The resulting plans are then always executable on board, but suboptimal due to an under-use of the station visibility windows.

In the work we present in this part, we want to introduce more autonomy on board satellites while keeping some predictability. The main idea is to share decision-making between ground and board to take advantage of the high computing power on the ground and of the low uncertainty on board.

First, we provide a model of the data download problem and analyze it. It is a complex scheduling problem with temporal and non-temporal constraints, close to RCPSP/max or flexible job-shop problems. We then design algorithms for the deterministic version of the problem. We finally propose a flexible decision-making scheme to deal with this problem.

This part of the work has been presented at the following international and national conferences : ICAPS'14 [Pralet et al., 2014c], ICAPS'15 [Maillard et al., 2015a], i-SAIRAS'14 [Maillard et al., 2014], JFPC'14 [Pralet et al., 2014a], ROADEF'14, and ROADEF'15. A paper concerning flexible scheduling for the data download problem has been accepted for publication in the AIAA Journal of Aerospace Information Systems. A paper developing theoretical aspects of the file scheduling subproblem is currently being reviewed for publication in Discrete Applied Mathematics.

As seen in Chap. 2, after the satellite has acquired data, it must download it during ground station visibility windows. In this chapter we formalize the problem of building an observation download schedule. Note that this problem involves real-life constraints that are harder to tackle than a theoretical combinatorial optimization problem. Finding an appropriate way to present these aspects in a complete model is a challenge. First, we give an informal description of the problem. Then, we give a formal model including input data, decision variables, constraints, and an optimization criterion. Finally, we make some comments on the problem nature and point some related works. In the remainder of this manuscript, the words *observation* and *acquisition* on one hand, and *download window* and *visibility window* on the other hand, refer to the same concept.

4.1 Informal description

To build a download schedule, decisions must be made at three levels:

1. observation download selection, assignment, and scheduling: this consists in deciding which observations will be downloaded and in building a sequence of observation downloads, each download being made of an observation o and of a download window w assigned to o; this problem is close to a Multi-knapsack problem [Kellerer et al., 2004] where objects are observations and sacks are windows, but first sacks are preferred as freshness of information has to be maximized, and conflicts may exist between sacks (overlapping windows);
2. file download assignment and scheduling: each observation generates several files; once an observation has been assigned to a download window, this subproblem consists in assigning each file of this observation to be downloaded an emission channel, and in building on each channel and on each memory bank a sequence of file downloads; in each download window, the scheduling problem is close to a *Flexible Open-shop Scheduling problem* [Pinedo, 2012] with two types of unsharable resources: channels and memory banks; each file download requires one resource of each type, but the choice of the channel is free, whereas the bank is pre-allocated; the file download order is also free;
3. time management: this consists either in computing earliest and latest dates for all the events, or in setting their precise dates; the events are the start and

end of file downloads, of antenna movements, and of key change table resettings; the resulting temporal problem has the form of a Simple Temporal Network (see Sect. 3.3.3); the only exceptions are the constraints of download duration and of transition between download windows which are time dependent (download and transition durations depend on the time at which they start); the result is a Time-dependent STN, for which STN techniques can be extended and polynomial algorithms can decide on consistency/inconsistency and compute the earliest/latest times for all the temporal variables.

These decisions must satisfy all the constraints that will be introduced in Sect. 4.2.3. Many of these constraints come from the limitations of the physical system: for example, the download durations. Some of them result from user requirements: for example, the fact that all the files generated by an observation must be downloaded towards one station s and within one of its associated download windows. Another example is the notion of *commitment* that can be associated to some observations, enforcing that some decisions made on the ground must not be changed on board. More details on this aspect are given in Sect. 6.1.3. Many of these constraints are temporal: for example, the absence of overlapping of file downloads on every emission channel and on every memory bank. Some are non temporal: for example, the fact that any observation o required by a user u must be downloaded towards a station s allowed by u.

Decisions made must also optimize as much as possible a global criterion which takes into account the downloaded observations, their priorities, the so-called information age (temporal distance between an observation and the data delivery to the user who required it), and the fair sharing of the system between its users.

To illustrate the data download scheduling problem we have to manage, we use a small illustrative example which will be systematically reused throughout this part. This example involves two overlapping download windows 1 and 2 respectively associated with two ground reception stations 1 and 2, eight observations to be downloaded from A to F, two priority levels 1 and 2, three users numbered from 1 to 3, five memory banks numbered from 1 to 5, and three emission channels numbered from 1 to 3. Table 4.1 shows the priorities and the users associated with each observation, as well as the files generated by each observation and the memory banks on which these files are recorded. For example, observation A is of priority 2 (low priority), is required by user 3, and generates three files that are recorded on memory banks 1, 2, and 4. Moreover, we assume that user 2 allows only station 2 to be used to download its data and that all the observations are finished before the beginning of windows 1 and 2, except observation B which will finish during window 1.

Observations	A	B	C	D	E	F	G	H
Priorities	2	1	1	2	2	2	1	2
Users	3	3	1	1	3	2	2	1
Bank 1	x	x	x		x	x		x
Bank 2	x	x	x				x	
Bank 3		x	x	x	x		x	
Bank 4	x	x	x	x			x	x
Bank 5		x	x	x	x	x		x

Table 4.1: Priorities, users, and generated files associated with each observation in the illustrative example.

Fig. 4.1 shows a possible download schedule and all its associated temporal constraints. In this schedule, the sequence of observation downloads is C, H, B, G, E with observations C, H, and B downloaded in window 1 and observations G and E downloaded in window 2. Observations A, D, and F are not downloaded (they will be downloaded later). If we go down at the file level, we see on which channel and in which order the several files generated by an observation are downloaded: for example for observation C, file recorded in bank 4 ($(C, 4)$) on channel 1, files recorded in banks 2 and 5 ($(C, 2)$ and $(C, 5)$) in sequence on channel 2, and files recorded in banks 1 and 3 ($(C, 1)$ and $(C, 3)$) in sequence on channel 3.

All the temporal constraints that result from this schedule are represented in Fig. 4.1:

1. channel precedence constraints: these constraints express the fact there must be no overlapping between file downloads on the same channel; for example, the download of $(C, 2)$ must precede the download of $(C, 5)$ on channel 2;
2. memory bank precedence constraints: they express the fact there must be no overlapping between file downloads from the same memory bank; for example, the download of $(H, 4)$ must precede the download of $(B, 4)$ because both are recorded on the same bank;
3. acquisition precedence constraints: they express the fact that no observation can be downloaded before being finished; for example, the download of $(B, 4)$ cannot start before t_1 which is assumed to be the end time of observation B (start and end of observations are provided by the observation schedule);
4. window precedence constraints: they express the fact that all the file downloads associated with an observation o must be included in the window assigned to o; for example, the download of $(G, 3)$ must start after the start time s_2 of window 2 and the download of $(E, 3)$ must end before the end time e_2 of the same window;
5. antenna move transition constraints: they express the fact that moving the mobile emission antenna from a station to another one takes some time during which no download is possible; for example, a minimum distance is required between the end of download of $(B, 5)$ and the start of download of $(G, 2)$; such a distance depends on both stations and on the time at which the transition starts (time-dependent transition time);
6. table resetting transition constraints: they express the fact that resetting the key change table used for encrypting user data takes some time during which no download is possible; for example, because we assume that we take advantage of the antenna movement to reset the key change table, another minimum constant distance is required between the end of download of $(B, 5)$ and the start of download of $(G, 2)$;
7. file download duration constraints: they are represented in the figure by the lengths of the rectangles associated with each file download; because the download rate is not constant over a download window and depends on the satellite-station distance, download durations depend on the time at which they start (time-dependent duration).

Fig. 4.2 shows the associated earliest schedule where all the downloads are scheduled as soon as possible.

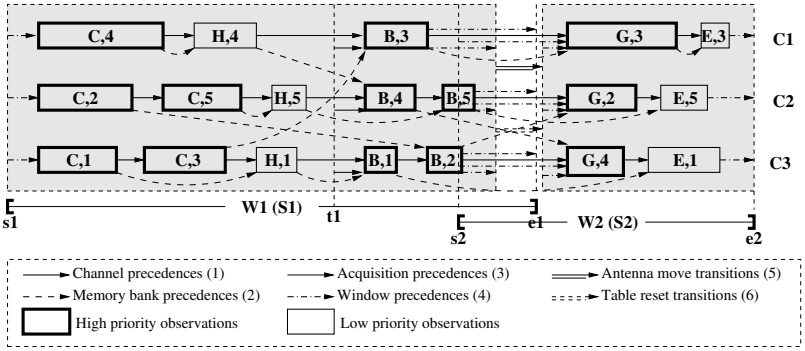


Figure 4.1: A download schedule and its temporal constraints.

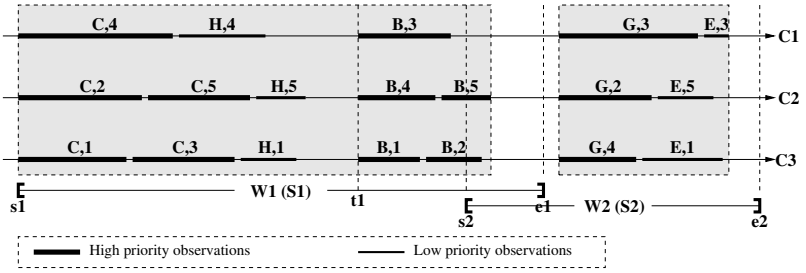


Figure 4.2: The associated earliest schedule.

4.2 Formal model of the data download scheduling problem

In this section, we show how the download scheduling problem can be modeled as a constraint optimization problem with its *data*, its decision *variables*, its *constraints* to be satisfied, and its *criterion* to be optimized.

We use the following conventions: constants and functions (data) appear in bold, starting with an upper case letter; variables and expressions appear in bold, but starting with a lower case letter; indices appear in italic.

4.2.1 Problem data

Problem data is made of *static* data which do not vary during the satellite lifetime and of *dynamic* data which do vary according to the satellite state and to the current objectives over the scheduling horizon.

4.2.1.1 Static data

Mass memory:

– number $\mathbf{Nm}$ of memory banks (positive integer).

- capacity $\mathbf{Cm}$ of each memory bank (positive integer).

Emission antenna:
- number $\mathbf{Nc}$ of emission channels (positive integer);
- angular velocity $\mathbf{Sa}$ of the orientable antenna (positive float).

Data encryption:
- maximum number $\mathbf{Nkmax}$ of changes per channel that can be recorded in the key change table (positive integer);
- duration $\mathbf{DuKct}$ required to reset the key change table (positive float).

Ground reception stations:
- number $\mathbf{Ns}$ of ground reception stations (positive integer).

Users:
- number $\mathbf{Nu}$ of users of the system (positive integer);
- for each user $u \in [\![1, \mathbf{Nu}]\!]$ and each reception station $s \in [\![1, \mathbf{Ns}]\!]$,
  - a boolean $\mathbf{All}_{s,u}$ indicating whether or not data required by user u may be downloaded towards station s;
  - a data transfer rate $\mathbf{Gtr}_{s,u}$ on the ground from s to the processing center of u (positive float).

Priorities:
- number $\mathbf{Np}$ of priority levels (positive integer).

4.2.1.2 Dynamic data

Scheduling horizon:
- $\mathbf{Ts}$: start time of the download scheduling horizon (positive float);
- $\mathbf{Te}$: end time of the download scheduling horizon (positive float).

Download windows:
- number $\mathbf{Nw}$ of (effective) download windows over the scheduling horizon (positive integer); these windows are computed from the ground reception station visibility windows and from the observation schedule and the associated satellite attitude trajectory;
- for each download window $w \in [\![1, \mathbf{Nw}]\!]$:
  - a start time $\mathbf{Sw}_w$ (positive float);
  - an end time $\mathbf{Ew}_w$ (positive float);
  - an associated reception station $\mathbf{St}_w$ (integer in $[\![1, \mathbf{Ns}]\!]$);
  - a function $\mathbf{Dr}_w : [\mathbf{Sw}_w, \mathbf{Ew}_w] \rightarrow \mathbb{R}^+$ which gives the download rate within download window w as a function of time; function $\mathbf{Dr}_w$ is assumed to be piecewise constant.

Observations:
- number $\mathbf{No}$ of observations that can be downloaded over the scheduling horizon (positive integer): observations that have been already performed but not downloaded yet and observations that have been not performed yet, but are scheduled (present in the observation schedule);

- for each observation $o \in [\![1, \mathbf{No}]\!]$:
  - a start time of observation $\mathbf{So}_o$ (positive float);
  - an end time of observation $\mathbf{Eo}_o$ (positive float);
  - the user $\mathbf{U}_o$ who requested o (integer in $[\![1, \mathbf{Nu}]\!]$);
  - a priority level $\mathbf{P}_o$ (integer in $[\![1, \mathbf{Np}]\!]$);
  - a type of commitment $\mathbf{Ct}_o$, only for high priority observations (integer in $[\![1, 2]\!]$);
  - a (normalized) weight $\mathbf{W}_o$ (float in $]0, 1]$) used to compare observations at a given priority level;
  - a function $\mathbf{Fresh}_o : [0, +\infty[\rightarrow]0, 1]$ which gives the freshness of o as a function of the temporal distance δ between the end time $\mathbf{Eo}_o$ of observation o and its data delivery time to the user $\mathbf{U}_o$ who required it; we assume the function $\mathbf{Fresh}_o$ defined by $\mathbf{Fresh}_o(\delta) = 2^{-\delta/\mathbf{Ht}_o}$ with $\mathbf{Ht}_o$ a parameter which depends on o; this implies a freshness which is maximum in case of immediate delivery ($\mathbf{Fresh}_o(0) = 1$), decreases as a function of δ, and is divided by 2 every $\mathbf{Ht}_o$ units of time;
  - a set $\mathbf{Sf}_o$ of generated files;
- for each observation $o \in [\![1..\mathbf{No}]\!]$ and each file $f \in \mathbf{Sf}_o$:
  - a memory bank $\mathbf{M}_{o,f}$ onto which f is memorized (integer in $[\![1, \mathbf{Nm}]\!]$);
  - a volume $\mathbf{V}_{o,f}$ of f (positive float): real volume if known, expected or maximum volume otherwise.

4.2.2 Decision variables

The set of decision variables can be decomposed into two sets: the set of *non temporal* variables, which represent the assignment and scheduling decisions, and the set of *temporal* variables, which represent the times at which all the relevant events occur.

Non-temporal variables The non-temporal variables allow the assignments of observation downloads to download windows and of file downloads to channels, as well as the sequences of observation downloads and of file downloads on each channel and from each memory bank to be represented:

- for each observation $o \in [\![1, \mathbf{No}]\!]$, a download window $\mathbf{dw}_o \in [\![0, \mathbf{Nw}]\!]$ in which o is downloaded: 0 if o is not included in the download schedule;
- a sequence $\mathbf{dSeq} = [o_1, \ldots, o_n]$ defining a total order between observation downloads in the schedule; this sequence contains exactly all the observations $o \in [\![1, \mathbf{No}]\!]$ such that $\mathbf{dw}_o \neq 0$;
- for each observation $o \in [\![1, \mathbf{No}]\!]$ and each file $f \in \mathbf{Sf}_o$ generated by o, a channel $\mathbf{ch}_{o,f} \in [\![0, \mathbf{Nc}]\!]$ onto which file f is downloaded: 0 if o is not included in the download schedule;
- for each channel $c \in [\![1, \mathbf{Nc}]\!]$, a sequence $\mathbf{cSeq}_c = [(o_1, f_1), \ldots, (o_q, f_q)]$ defining a total order between file downloads on channel c; a pair (o, f) in the sequence represents the download of file f of observation o on channel c; each sequence $\mathbf{cSeq}_c$ contains exactly all the pairs (o, f) such that $o \in [\![1, \mathbf{No}]\!]$, $\mathbf{dw}_o \neq 0$, $f \in \mathbf{Sf}_o$, and $\mathbf{ch}_{o,f} = c$;
- for each memory bank $m \in [\![1, \mathbf{Nm}]\!]$, a sequence $\mathbf{mSeq}_m = [(o_1, f_1), \ldots, (o_r, f_r)]$ defining a total order between file downloads from memory bank m; a pair (o, f) in the sequence represents the reading of file f of observation o

Assignment of observation downloads to download windows:
$\mathbf{dw}_A = 0, \mathbf{dw}_B = 1, \mathbf{dw}_C = 1, \mathbf{dw}_D = 0, \mathbf{dw}_E = 2, \mathbf{dw}_F = 0, \mathbf{dw}_G = 2, \mathbf{dw}_H = 1$
Sequence of observation downloads:
$\mathbf{dSeq} = [C, H, B, G, E]$
Assignment of file downloads to channels:
$\mathbf{ch}_{A,1} = 0, \mathbf{ch}_{A,2} = 0, \mathbf{ch}_{A,4} = 0$
$\mathbf{ch}_{B,1} = 3, \mathbf{ch}_{B,2} = 3, \mathbf{ch}_{B,3} = 1, \mathbf{ch}_{B,4} = 2, \mathbf{ch}_{B,5} = 2$
$\mathbf{ch}_{C,1} = 3, \mathbf{ch}_{C,2} = 2, \mathbf{ch}_{C,3} = 3, \mathbf{ch}_{C,4} = 1, \mathbf{ch}_{C,5} = 2$
$\mathbf{ch}_{D,3} = 0, \mathbf{ch}_{D,4} = 0, \mathbf{ch}_{D,5} = 0$
$\mathbf{ch}_{E,1} = 3, \mathbf{ch}_{E,3} = 1, \mathbf{ch}_{E,5} = 2$
$\mathbf{ch}_{F,1} = 0, \mathbf{ch}_{F,5} = 0$
$\mathbf{ch}_{G,2} = 2, \mathbf{ch}_{G,3} = 1, \mathbf{ch}_{G,4} = 3$
$\mathbf{ch}_{H,1} = 3, \mathbf{ch}_{H,4} = 1, \mathbf{ch}_{H,5} = 2$
Sequences of file downloads on channels:
$\mathbf{cSeq}_1 = [(C,4), (H,4), (B,3), (G,3), (E,3)]$
$\mathbf{cSeq}_2 = [(C,2), (C,5), (H,5), (B,4), (B,5), (G,2), (E,5)]$
$\mathbf{cSeq}_3 = [(C,1), (C,3), (H,1), (B,1), (B,2), (G,4), (E,1)]$
Sequences of file downloads from memory banks:
$\mathbf{mSeq}_1 = [(C,1), (H,1), (B,1), (E,1)]$
$\mathbf{mSeq}_2 = [(C,2), (B,2), (G,2)]$
$\mathbf{mSeq}_3 = [(C,3), (B,3), (G,3), (E,3)]$
$\mathbf{mSeq}_4 = [(C,4), (H,4), (B,4), (G,4)]$
$\mathbf{mSeq}_5 = [(C,5), (H,5), (B,5), (E,5)]$
Number of key changes since the last key change table resetting:
$\mathbf{nkc}_A = 0, \mathbf{nkc}_B = 2, \mathbf{nkc}_C = 1, \mathbf{nkc}_D = 0, \mathbf{nkc}_E = 2, \mathbf{nkc}_F = 0, \mathbf{nkc}_G = 1, \mathbf{nkc}_H = 1$

Table 4.2: Assignment of the non-temporal variables associated with the download schedule of Fig. 4.2.

from memory bank m; each sequence $\mathbf{mSeq}_m$ contains exactly all pairs (o, f) such that $o \in [\![1, \mathbf{No}]\!]$, $\mathbf{dw}_o \neq 0$, $f \in \mathbf{Sf}_o$, and $\mathbf{M}_{o,f} = m$;
- for each observation $o \in [\![1, \mathbf{No}]\!]$, the number $\mathbf{nkc}_o \in [\![0, \mathbf{Nkmax}]\!]$ of key changes performed since the last key change table resetting, including the key change which may be required by o: 0 if o is not included in the download schedule.

Table 4.2 shows the assignment of the non-temporal variables associated with the download schedule of Fig. 4.2.

Temporal variables The temporal variables represent the start and end times of file downloads: for each observation $o \in [\![1, \mathbf{No}]\!]$ and each file $f \in \mathbf{Sf}_o$:

– a download start time $\mathbf{sdf}_{o,f} \in [\mathbf{Ts}, \mathbf{Te}]$;
– a download end time $\mathbf{edf}_{o,f} \in [\mathbf{Ts}, \mathbf{Te}]$.

To write more easily the temporal constraints, we add temporal variables which represent the start and end times of observation downloads: for each observation $o \in [\![1, \mathbf{No}]\!]$:
– a download start time $\mathbf{sdo}_o = \min_{f \in \mathbf{Sf}_o} \mathbf{sdf}_{o,f}$;
– a download end time $\mathbf{edo}_o = \max_{f \in \mathbf{Sf}_o} \mathbf{edf}_{o,f}$.

4.2.3 Constraints

The set of constraints can be decomposed into three sets: the set of *assignment* constraints which constrain the possible assignments of observation downloads to download windows and of file downloads to channels, the set of *scheduling* constraints which constrain the possible schedules of observation downloads and of file downloads on channels and from memory banks, and the set of *temporal* constraints which constrain the possible start and end times of file downloads.

Assignment constraints
– an observation must be downloaded towards an allowed station:

$$\forall o \in [\![1, \mathbf{No}]\!] : (\mathbf{dw}_o \neq 0) \rightarrow \mathbf{All}_{\mathbf{St}_{\mathbf{dw}_o}, \mathbf{U}_o} \tag{4.1}$$

– a channel must be assigned to each file generated by a downloaded observation:

$$\forall o \in [\![1, \mathbf{No}]\!], \forall f \in \mathbf{Sf}_o : (\mathbf{dw}_o \neq 0) \leftrightarrow (\mathbf{ch}_{o,f} \neq 0) \tag{4.2}$$

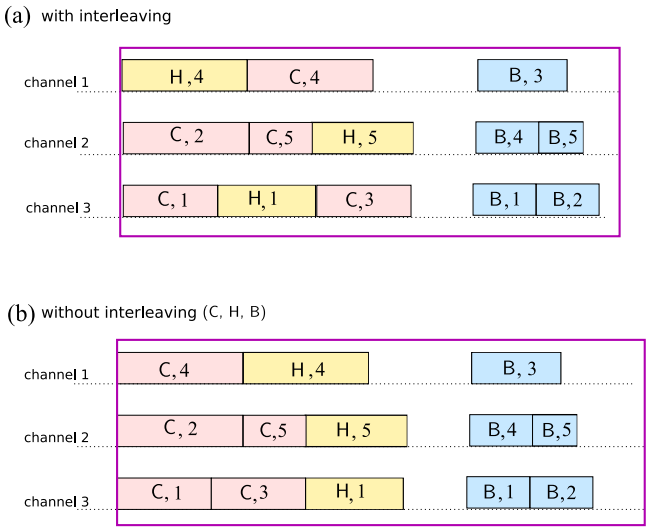


Figure 4.3: Sequence of downloads of three observations A, B, C. A, i is the file of observation A stored onto memory bank i.

Scheduling constraints File download scheduling on each channel and from each memory bank must be consistent with observation download scheduling (see Fig. 4.3):

- if $\mathbf{dSeq} = [d_1, \ldots, d_n]$ is the sequence of observation downloads and, for each channel $c \in [\![1, \mathbf{Nc}]\!]$, if $\mathbf{cSeq}_c = [(o_1, f_1), \ldots, (o_q, f_q)]$ is the sequence of file downloads on channel c, if two observation downloads d_{i-1}, d_i are ordered, no file from d_{i-1} should be scheduled after a file of d_i on the same channel :

$$\forall i \in [\![2, n]\!] : \max_{j \in [\![1,q]\!] \mid o_j = d_{i-1}} j < \min_{j \in [\![1,q]\!] \mid o_j = d_i} j \tag{4.3}$$

- the same way, if $\mathbf{dSeq} = [d_1, \ldots, d_n]$ is the sequence of observation downloads and, for each memory bank $m \in [\![1, \mathbf{Nm}]\!]$, if $\mathbf{mSeq}_m = [(o_1, f_1), \ldots, (o_r, f_r)]$ is the sequence of file downloads from memory bank m, if two observation downloads d_{i-1}, d_i are ordered, no file from d_{i-1} should be scheduled after a file of d_i on the same memory bank :

$$\forall i \in [\![2, n]\!] : \max_{j \in [\![1,r]\!] \mid o_j = d_{i-1}} j < \min_{j \in [\![1,r]\!] \mid o_j = d_i} j \tag{4.4}$$

Temporal constraints

- an observation can be downloaded only when finished:

$$\forall o \in [\![1, \mathbf{No}]\!] : (\mathbf{dw}_o \neq 0) \rightarrow (\mathbf{sdo}_o \geq \mathbf{Eo}_o) \tag{4.5}$$

- all the files generated by an observation o must be downloaded within the download window assigned to o:

$$\begin{aligned} &\forall o \in [\![1, \mathbf{No}]\!] : \\ &(\mathbf{dw}_o \neq 0) \rightarrow (\mathbf{sdo}_o \geq \mathbf{Sw}_{\mathbf{dw}_o}) \wedge (\mathbf{edo}_o \leq \mathbf{Ew}_{\mathbf{dw}_o}) \end{aligned} \tag{4.6}$$

- file downloads on a channel must not overlap, that is for each channel $c \in [\![1, \mathbf{Nc}]\!]$, if $\mathbf{cSeq}_c = [(o_1, f_1), \ldots, (o_q, f_q)]$:

$$\forall i \in [\![2, q]\!] : \mathbf{sdf}_{o_i, f_i} \geq \mathbf{edf}_{o_{i-1}, f_{i-1}} \tag{4.7}$$

- file downloads from a memory bank must not overlap, that is for each memory bank $m \in [\![1, \mathbf{Nm}]\!]$, if $\mathbf{mSeq}_m = [(o_1, f_1), \ldots, (o_r, f_r)]$,

$$\forall i \in [\![2, r]\!] : \mathbf{sdf}_{o_i, f_i} \geq \mathbf{edf}_{o_{i-1}, f_{i-1}} \tag{4.8}$$

- the duration of each file download must be respected; let $\mathbf{DuDl}(v, w, t)$ be the duration required to download a volume v within download window w when starting the download at time t; because the download rate $\mathbf{Dr}_w(t)$ is assumed to be a piecewise constant function of time t, the download duration $\mathbf{DuDl}(v, w, t)$ can be easily computed in an exact manner:

$$\begin{aligned} &\forall o \in [\![1, \mathbf{No}]\!], \forall f \in \mathbf{Sf}_o : \\ &(\mathbf{dw}_o \neq 0) \rightarrow (\mathbf{edf}_{o,f} - \mathbf{sdf}_{o,f} = \mathbf{DuDl}(\mathbf{V}_{o,f}, \mathbf{dw}_o, \mathbf{sdf}_{o,f})) \end{aligned} \tag{4.9}$$

- there must be enough time to move the antenna from a station to another one and to reset the key change tables when necessary; let $\mathbf{DuMvAnt}(w, w', t)$

be the minimum duration required to move from a pointing towards station s associated with download window w to a pointing towards station s' associated with download window w' when starting the movement at time t; an iterative method, similar to the algorithms used to compute satellite attitude movements in minimum time, is used to compute $\mathbf{DuMvAnt}(w, w', t)$; if $\mathbf{dSeq} = [o_1, \ldots, o_n]$ is the sequence of observation downloads:

$$\forall i \in [\![2, n]\!] : \\ (\mathbf{sdo}_{o_{i-1}} \leq \mathbf{sdo}_{o_i}) \wedge (\mathbf{edo}_{o_{i-1}} \leq \mathbf{edo}_{o_i}) \tag{4.10}$$

$$(\mathbf{dw}_{o_i} \neq \mathbf{dw}_{o_{i-1}}) \rightarrow ((\mathbf{sdo}_{o_i} - \mathbf{edo}_{o_{i-1}} \geq \\ \max(\mathbf{DuKct}, \mathbf{DuMvAnt}(\mathbf{dw}_{o_{i-1}}, \mathbf{dw}_{o_i}, \mathbf{edo}_{o_{i-1}}))) \wedge \\ (\mathbf{nkc}_{o_i} = 1)) \tag{4.11}$$

$$((\mathbf{dw}_{o_i} = \mathbf{dw}_{o_{i-1}}) \wedge (\mathbf{U}_{o_i} \neq \mathbf{U}_{o_{i-1}}) \wedge \\ (\mathbf{nkc}_{o_{i-1}} = \mathbf{Nkmax})) \rightarrow ((\mathbf{sdo}_{o_i} - \mathbf{edo}_{o_{i-1}}) \geq \mathbf{DuKct}) \wedge \\ (\mathbf{nkc}_{o_i} = 1)) \tag{4.12}$$

$$((\mathbf{dw}_{o_i} = \mathbf{dw}_{o_{i-1}}) \wedge (\mathbf{U}_{o_i} \neq \mathbf{U}_{o_{i-1}}) \wedge \\ (\mathbf{nkc}_{o_{i-1}} < \mathbf{Nkmax})) \rightarrow (\mathbf{nkc}_{o_i} = \mathbf{nkc}_{o_{i-1}} + 1) \tag{4.13}$$

$$\Big((\mathbf{dw}_{o_i} = \mathbf{dw}_{o_{i-1}}) \wedge (\mathbf{U}_{o_i} = \mathbf{U}_{o_{i-1}})\Big) \rightarrow (\mathbf{nkc}_{o_i} = \mathbf{nkc}_{o_i} \tag{4.14}$$

Constraint 4.10 enforces that two successive observations downloads in the sequence are actually scheduled in the same order on channels. Constraint 4.11 enforces that, when two successive observation downloads are not in the same download window, there must be enough time between the end of the first download and the start of the second to move the antenna from the first station to the second one and to reset the key change table (this reset is performed at each transition to make the problem easier to solve). The current number of key changes is then reset to 1. That means that we take systematically benefit of antenna moves to reset the key change table. Constraint 4.12 enforces that when two successive observation downloads on the same download window do not belong to the same user and when the maximum number of changes in the table is reached, there must be enough time between the end of the first download and the start of the second to reset the key change table. The current number of key changes is then reset to 1. Constraint 4.13 enforces that when two successive observation downloads on the same download window do not belong to the same user and when the maximum number of changes in the table is not reached, the current number of key changes is incremented. Finally, constraint 4.14 enforces that when two successive acquisition downloads on the same download window belong to the same user, the number of key changes remains unchanged. That means that we reset the key change table only when necessary.

Computing antenna movement durations The term $\mathbf{DuMvAnt}(w, w', t)$ represents the duration required for the antenna to move from a pointing towards the station associated with w to a pointing towards the station associated with w' if

the transition is started at time t. This duration may be computed iteratively via algorithms similar to those ones used for computing attitude transition duration in minimum time under the hypothesis, true in the practical case, that the antenna is more agile than the satellite [Pralet et al., 2014b]. This method is directly inspired from the *false position method*.

To compute $\mathbf{DuMvAnt}(w, w', t)$, it is necessary to get, in the satellite reference frame, the antenna orientation vector or associated with downloading in window w at time t. This vector corresponds to the antenna angular position at the start of the transition.

The method uses the notion of *delay function* F associated with the transition. Let $or'(u)$ be the antenna orientation vector associated with downloading in window w' at time u, the delay function is defined as $F(u) = t + angle(or, or'(u))/\mathbf{Sa} - u$. This function expresses the delay obtained if the movement is started at time t to finish at time u. The expression is the difference between, on one hand the arrival date of a transition allowing to get orientation $or'(u)$ required at time u ($t + angle(or, or'(u))/\mathbf{Sa}$), and on the other hand the date u itself. If the delay is positive, it means that it is not possible to get to the required orientation to download on w' at time u. If the delay is negative, it means that the antenna rotation speed is sufficient to download on w' at time u. A null delay corresponds to an antenna movement of minimum duration.

The iterative method considers at each step two potential end dates t_1 and t_2 for the antenna movement. These dates may be, for example, initialized to $\mathbf{Sw}_{w'}$ and $\mathbf{Ew}_{w'}$ respectively, that is to the lowest and greatest end dates for the antenna movement, which must end during visibility window w'. The method then proceeds as following :

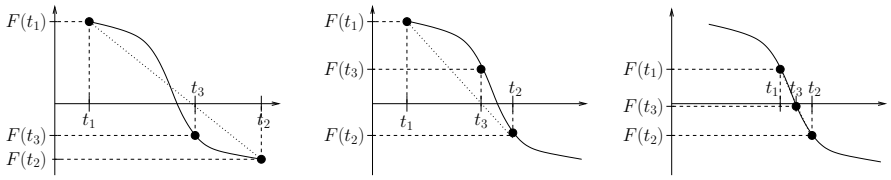
1. it computes the delay $F(t_1)$ associated with an antenna movement ending at date t_1; if the delay is negative or null ($F(t_1) \leq 0$), then it is possible for downloads to start at t_1; then, it returns $\mathbf{DuMvAnt}(w, w', t) = angle(or, or_{w'}(t_1))/\mathbf{Sa}$;

2. otherwise, if $F(t_1) > 0$, it computes the delay $F(t_2)$ associated with an antenna movement ending at t_2; if the delay is strictly positive ($F(t_2) > 0$), then it means that antenna speed is not sufficient for downloading anything in visibility window w'; in this case, it returns $\mathbf{DuMvAnt}(w, w', t) = +\infty$;

3. otherwise (when it is not possible to start downloading at t_1 but it is possible to start downloading at t_2), it computes a "good" temporal position t_3 that is between t_1 and t_2 with a linear interpolation ($t_3 = ((t + \delta_1) \cdot t_2 - (t + \delta_2) \cdot t_1)/((\delta_1 - t_1) - (\delta_2 - t_2))$); see Fig. 4.4(a) for an illustration;

4. it computes the delay $F(t_3)$ associated with an antenna movement ending at t_3; if the delay is negative or null ($F(t_3) \leq 0$), then it is possible to start downloading at t_3 on w'; in this case, step 3 is done again with $t_2 = t_3$ (see the transition between Fig. 4.4(a) and Fig. 4.4(b)); otherwise, if $F(t_3) > 0$ (it is not possible to start downloading at t_3 in w'), step 3 is iterated again with $t_1 = t_3$ (see the transition between Fig. 4.4(b) and Fig. 4.4(c)).

The algorithm stops when the maximum number of iterations has been reached or when reaching a sufficient precision on dates, such as in iteration 3 on Fig. 4.4(c).



(a) iteration 1 (b) iteration 2 (c) iteration 3

Figure 4.4: Computing antenna movement durations, with F the delay function associated to the transition.

4.2.4 Optimization criterion

4.2.4.1 Resources allocation and equity

Resource allocation is a major problem in computer science and economics in which several types of resources (dividable, non-dividable, time, objects) must be allocated to different agents. Agents may have preferences regarding what they receive. Resource allocation may be distributed or centralized. Agent preferences may change over the allocation process such as in an auction. The objective is to find either a feasible allocation (satisfying some constraints) and/or to find an optimal allocation regarding agent preferences. To evaluate or qualify this allocation, an aggregated of individual preferences may be computed, the *social welfare*.

Social welfare for the Earth-observation satellite domain has been studied [Lemaître et al., 1999] [Lemaître et al., 2003]. A more general review of allocation problems has been published in [Chevaleyre et al., 2006]. In the problem we are studying, the resources are indivisible. Users of the systems (country, group of countries, military instances, companies) are the agents. Each entity has some preferences. A central instance is in charge of computing activity plans and must ensure that preferences are taken into consideration when allocating the satellite resource. The allocation is centralised but negociations happen and several plans are usually proposed before every entity agrees. The goals of each user are often *antagonistic* (increasing the utility of one user often results in decreasing the utility of another user) but it is not always the case. Several users may want the same observation or observations may be geographically distributed over the satellite's orbit. In this case, there is no *conflict*. Also, users shall get a return on investment proportional to their contribution to the system.

To compare several activity plans, there are several operators in the literature. Let us detail three different examples. Let A, B, C be three activity plans, let n be the number of users, and let $Up_x = (u_1, ..., u_n)$ be the vector of individual utilities for an activity plan x. An individual utility u_i measures the satisfaction of user i in the plan. Let $Up_A = (10, 1, 1)$ be the utility vector of plan A, which produces an utility equal to 10 for user 1 and an utility equal to 1 for users 2 and 3. Similarly, let $Up_B = (4, 2, 3)$ and $Up_C = (5, 2, 5)$ be two other utility vectors.

To compare these plans, it is first necessary to define *Pareto optimality*. A vector of individual utilities represents the satisfaction of each individual. A vector of individual utilities is *Pareto efficient* if, compared to other vectors, it is not possible to increase the satisfaction of one user without decreasing the satisfaction of other individuals (see Fig. 4.5 for an example). For example, Up_B is not Pareto efficient because Up_C has either higher or equal individual utilities. Up_C is then always more

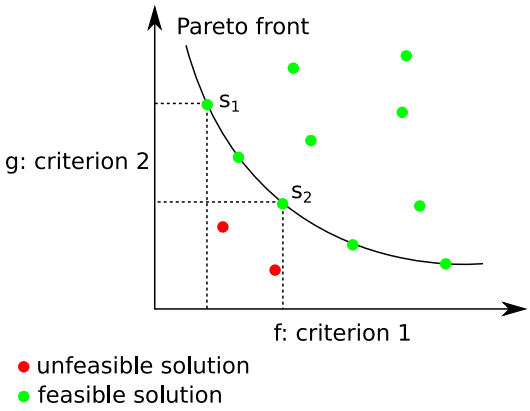


FIGURE 4.5: Example of Pareto front for a 2-objective problem. Solutions s_1 and s_2 are on the front because $f(s_1) > f(s_2)$ and $g(s_1) < g(s_2)$. All solutions on the Pareto front are Pareto-efficient.

interesting, in the sense of Pareto, than Up_B. It is generally admitted that an operator choosing a solution among several must choose a Pareto-efficient solution. The difficulty now is to choose a solution among several Pareto-efficient solutions.

Let now compare Up_A and Up_C with the $\sum$ operator that sums all utilities. Sum is $10 + 1 + 1 = 12$ for Up_A and $5 + 2 + 5 = 12$ for Up_C. The $\sum$ operator is said to be *utilitarist*, it is Pareto-efficient and does not take into account *equity* between individuals. If equity matters, the min operator can be used instead to choose the solution maximizing the minimum individual utility. Unfortunately, it is not Pareto-efficient. If Up_B and Up_C are compared, minimum individual utilities are equal, this operator may choose Up_B which is not Pareto-efficient. The leximin operator is also favoring equality but is Pareto-efficient. If Up_B and Up_C are compared, their vectors are first ordered by increasing individual utility. Vectors $(2, 3, 4)$ and $(2, 5, 5)$ are then lexicographically compared until one individual utility dominates the other (here, 5 dominates 3). Here, Up_C is chosen. In the following, we present a criterion which produces these individual utilities and includes a mechanism to support utilitarism or egalitarism.

4.2.4.2 Criterion for the Data Download Problem

The optimization criterion we consider to discriminate several consistent download schedules (which satisfy all the constraints) takes into account the user preferences (observation priorities and weights), the information age (temporal distance between observations and data delivery to the users who required them), and the fair sharing of the system between its users.

Let Π be a fixed download schedule where all decision variables (non temporal and temporal) have been set.

We can associate with each observation $o \in [\![1, \mathbf{No}]\!]$ a *freshness value* $\mathbf{fr}_o$ between 0 et 1. This value depends on the temporal distance between the end time $\mathbf{Eo}_o$ of observation o and the time $\mathbf{dt}_o$ at which the last file of o is delivered to the user u that required o (including the ground transfer duration between the reception station

and the data production center of u, which can be estimated via simulation). More precisely, the freshness $\mathbf{fr}_o$ of o is defined by:

$$\mathbf{fr}_o = \begin{cases} \mathbf{Fresh}_o(\mathbf{dt}_o - \mathbf{Eo}_o) = 2^{(\mathbf{Eo}_o - \mathbf{dt}_o)/\mathbf{Ht}_o} \text{ if } (\mathbf{dw}_o \neq 0) \\ 0 \text{ otherwise} \end{cases} \tag{4.15}$$

We can now associate with each observation o a *corrected weight* $\mathbf{wc}_o$ which is the product of its (normalized) weight $\mathbf{W}_o$ by its freshness $\mathbf{fr}_o$: $\mathbf{wc}_o = \mathbf{W}_o \cdot \mathbf{fr}_o$. This way, we penalize observations regarding their freshness while taking their weight into account.

In such conditions, we can associate with each priority level $p \in [\![1, \mathbf{Np}]\!]$ and each user $u \in [\![1, \mathbf{Nu}]\!]$ what we refer to as the *individual utility* $\mathbf{ui}(\Pi, p, u)$ of the download schedule Π at priority level p for user u. This utility is defined by:

$$\mathbf{ui}(\Pi, p, u) = \begin{cases} 1 \text{ if } \nexists o \in [\![1, \mathbf{No}]\!] \mid (\mathbf{P}_o = p) \wedge (\mathbf{U}_o = u) \\ \displaystyle\sum_{o \in [\![1, \mathbf{No}]\!] \mid (\mathbf{P}_o = p) \wedge (\mathbf{U}_o = u)} \mathbf{wc}_o \text{ otherwise} \end{cases} \tag{4.16}$$

It is equal to the sum of the corrected weights of the observations that are downloaded in Π and that are of priority p and required by u if any exists. If none exists, it is is equal to 1 (perfect satisfaction because nothing is required). Thanks to weight normalization, it takes its value in $[0, 1]$.

The problem is now to aggregate individual utilities. Following [Lemaître et al., 2003], we associate with each priority level $p \in [\![1, \mathbf{Np}]\!]$ what we refer to as the *social utility* $\mathbf{us}(\Pi, p)$ of the download schedule Π at priority level p. Let $\mathbf{Ps}$ be a sharing parameter set in agreement with the system users. This utility is defined by:

$$\mathbf{us}(\Pi, p) = \begin{cases} \left(\frac{1}{\mathbf{Nu}} \cdot \sum_{u \in [\![1, \mathbf{Nu}]\!]} (\mathbf{ui}(\Pi, p, u))^{\mathbf{Ps}} \right)^{1/\mathbf{Ps}} & \text{if } \mathbf{Ps} \neq 0 \\ \left(\displaystyle\prod_{u \in [\![1, \mathbf{Nu}]\!]} \mathbf{ui}(\Pi, p, u) \right)^{1/\mathbf{Nu}} & \text{if } \mathbf{Ps} = 0 \end{cases} \tag{4.17}$$

As individual utilities do, it takes its value in $[0, 1]$. Parameter $\mathbf{Ps}$ allows the trade-off between utilitarism (maximizing the total utility) and egalitarism (allowing every user to have an equal utility) to be adjusted. When $\mathbf{Ps} = 1$, the social utility is the mean of the individual utilities: utilitarian approach without equity preoccupation. When $\mathbf{Ps}$ tends towards $-\infty$, social utility tends towards the minimum of the individual utilities: pure egalitarian approach with a high equity preoccupation. $\mathbf{Ps} = 0$ is a compromise between utilitarian and egalitarian approaches.

We can now associate with each download schedule Π a *social utility vector* where a social utility is associated with each priority level:

$$\mathbf{vus}(\Pi) = (\mathbf{us}(\Pi, 1), \ldots, \mathbf{us}(\Pi, \mathbf{Np})) \tag{4.18}$$

Two download schedules Π_1 et Π_2 can finally be compared by comparing their associated social utility vectors in a lexicographic way from the highest to the lowest priority level. This expresses that any improvement at a priority level is preferred to any improvement at lower priority levels.

4.2.5 Problem analysis

Memory capacity The checking of the memory constraint (absence of memory overwriting) does not appear in the above download scheduling problem definition. However, once a download schedule is built, the associated memory profile can be built and checked. There are two main justifications for such a two stage approach. First, thanks to the very high capacity of new generation onboard mass memories, this constraint is rarely violated. Second, the memory profiles that can be built remain uncertain due to the uncertainty about (1) the actual volumes generated by observations, (2) the arrival of data reception acknowledgments which allow data to be removed from the mass memory and (3) the actual download decisions made on board in the onboard and mixed approaches. This is why we prefer to use the observation execution mechanism that will be presented in Sect. 6.1.2 and prevents any memory overwriting from happening: if performing an observation o may lead to a memory overwriting, either o is not performed, or lower priority data is removed from memory to free space.

Temporal part of the problem It must be stressed that, once all the non temporal variables have been set, that is once all the assignment and scheduling decisions have been made, the remaining part of the problem is made of temporal variables and constraints. This temporal part has the form of a TSTN (Time-dependent Simple Temporal Network [Pralet and Verfaillie, 2013b]). The TSTN framework is an extension of the STN framework. In the STN framework, all the variables represent temporal positions and all the constraints are binary constraints of the form $x - y \geq d$ where x and y are variables and d is a constant, referred to as simple temporal constraints. In the TSTN framework, all the constraints are of the form $x - y \geq d(x, y)$ where x and y are variables and d is a function of x and y (or of the form $x - y \geq d(x)$ where d is a function of x), referred to as time-dependent simple temporal constraints.

Constraints from 4.5 to 4.8, as well as Constraint 4.12 are simple temporal constraints, whereas Constraints 4.9 and 4.11 are time-dependent simple temporal constraints (time-dependent durations of file downloads and antenna movements).

In [Pralet and Verfaillie, 2013b], it has been proven that the main results established in the STN framework can be under some assumptions about the form of the distance d functions (e.g. monotonicity, assumption that is satisfied in our application) extended to the TSTN framework: existence of polynomial constraint propagation algorithms that allow consistency or inconsistency of the TSTN to be determined and, in case of consistency, the earliest and latest times of each temporal variable to be computed; consistency of the assignment where all the temporal variables are set to their earliest time (or to their latest time).

This suggests an algorithmic constraint-based approach where assignment and scheduling decisions are made first, temporal constraints are then propagated and, in case of consistency (no empty domain of a temporal variable), temporal variables are set to their earliest time in order to minimize information age.

Instance size As we will see in Sect. 6.5.2, the real-world instances we have to manage involve more than one thousand of observations over each day. As a consequence, if we consider only download scheduling decisions (variable $\mathbf{dSeq}$), the size of the search space is higher than !1000 (we note that the size of the search space does not imply that the problem is hard as we can prune it). The size of this search space and the absence of polynomial scheduling algorithms prevent us from using

exact algorithms and suggest the use of approximate algorithms (based on greedy or local searches) to make assignment and scheduling decisions and then to propagate the resulting temporal constraints (time-dependent or not).

4.2.6 Related works

Data download planning Whereas the problem of selecting and scheduling observations for Earth observation satellites has been extensively studied (see [Lemaître et al., 2002; Globus et al., 2004b] for partial surveys), studies about data download planning are more seldom. The problem of planning (offline on the ground) data downloads from a Mars orbiter to Earth has been studied in [Oddi et al., 2003; Cesta et al., 2007]. In [Oddi and Policella, 2004; Righini and Tresoldi, 2010], it has been shown that this problem can be modeled as a Max-Flow problem and thus solved using either Max-Flow or Linear Programming polynomial algorithms. To use a similar approach in our context, we should linearize the criterion and relax many constraints, for example the constraints that enforce that a file download cannot be interrupted and distributed over several channels or download windows. Hence, a Max-Flow formulation of our problem can only produce upper bounds on the maximum utility. In the context of the EO-1 experiment [Chien et al., 2004a], observation requests may be generated either by ground users, or autonomously on board following the detection of ground phenomena such as volcanic eruptions, floods, or ice breakups by onboard data analysis algorithms. In such a context, observation and download plans are built or adapted on board using an iterative repair approach [Chien et al., 2000]: local search algorithms implemented in the generic CASPER planning tool.

Uncertain volumes The situation where the amount of data that is produced and must be downloaded is variable and uncertain is present in planetary exploration as well as in Earth observation and surveillance missions. In [Castano et al., 2007] and [Woods et al., 2009a], onboard planning and scheduling (including data download planning and scheduling) is studied in order to allow a planetary exploration rover to adapt itself to what it detects. More precisely, in [Thompson et al., 2008], what is downloaded is selected in order to meet download limitations and to maximize the information value that scientists will be able to extract from it. In [Oddi and Policella, 2004], an iterated procedure is designed to lower filling peaks in memory banks in order to manage uncertainty about the volumes of data to be downloaded by a Mars orbiter and to limit the risk of memory bank overwriting. In [Righini and Tresoldi, 2010], Integer Linear Programming formulations are proposed to address this problem. In [Chien et al., 2004a], uncertainty is managed using a reactive iterative repair planning approach. In [Verfaillie et al., 2011], the operational context is an electromagnetic Earth surveillance mission where the variability of the generated data volumes is very high. Several mechanisms, especially designed for onboard download decision-making, are compared in terms of computing time, number of downloads, and window utilization: decision rules, reactive planning, or sampling + planning.

Algorithms for the deterministic Data Download Planning Problem 5

We now present algorithms to solve the deterministic version of the data download scheduling problem, that is the version without uncertainties on data volumes. We look at an exact method first and then we move to greedy searches and metaheuristics.

5.1 Exact method: constraint programming

The data download problem has been described as a constraint satifaction problem in Chap. 4. Then, we can try to solve it with classical CP solvers, such as *IBM CPLEX CP Optimizer*. The latter is an optimization software developed by IBM in which specific data structures for scheduling are available such as time intervals (visibility windows or data downloads) or sequences of time intervals (channel or memory banks resources). Generic algorithms then find an optimal solution to the problem. Users may define heuristics in the model to speed the search. The data download problem that we have described in Sect. 4.2 has been written almost directly in OPL (*Optimization Programming Language*) which allows to express data, constraints and the objective function. In the OPL model that can be found in Appendix C, simplifications have been made.

The model had to be adapted. The objective function defined in Sect. 4.2.4.2 includes several priorities. The plan has to be optimized for all priorities while maintaining a strict prioritization between them. As it is not possible to define a *multi-objective* function in *OPL*, we have built an iterative hierarchic optimization mechanism. The idea is to optimize only one priority level at a time while saving the optimal value of the objective function for priority levels that have already been processed.

More precisely, the algorithm calls the solver for each priority level. Let p_c be the current optimized priority. The solver starts with the highest priority $p_c = 1$ and find a schedule Π. An optimal value $\mathbf{us}(\Pi, p_c)$ is found. Because only priority p_c was optimized, it means that downloads from other priorities are not in the plan, because they do not contribute to the current optimization criterion $\mathbf{us}(\Pi, p_c)$. For the same reason, if the algorithm simply launches the solver for the next priority $p_c + 1$, it will not include the downloads of priority p_c. To bypass this difficulty we add $utilMin[p]$ for all priorities $p \in [\![1, \mathbf{Np}]\!]$ to the data part of the model. This quantity represents the minimum value that the solver must attain for priority

```
forall(i in Nfs){

    // each file downloaded using at most one of the alternative channels
    alternative(itvs[i], all(c in Ncs) altItvChannels[i][c]);

    // each file downloaded using at most one of the alternative windows
    alternative(itvs[i], all(w in Nws) altItvWindows[i][w]);

    //all files linked to an image should be downloaded in a single window
    //forall(j in Nfs : Acqf[i] == Acqf[j] && i< j){
    forall(w in Nws){
        presenceOf(altItvWindows[i][w]) == presenceOf(altItvWindows[0][w]);
    }
    //}

    /*a file should be downloaded only after the end of the acquisition it is part of*/
    startOf(itvs[i],AstopInt[Acqf[i]]) >= AstopInt[Acqf[i]];

    /*an acquisition should be downloaded on one of the entity's station*/
    forall(w in Nws:(Ss[Ea[Acqf[i]]][Stc[w]] == 0)){
        presenceOf(altItvWindows[i][w]) == 0  ;
    }

}
```

Figure 5.1: OPL code extract.

p. At the beginning $utilMin[p] = 0, \forall p \in [\![1, \mathbf{Np}]\!]$. Then, at each iteration of the algorithm, $utilMin[p_c]$ is set to the optimal value $\mathbf{us}(\Pi, p_c)$ found. Also, this constraint is added to the model :

$$\forall p \in [\![1, p_c - 1]\!], \mathbf{us}(\Pi, p) \geq utilMin[p] \tag{5.1}$$

This forces the solver to optimize priority p_c while maintaining a utility as good as previously found for higher priorities.

On Fig 5.2, an example of iterative optimization is shown. Gradually, each set of observations corresponding to a priority is added to the plan. On this example, the scheduling remains the same for previous downloads but it is not always the case. That is why a general constraint such as Constraint 5.1 is needed. The evolution of download windows can be observed. On the upper part, we see how the high-priority downloads are scheduled first. Optimizing the freshness is equivalent to downloading them at the earliest possible date. Once high-priority acquisitions are scheduled, priority two and then three are considered.

The main difficulty, when using such generic tools, is to provide a good modelling of the problem. Changing the model may seriously affect the performances and it is somehow necessary to know what type of algorithm is processing the model to do an efficient modeling. We must also note that it is not possible to model time-dependent constraints (such as those used for the telemeasure antenna) in OPL. Moreover, computing an optimal solution for real instances is far from possible because of their sizes (> 5000 files). Finally, embedding such a solver on board a satellite is not acceptable due to embedded software constraints. This is why this approach has been discarded.

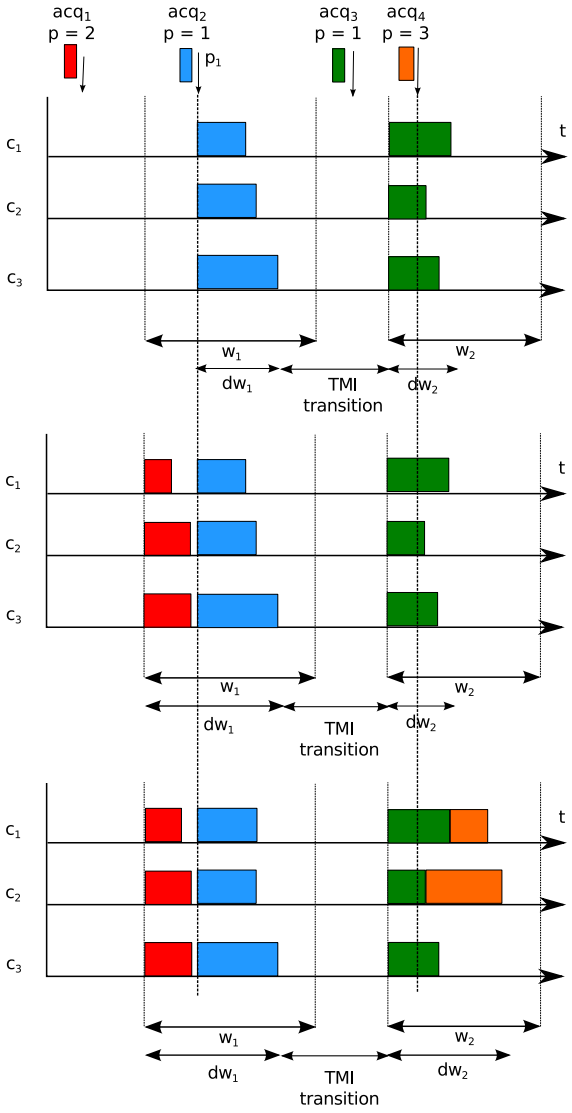


FIGURE 5.2: Iterative hierarchic optimization on an instance involving 3 priorities (p_1, p_2 et p_3), 3 channels (c_i), 4 acquisitions and 2 visibility windows (w_1 and w_2). An acquisition is associated with a color, blocks of the same color are files of the same acquisition. Vertical arrows represent end dates for acquisitions.

5.2 Greedy algorithms

Because exact methods are not suitable for the large instances of the data download problem, we developed several greedy algorithms. In this section, we present an algorithm to schedule file downloads on channels and memory banks, and two algorithms building complete plans: a chronological algorithm that is destined to be used on board the satellite and a non-chronological algorithm that is destined to be used on the ground (because it requires a lot more constraint cheking than the chronological one as it does not maintain a current *state*).

5.2.1 Scheduling file downloads onto channels and memory banks

Once an observation is assigned to a visibility window, the download of its files must be scheduled onto channels and memory banks. The search space for this problem is very large as all combinations with other file downloads from other observations have to be considered before declaring the schedule to be optimal. After trying many algorithms we decided to shrink the search space by applying three restrictions. The first one is to forbid interleaving of file downloads belonging to different observations (see constraint 4.3 in the model and Fig. 4.3). The second one is the way we manage key change table updates (see Constraints 4.11 to 4.14). The third one concerns how file downloads are scheduled on channels and memory banks via the following greedy rule.

R1 file insertion rule To insert the download of an observation o, we try to insert the download of all its files from the largest to the smallest size (see Algorithm 3). For each file f recorded on memory bank $\mathbf{M}_{o,f}$, the following rule is applied: if there exists a channel c such that the memory bank previously read by c is $\mathbf{M}_{o,f}$, then select c in order to avoid as much as possible conflicts of access to memory banks; otherwise, select a channel c that allows f to be downloaded as soon as possible.

It is possible to prove that, if used in a chronological order, this rule produces plans such that on every channel, there is no idle period between two consecutive file downloads except in the case where the end time of an observation must be waited for. The channel selection rule avoids memory bank access conflicts and interleaving between observation downloads. See Appendix B for details.

In the illustrative example of Fig. 4.2, one can check that this rule has been effectively applied to insert the download of observation H in window 1 just after the download of observation C, assuming that the downloads of high priority observations C, B, and F have been first inserted and that $[(H,4),(H,1),(H,5)]$ is the file size decreasing order for H. The insertion succeeds if an only if all the files have been inserted and all the resulting temporal constraints can be satisfied.

5.2.2 A chronological greedy algorithm

In this section, we give a description of a chronological algorithm that computes executable plans from scratch and that is destined to be used for a full onboard decision-making scheme (it can also be used for ground planning but it is primarily designed to be fast). Later on, we will compare this pure onboard approach with pure ground and flexible algorithms.

At each iteration, the algorithm adds an acquisition download to the end of the current download plan. We do not describe precisely what happens with files and channels as this is handled by the R1 insertion rule.

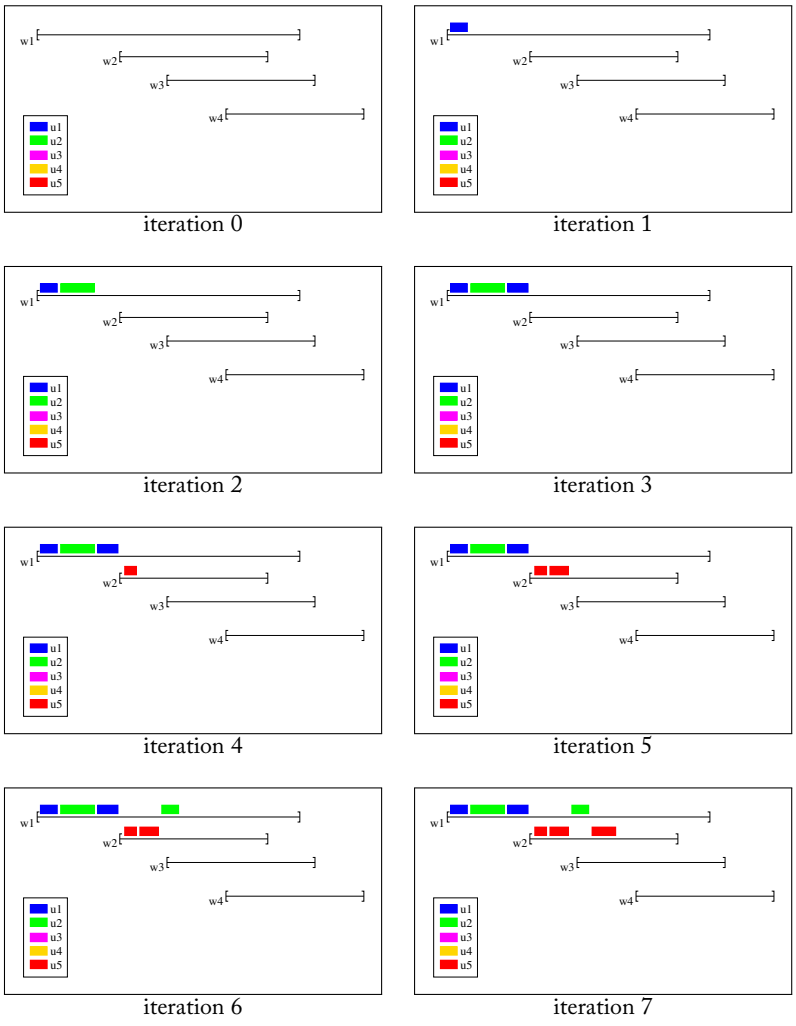


FIGURE 5.3: Iterations of the chronological greedy algorithm, macroscopic view without files and channels.

Algorithm 3: R1 file insertion rule

Data : observation o to insert; for each channel $c \in [\![1, \mathbf{Nc}]\!]$, (o_c, f_c), the last file download on c (download of file f_c of observation o_c). If no download has been done using channel c since the beginning of the current download window, then $(o_c, f_c) = \varnothing$;

1. select a file $f \in [\![1, \mathbf{Nf}_o]\!]$ not yet downloaded and of maximum size;
2. select a channel c for f:
   (a) if there exists $c \in [\![1, \mathbf{Nc}]\!]$ such that the last memory bank read by c is the same as the memory bank of f ($(o_c, f_c) \neq \varnothing$ and $\mathbf{M}_{o_c, f_c} = \mathbf{M}_{o,f}$), then select channel c;
   (b) otherwise, select the earliest available channel;
3. insert f on the chosen channel c;
4. $(o_c, f_c) \leftarrow (o, f)$.

5.2.2.1 Greedy chronological algorithm main loop

The algorithm maintains a state through the execution. At each iteration, the algorithm moves forward in time and updates its current state regarding several types of events : start and end of a visibility window, end of an acquisition, end of an antenna movement, end of updates of the key change tables and end of file downloads. To update the state, it simulates all these events. When this is done, the algorithm chooses an acquisition to be downloaded at the current time and a visibility window for downloading this acquisition. It then inserts the acquisition download at the end of the download plan. This scheme is repeated until the current time reaches the end of the scheduling horizon (see Algorithm 4).

Algorithm 4: Full board algorithm main loop

1. Update current state;
2. if there is any candidate observation and candidate visibility window :
   a) choose an observation o to download using heuristic **HD1** (described in section 5.2.2.2);
   b) choose a visibility window w for o using heuristic **HD2** (described in section 5.2.2.3);
   c) insert o in w at the end of the current plan using the **R1** file download insertion rule described in section 5.2.1.

5.2.2.2 Decision heuristic HD1

The greedy chronological algorithm makes use of a first heuristic to choose an acquisition to download at the end of the current download plan. To be chosen, this acquisition must be:

– stored in memory at the current date;

– downloadable in one of the active visibility windows.

Among the acquisitions that satisfy the previous conditions, the chosen acquisition is the one:

– of maximum priority;
– at the same priority level, of minimum end date, to minimize data freshness.

To ensure that an acquisition is downloadable in one of the active visibility windows, it would be necessary to consider antenna movements and to try to schedule file downloads onto channels and memory banks to check whether or not download is physically possible. To make this computation faster, we use a pessimist approximation that uses an upper bound on the duration of these transitions and file downloads.

More precisely, a pessimistic volume $vmaxc$ to be downloaded for acquisition o on every channel is computed; let $Vtot_o = \sum_{f \in [1..\mathbf{Nf}_o]} \mathbf{V}_{o,f}$ be the total volume associated with o and let $Vmax_o = \max_{f \in [\mathbf{Sf}_o]} \mathbf{V}_{o,f}$ be the maximum volume for a file associated with o; let $vminc$ and $vmaxc$ be the minimum and maximum volume to be downloaded on any channel to download only o; it is possible to demonstrate that the R1 algorithm builds a download plan such that $vmaxc \leq vminc + Vmax_o$, that is the difference between the least busy channel and the most busy one is less than or equal to the maximum volume of a file $Vmax_o$ of o (see Appendix B); in the worst case, a channel has a charge equal to $vmaxc$ and all others have a charge equal to $vmaxc - Vmax_o$; because the total volume is equal to $Vtot_o$, in the worst case:

$$vmaxc + (\mathbf{Nc} - 1) \cdot (vmaxc - Vmax_o) = Vtot_o$$

so:

$$vmaxc = (Vtot_o - Vmax_o)/\mathbf{Nc} + Vmax_o$$

5.2.2.3 Decision heuristic HD2

The greedy chronological algorithm makes use of a second heuristic to choose a window in which the chosen acquisition should be downloaded at the end of the current download plan. To be chosen, this visibility window must be:

– candidate at the current time, that is not yet finished,
– able to receive the chosen acquisition inside its time bounds (same pessimistic estimation as in HD1),
– allowed by the user.

Among the windows that satisfy the previous conditions, the chosen window is :

– either leading to a minimum download end date ([HD2-1] version); this heuristic penalizes long antenna pointing transfers and favors staying on already pointed stations;
– or minimizing the download duration ([HD2-2] version); it favors high-rate periods although antenna pointings may take time.

Heuristic HD1 (resp. HD2) should be used in scenarios where the total volume to download is high (resp. low). A decision leads either to stay on the current visibility window (see iteration 2 to iteration 3 on Fig. 5.3), or to perform an antenna movement to point towards another station (see iteration 3 to iteration 4 on Fig. 5.3). In addition, we see that the algorithm may choose an already visited station (see iteration 5 to iteration 6 on Fig. 5.3). Other heuristics could be used, to take into account ground transfer durations for example.

5.2.3 A non-chronological greedy algorithm

This algorithm, as the chronological algorithm, builds a download schedule from scratch. It is primarily designed to be used on the ground as it is non-chronological and then requires constraint propagation at each iteration.

The principle of the non-chronological greedy algorithm is very basic. It inserts acquisition downloads one after each other in the plan following an insertion order A (see Algorithm 5) given as an input (ordering observations by priorities and then by weights is a straightforward method to produce such an insertion order). The algorithm is said to be non-chronological because an observation that is considered later according to the specified order may be finally downloaded earlier in the resulting download plan [Pralet et al., 2014b].

Algorithm 5: Non-chronological greedy algorithm

1 **Data** : $O = [o_1, ..., o_n]$ the ordered set of candidate acquisitions over the planning horizon

1. Init P, an empty plan
2. For $i : 1 \rightarrow n$
   a) Choose a visibility window $w_i \in [\![1, \mathbf{Nw}]\!]$ and a position $k_i \in [\![1, n]\!]$ in the sequence of acquisition downloads for o_i using heuristic **H1**;
   b) Insert (o_i, w_i) at position k_i in the sequence of downloads
   c) Insert all files of o_i using rule **R1**;
   d) Check constraints;
   e) If constraint violation, undo changes and go to (a) when other (w_i, k_i) pairs can be selected.

The algorithm starts from an empty schedule and considers all the observations in the specified order. For each of them, it tries to insert its download in the current download schedule. To insert the download of an observation o, the algorithm explores all the pairs made of a download window w and of a position k in the sequence of downloads already scheduled. Pairs are explored chronologically from the earliest to the latest in order to minimize information age. After each insertion, all the non-temporal constraints are checked and all the temporal constraints are propagated in the current TSTN (see Sect. 5.2.3.2 for more details). If any constraint is violated, another pair window/position is selected for o until all the possible pairs are exhausted. If all of them have been exhausted without any success, o is rejected.

Fig. 5.4 shows the first iterations of the non-chronological greedy algorithm. There are five users (u_1 to u_5). Each iteration adds a new observation download in the plan, not necessarily at the end. See for example between iteration 3 and iteration 4, where an acquisition associated with user u_2 is added before other downloads already in the plan.

5.2.3.1 H1 decision heuristic

The H1 decision heuristic chooses an allowed visibility window w and a position p for o in the current download sequence $\mathbf{dSeq} = [o_1, \ldots, o_n]$. There are two

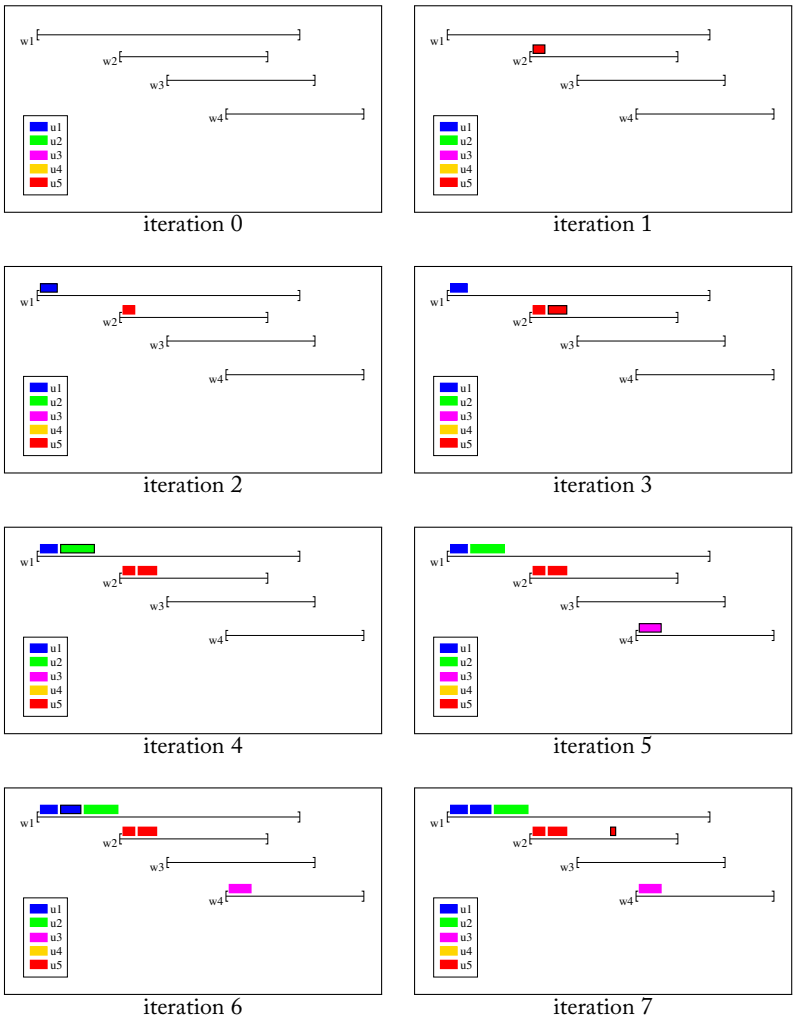


FIGURE 5.4: Iterations of a non-chronological greedy algorithm; macroscopic view (without files and channels)

versions for this heuristic. In the following paragraph, let $Vtot_o$ be the total volume associated with observation o, with $Vtot_o = \sum_{f \in [\mathbf{Sf}_o]} \mathbf{V}_{o,f}$.

Definition: Idle period Let $\mathbf{dSeq} = [o_1, \ldots, o_n]$ be the sequence of observation downloads. It is said that there is an *idle period* between two downloads o_{i-1} and o_i in the sequence if and only if the download end date $\mathbf{sdo}_{o_{i-1}}$ of the files of o_{i-1} is not constraining the download start date of the files of o_i. More formally, there is an idle period before the observation download o_i if and only if the smallest download start date of a file of o_i is equal to the visibility window start date which contains o_i (date $\mathbf{Sw}_{\mathbf{dw}_{o_i}}$) or to the end of the observation o_i (date $\mathbf{Eo}_{o_i}$). This definition is valid even if o_i is the first observation download of the plan. Note that observations ending during the planning horizon may induce idle periods.

[H1-1] version The first version of the H1 heuristic, H1-1, aims at optimizing data freshness. The heuristic chooses first an allowed visibility window w such that the estimated delivery date to the requesting user is minimized for the selected observation.

To compute this estimated delivery date, we maintain for each visibility window w and each user u allowing downloads onto w, the volume $Vgnd_{w,u}$ of data downloaded onto w and aimed at u in the schedule. These volumes are initialized to 0.

An optimistic estimation $e_{o,w}$ of the delivery date for the observation o to be downloaded onto w is given by:

$$\begin{aligned} e_{o,w} = \quad & \max(\mathbf{Sw}_w, \mathbf{Eo}_o) \\ & + \mathbf{DuDl}(Vtot_o/\mathbf{Nc}, w, \max(\mathbf{Sw}_w, \mathbf{Eo}_o)) \\ & + (Vgnd_{w,\mathbf{U}_o} + Vtot_o)/\mathbf{Gtr}_{\mathbf{St}_w,\mathbf{U}_o} \end{aligned}$$

or, in other words, it is the sum of:

- an optimistic estimation of the earliest download start date of o ($\max(\mathbf{Sw}_w, \mathbf{Eo}_o)$);
- an optimistic estimation of the required duration for downloading o with an equitable repartition of the volume over the channels ($\mathbf{DuDl}(Vtot_o/\mathbf{Nc}, w, \max(\mathbf{Sw}_w, \mathbf{Eo}_o))$, with $\mathbf{DuDl}(v, w, t)$ which returns the required duration to download a volume v onto window w when starting at time t;
- a realistic estimation of the required duration for transmitting the acquisition received by w; the estimation uses the ground transfer rate between the ground reception station $\mathbf{St}_w$ and the production center associated to the user requesting o ($(Vgnd_{w,\mathbf{U}_o} + Vtot_o)/\mathbf{Gtr}_{\mathbf{St}_w,\mathbf{U}_o}$); it also implicitly models network congestion by considering that o is transmitted last.

The H1-1 heuristic selects a visibility window w, allowed for user $\mathbf{U}_o$, such that the estimation $e_{o,w}$ is minimized. When an acquisition download o is actually inserted into w, the volume of data downloaded onto w is incremented as:

$$Vgnd_{w,\mathbf{U}_o} \leftarrow Vgnd_{w,\mathbf{U}_o} + Vtot_o$$

Once w has been selected, an insertion position p is to be chosen. To find a good position, the heuristic chronologically traverses the download sequence $\mathbf{dSeq} = [o_1, \ldots, o_n]$. At each step, if o_i is the next acquisition to be downloaded:

- if there is an *idle period* before o_i and if there is an overlap between this idle period and the selected visibility window w, then the algorithm tries to insert o in w just before o_i;
- otherwise, if o_i is in the same visibility window w ($\mathbf{dw}_{o_i} = w$) and if o_i is the last observation download on w (or o_i is the last observation download in the plan, or $\mathbf{dw}_{o_{i+1}} \neq w$), then the algorithm tries to add o between the start of w and the start of o_i; downloads belonging to the same user are gathered to minimize the number of key changes; more precisely, the algorithms looks in the sequence for the first acquisition o' before o_i such that (a) o' is after the last considered idle period, (b) all the tasks between o' and o_i are in w, and (c) o' belongs to user $\mathbf{U}_o$; if such an acquisition o' is found, the algorithm tries to insert o just after o'; otherwise it tries to insert o after o_i;
- if the insertion succeeds, the exploring stops and the non-chronological greedy algorithm selects a new observation to download; otherwise, the chronological traversing continues.

If sequence **dSeq** has been integrally explored without any success at inserting, then the algorithm tries to insert o in w at the end of the download schedule. If it does not succeed, then another visibility window is selected. If all visibility windows have been tried, then the observation is labeled as non-downloadable, and the algorithm selects the next observation in ordered set O.

An example of how the insertion position is chosen is depicted on Fig. 5.5. On this figure, an observation o has already been selected for insertion. Fig. 5.5(a) gives the download schedule before the insertion of o.

On Fig 5.5(b), it is assumed that w_3 has been selected to download o. To choose an insertion position for o, the download sequence is chronologically traversed. The first idle period before o_1 is examined; because there is no overlap between this idle period and w_3, the sequence is traversed until w_3 is used or until the next idle period. Doing so, inserting o before o_6 is considered. Because there is an overlap between this idle period and w_3, the insertion is tried and it succeeds.

On Fig. 5.5(c), it is assumed that w_1 has been selected to download o. As in the previous case, the sequence is chronologically traversed. The idle period just before o_1 is first considered. But there is no overlap with w_1. The algorithm continues until $o_3 \rightarrow o_4$, as o_3 is the last observation download on w_1. The sequence is then traversed in the reverse order until it meets a task downloaded in w_1 and belonging to the same user. Observation o_2 belongs to the same user, the algorithm tries to insert o just after o_2. If the insertion succeeds, the algorithms goes on and selects a new observation. Otherwise, the traversing goes on until the second idle period, just before o_6. In that case, the algorithm tries to insert o in w_1 just before o_6. If it does not succeed, the algorithm tries to insert o in w_1 at the end of the download plan.

More globally, the H1-1 heuristic selects a visibility window such that ground transfer durations are minimized. Then it tries to insert the observation into an idle period or into an existing download window. Finally, observations that are destined to the same user are grouped to minimize the number of key changes.

[H1-2] version The second version of the H1 heuristic, aims at minimizing satellite use for downloading observations. This heuristic is adapted to a scenario where the satellite download capacity is limited. Basically, the H1-2 heuristic tries to insert downloads during high transfer rate periods, that is when the satellite is close to the considered ground reception station.

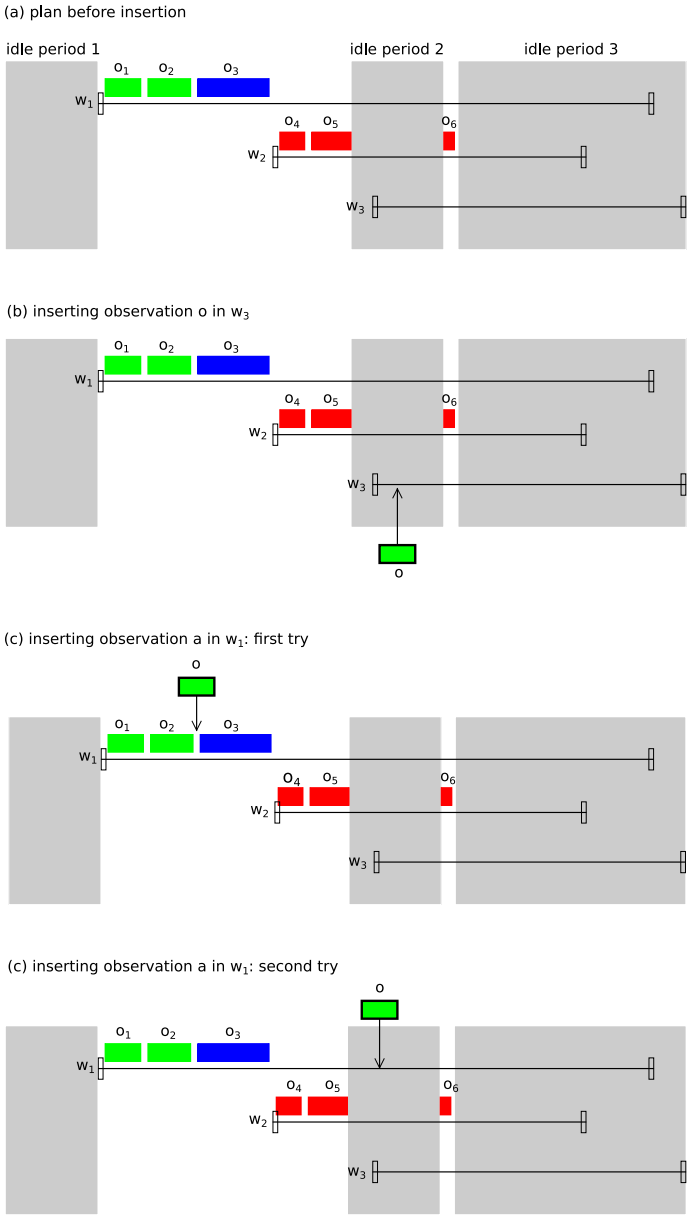


Figure 5.5: Example for the H1-1 heuristic.

Compared to the H1-1 heuristic which chooses first a window and then a position in the download sequence, the H1-2 heuristic chooses simultaneously a visibility window w and a position p. To make this choice (w, p), H1-2 chronologically traverses idle periods.

For each idle period P, the algorithm tries to insert o into P. More precisely, let o' denote the next observation just after P, if it exists. Then,

- the algorithm evaluates the insertion of o just before o' for all allowed visibility windows w that overlap P; this evaluation uses the metric $duVid_o$, an estimate of the download duration of o; the visibility window w which minimizes $duVid$ is kept, to consider first high transfer rate periods; let $duVid^*$ be the duration given by this *best* option; in what follows, any better option for inserting o must not decrease the idle period by a duration higher than $duVid^* + 1$ (1 is a penalty for depointing);
- the algorithm evaluates the insertion of o in download windows that are before the current idle period; to do this, the sequence of downloads is traversed in reverse order from o'; when there is a task o_i downloaded in a different visibility window w than the one used for o_{i+1}, the algorithm assess whether inserting o into w after o_i is better than the best option found so far; if it is the case, the current insertion position is kept;
- when the previous analysis has been done, the algorithm tries to insert the observation at the best position, and observations that are destined to the same user are grouped to minimize the number of key changes.

If an insertion has failed, the next idle period is analyzed. As soon as an insertion succeeds, the algorithm selects a new observation to download.

Fig. 5.6 shows an example of insertion positions considered by H1-2. The considered idle period is the one before o_6. The H1-2 heuristic tries to fill the idle period and to minimize satellite use.

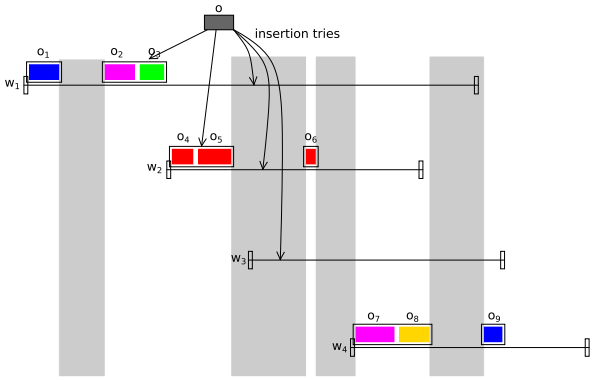


FIGURE 5.6: Example for the H1-2 heuristic.

5.2.3.2 Constraint checking

After each observation download insertion, the resulting download schedule can be checked with regard to all the temporal constraints. As already said in Sect. 4.2.5, the resulting temporal problem forms a TSTN. The InCELL library [Pralet and

Verfaillie, 2013a] is used to determine whether or not the TSTN is consistent and, in case of consistency, to compute the earliest and latest times of each temporal variable.

InCELL is a generic constraint-based evaluation library developed at Onera which allows constraint-based models to be formulated, non temporal constraints and criteria to be evaluated, and temporal (time-dependent or not) constraints to be propagated. Its has been built from the ideas of constraint-based local search [Hentenryck and Michel, 2005] and around the notion of invariant and of DAG (directed acyclic graph) of invariants. Any constraint optimization problem is modeled as a DAG of invariants with decision variables at the roots of the DAG and constraints and criterion at the leaves. DAG modeling allows any assignment of the decision variables to be evaluated (checking of the constraints and evaluation of the criterion). In the context of a greedy or local search, it allows any new (neighbor) assignment to be very quickly reevaluated thanks to an incremental reevaluation mechanism. This allows each local move to be quickly performed and a huge number of local moves to be performed in a limited amount of time. Moreover, InCELL includes several modeling and solving functionalities specifically dedicated to the management of (oversubscribed) scheduling problems: for example, existence of global (T)STN invariants in charge of the propagation of temporal constraints.

When a download schedule has been built, with all its constraints checked, an earliest schedule can be produced (see the example of Fig. 4.2). However temporal variables can remain unassigned and a partial order schedule defined by a precedence graph can also be produced to allow a flexible schedule execution (see Sect. 6.4).

5.3 Metaheuristic: Squeaky Wheel Optimization

The non-chronological algorithm takes an insertion order as input. This order can be determined either with a heuristic once for all or updated during search. One issue is that this insertion order has a great influence on the result and the heuristic can be bad. That is why an algorithm designed to avoid such a situation has been used: *Squeaky Wheel Optimization*, which is a general method for solving combinatorial problems [Joslin and Clements, 1999]. Basically, there are three components in an SWO algorithm :

- a *builder*, which consists in a greedy algorithm without backtrack; an ordered list of elements is used by this algorithm in order to allocate resources in an ordered manner;
- the *grader*; after a valid plan has been produced, the grader analyzes the plan and blames elements which deteriorate the most the solution; it generates a grade for each element: the higher the grade, the higher the deterioration;
- the *prioritizer*, which changes the order of elements in the list according to their grades; the more the grade of an element is high the more this element is moved forward in the list; the idea is to allocate resources earlier for elements which have a high grade because they deteriorate the most the solution.

SWO is primarily moving into the priority space. A small number of changes into the priority space may be a large step into the solution space. This characteristic is important for searching solutions for scheduling problem as this class of problems often have large plateaus in terms of criterion [Barbulescu et al., 2006]. Algorithm 6 shows how we build upon the non-chronological greedy algorithm presented in Sec. 5.2.3 to obtain an SWO search scheme.

For the first greedy search, observations are ordered in O' from the highest to the lowest priority level and, at the same priority level, from the biggest to the lowest

Algorithm 6: Squeaky Wheel Optimization

1 **Data** : O, the set of candidate observations over the planning horizon;

1. Init O', an ordering of O;
2. $i \leftarrow 0$;
3. While time is available :
   a) $i \leftarrow i + 1$;
   b) From O', build a plan P_i using a greedy algorithm (see Algorithm 5 and Algorithm 4);
   c) Save P_i;
   d) Analyze weaknesses of P_i;
   e) Modify O' by moving forward observations whose freshness is low in P_i;
4. Select P in $\{P_1, ..., P_i\}$ such that the optimization criterion is maximum.

weight, in order to favor high-priority acquisitions. In our case, the grade associated with each acquisition o for a plan P_i depends on the freshness of o in P_i. After each greedy search producing a plan P_i, the insertion order O', between acquisitions of the same priority, is updated by moving sequentially forward in O' acquisitions whose freshness is lower than the mean freshness. This move is proportional to the difference between both freshnesses.

To be more concrete, let $\mathbf{fr}_o$ be the freshness of o in a plan P. Let $\overline{f}$ be the mean freshness over all downloaded acquisitions. Let f^W and f^B be respectively the worst and best freshness in the plan, and let be i_p the index of o in the sub-sequence of priority p in O'. If $\mathbf{fr}_o < \overline{f}$, acquisition o is moved forward in O' and its new index i'_p in the sub-sequence of priority p is :

$$i'_p = \begin{cases} \left\lfloor i_p - \dfrac{\overline{f} - \mathbf{fr}_o}{f^W - f^B} \times i_p \right\rfloor & \text{if } \mathbf{fr}_o < \overline{f} \\ i_p \text{ otherwise} \end{cases} \tag{5.2}$$

This process is repeated as shown in Algorithm 6. The best plan over all the runs according to the optimization criterion (see Sect 4.2.4.2) is finally selected. An example can be seen on Fig. 5.7. This example aims at showing that the insertion order has a great important with respect to the solution quality. In this example, there are four observations $\{o_1, o_2, o_3, o_4\}$ to be downloaded. Each observation o_i may only be downloaded in one visibility window w_i. At each iteration, a chronological algorithm takes observations from the ordered list, puts them in the plan and builds a complete plan. In the first iteration, o_1 and o_2 are downloaded as soon as possible which results in preventing o_3 and o_4 from being downloaded. In iteration 2, the order is updated with respect to the absence of o_3 and o_4 in the schedule, moving them forward in the insertion order. In iteration 3, o_4 is inserted first which allows to download it during its only possible window.

The SWO scheme may be used with the non-chronological algorithm and stay very useful as the order of insertion is critical in this algorithm with regard the exploration of the solution space.

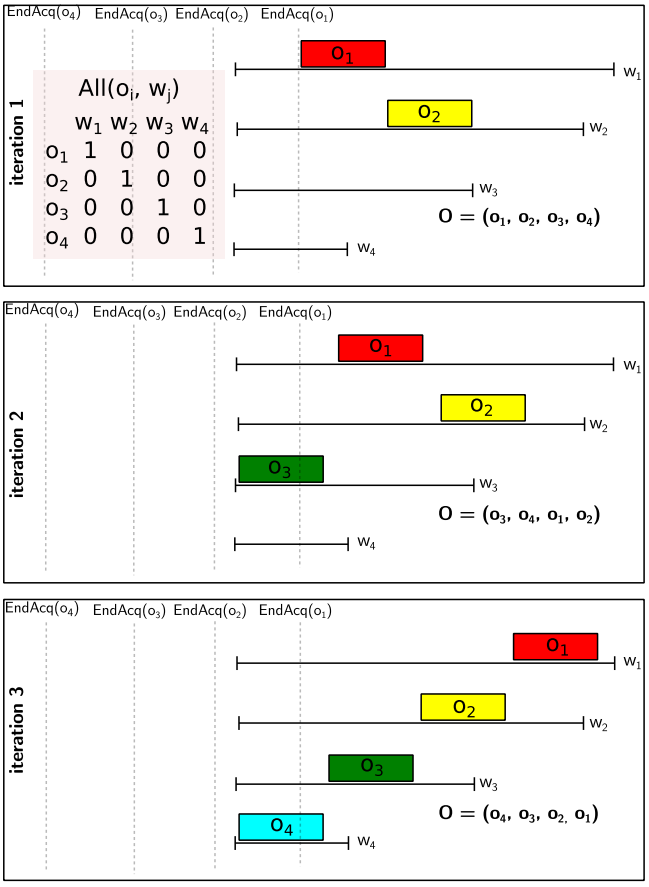


Figure 5.7: Three iterations of an SWO algorithm coupled with a chronological algorithm (macroscopic view without file downloads). $All(o, w)$ returns 1 if observation o is allowed on the station associated with download window w.

5.4 Analysis

Each algorithm has its advantages and drawbacks. The greedy chronological algorithm is faster and has a smaller memory consumption. It is because it only maintains a current state rather than maintaining a model of the problem over the whole planning horizon. On the negative side, building efficient heuristics for this type of algorithm is not always easy and the algorithm often ends up to be myopic.

The non-chronological greedy algorithm is slower and has a greater memory consumption but it is built upon a formal model which is easily updatable. In addition, it is less myopic that the chronological version because it is reasoning over the whole planning horizon. Typically, it produces plans with less go-and-back between ground stations and less key changes, thanks to the heuristic that groups downloads associated with the same user. By contrast with the chronological version, when inserting a download before existing downloads, it must often propagate and check the constraints through the whole horizon, which is a costly operation. Another drawback is the high dependence on the input ordered list.

That is why a Squeaky Wheel Optimization scheme has been developed upon the non-chronological algorithm. By updating the list after each iteration, it explores the solution space efficiently. However, it still depends on the internal heuristics of the algorithm. A way to deal with this would be to introduce *stochastic* parameters into the heuristics so that there is a probability to make another decision given the same initial state.

All these algorithms are used in the next chapter to design full decision-making schemes.

Flexible Scheduling for the Data Download Problem 6

In the previous chapters, we presented the data download problem and algorithms to solve its deterministic version. In this chapter, we present a flexible approach to share decision-making on data downloads between ground and board. For that we describe the decision-making chain organization from the ground planning to the onboard execution for several approaches: pure ground scheduling, pure onboard scheduling, and the contribution of this chapter, flexible scheduling.

6.1 Overview of the decision-making organization

In this section, we show how decision-making about observations and downloads can be organized in each of the three approaches we compare.

6.1.1 Overview of the pure ground approach

Fig. 6.1 shows the overall organization of decision-making in the pure ground approach, where all the decisions are made on the ground in the mission center. Such an approach is used to manage the currently operational Earth-observing Pleiades satellites.

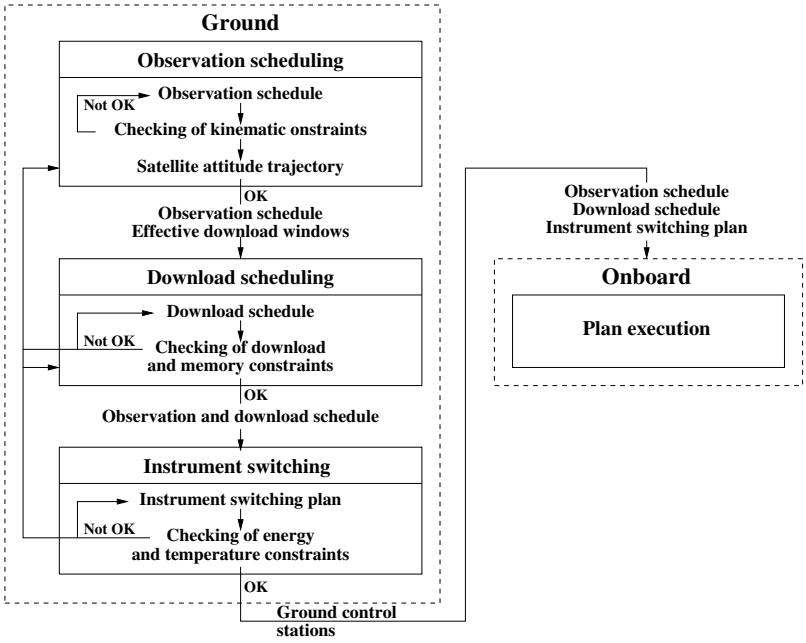


Figure 6.1: Overview of the decision-making organization in the pure ground approach.

Ground observation scheduling Roughly speaking, the ground builds regularly (typically every eight hours) observation schedules over some scheduling horizon (typically one day) by selecting and scheduling the observations to be done and checking the satellite kinematic constraints (satellite orbit and attitude movements). Once an observation schedule has been chosen with precise observation times, the associated satellite attitude trajectory can be built. This attitude trajectory allows the effective download windows to be determined for each ground reception station by removing from the station visibility windows the sub-windows over which download is impossible due to the current satellite attitude.

Ground download scheduling The ground uses the observation schedule and the effective download windows to build a download schedule over the same scheduling horizon by selecting and scheduling downloads to be done and checking the download and memory constraints (no memory overwriting). To be sure that download schedules be executable, maximum volumes are assumed for all the observations.

Ground instrument switching scheduling Once the observation and download schedules have been chosen with precise observation and download times, it is possible to build an instrument switching schedule (optical instrument and download antenna) and to check the energy and instrument temperature constraints, as well as

the instrument limitations in terms of On duration and number of On/Off over the scheduling horizon.

Onboard schedule execution Once all these schedules have been built on the ground, they can be sent to the satellite using any ground control station visibility window and then executed as they are, without any change.

The first justification for this decision-making organization is that one needs to know the observation schedule to build the satellite attitude trajectory, the effective download windows, and then the download schedule, and that one needs to know the observation and download schedules to build the instrument switching schedule. The second justification is that is has been observed on the currently operational Earth-observing Pleiades satellites that the limitations of the system in terms of observation capacity come in a decreasing order of impact from (1) the kinematic satellite attitude constraints, (2) the download constraints, (3) the energy constraints, (4) the instrument temperature constraints, and (5) the mass memory constraints. Checking first the most limiting constraints is known as a good practice.

6.1.2 Overview of the pure onboard approach

Fig. 6.2 shows the overall organization of decision-making in the pure onboard approach, where all the decisions about downloads are made on board. Such an approach has been explored in [Pralet et al., 2014c].

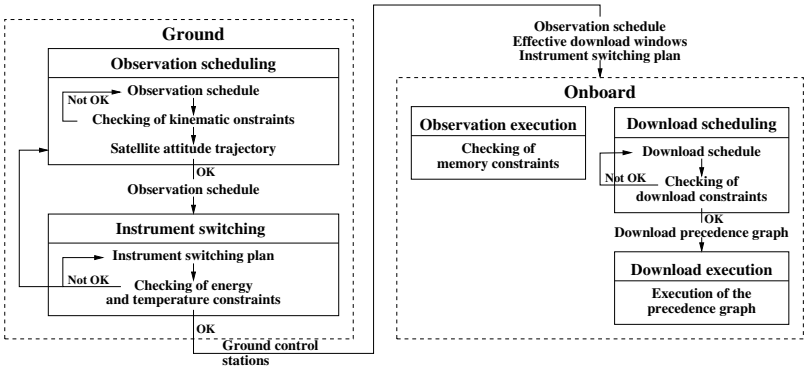


FIGURE 6.2: Overview of the decision-making organization in the pure onboard approach.

Ground observation scheduling Observation scheduling is performed on the ground in the mission center exactly as it is in the pure ground approach.

Ground instrument switching scheduling Because we do not know on the ground the downloads that will be decided on board, we make the pessimistic assumption that all the effective download windows will be entirely used for data downloading. On this basis, it is possible to build an instrument switching schedule and to check the

energy and instrument temperature constraints, as well as the instrument limitations in terms of On duration and number of On/Off.

Onboard download scheduling The observation schedule and the effective download windows can then be sent to the satellite using any ground control station visibility window and used for download scheduling. On board, because information about the actual volumes to be downloaded arrives as observations finish, download scheduling is performed just before any group of overlapping download windows (at the latest time to take advantage of the maximum information), with this group of windows as scheduling horizon. Two download windows are considered as overlapping if and only if they effectively overlap or there is not enough time between the end of the first and the beginning of the second to decompose the download scheduling problem into two subproblems. Exact volumes are considered for all the observations that are finished at the scheduling time and maximum volumes for those that will finish between the scheduling time and the end of the scheduling horizon in order to guarantee that download schedules be executable.

Onboard observation and instrument switching execution Observations are performed and instruments are switched as specified by the ground observation and instrument switching schedules. However, because the memory constraint cannot be checked on the ground (downloads not decided yet), a simple mechanism is used on board when executing observations in order to avoid any memory overwriting: if performing an observation o may lead to a memory overwriting, either o is not performed, or lower priority data is removed from memory to free space.

Onboard download execution Downloads are performed as specified by the successive download schedules, over the successive groups of overlapping download windows. However, to take into account the remaining uncertainty about the volumes of data produced by the observations that will finish between the scheduling time and the end of the scheduling horizon, the download schedules sent for execution are not fixed, but remain temporally flexible. Concretely, they have the form of a precedence DAG (Directed Acyclic Graph) where each node is executed as soon as all its predecessor nodes have been executed. This allows to take advantage of downloads whose duration is strictly smaller than the maximum duration that has been considered when scheduling and thus to start downloads earlier than in the pessimistic download schedule.

Keeping observation scheduling on the ground is justified by the fact that observation scheduling calls for time consuming computations of satellite attitude movements: movements to move from any observation to the following and precise movements to perform any observation with the required quality level. These movements are far more complex with agile satellites (movements of the whole satellite) than with previous generation non agile satellites, such as the first Spot satellites (movements of a mobile mirror). Engineers consider that the current technology does not allow these movements to be efficiently computed on board. On the contrary, moving download scheduling on board is justified by the fact that download scheduling is far less time consuming than observation scheduling is. This organization is also justified by the fact that uncertainty about volumes of data directly impacts download scheduling, but only indirectly observation scheduling.

6.1.3 Overview of the ground-onboard mixed approach

Fig. 6.3 shows the overall organization of decision-making in the ground-onboard mixed approach, where the decisions about downloads are shared between ground and onboard.

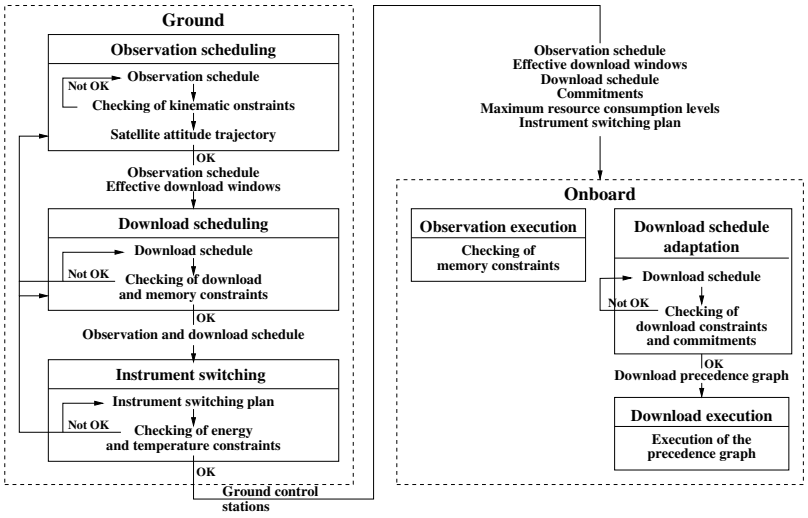


FIGURE 6.3: Overview of the decision-making organization in the mixed ground-onboard approach.

With each observation, is associated a priority level defined by an integer between 1 and **Np**, with 1 the highest priority level and **Np** the lowest. In the sequel of this chapter, we refer to as *high priority observations* all the observations of priority level 1 and as *low priority observations* all the others, of priority level greater than or equal to 2.

Ground observation scheduling Observation scheduling is performed on the ground in the mission center exactly as it is in the previous two approaches.

Ground download scheduling The ground uses the acquisition schedule and the effective download windows to build a download schedule over the same scheduling horizon by selecting and scheduling downloads to be done and checking the download and memory constraints as it is done in the pure ground approach. The only difference is that maximum volumes are assumed only for the high priority observations in order to guarantee that these observations be anyway downloaded. For the low priority observations, for which the same guarantee is not required, only expected volumes are assumed. With each high priority observation o, is associated a commitment of type 1 or 2 defined by the user who requires o. A commitment of type 1 associated with an observation means that it must be downloaded during the download window decided on the ground. In other words, on board, it can only be

moved forward in same window. A commitment of type 2 allows the observation to be downloaded during the download window decided on the ground or earlier, possibly in an earlier download window.

Ground instrument switching scheduling Because we do not know on the ground the downloads that will be finally decided on board (download schedules produced on the ground will be adapted on board), we make the pessimistic assumption that all the effective download windows will be entirely used for data downloading (same assumption as in the previous pure onboard approach). On this basis, it is possible to build an instrument switching schedule and to check the energy and instrument temperature constraints, as well as the instrument limitations in terms of On duration and number of On/Off.

Onboard download schedule adaptation The observation schedule, the effective download windows, the download schedule (including commitments to high priority observations), and the instrument switching schedule can then be sent to the satellite, using any ground control station visibility window. It can be used for onboard download schedule adaptation. On board, as in the previous pure onboard approach, download scheduling is performed just before any group of overlapping download windows with this group of windows as scheduling horizon, by considering exact volumes for all the observations that are finished at the scheduling time and maximum volumes for those that will finish between the scheduling time and the end of the scheduling horizon. The only difference is that download scheduling is not performed from scratch, but uses the download schedule built on the ground. It adapts this ground schedule while guaranteeing that all the commitments to high priority observations be met.

Onboard schedule execution Observations, instrument switchings, and downloads are executed on board exactly as in the previous pure onboard approach.

Fig. 6.4 offers another view of how decision-making is temporally organized between ground and onboard in the ground-onboard mixed approach.

The first justification for this decision-making organization with regard to the previous one is that, although download scheduling is far less time consuming than observation scheduling, it remains consuming if we consider the strictly limited computing resources available on board. As a consequence, this may be a good idea to take advantage of the computing resources available on the ground to build high quality download schedules and to use only simple decision rules to adapt them on board. These schedules are built by considering maximum volumes for high priority observations in order to guarantee their download, and expected volumes for low priority observations with the idea that higher volumes will be compensated by lower ones (see [Bonfietti et al., 2014] for a theoretical and experimental study of this phenomenon). The second justification is that such an organization increases the download predictability, at least for the high priority observations: for each of them, the user knows in which window or in which window at the latest it will be downloaded.

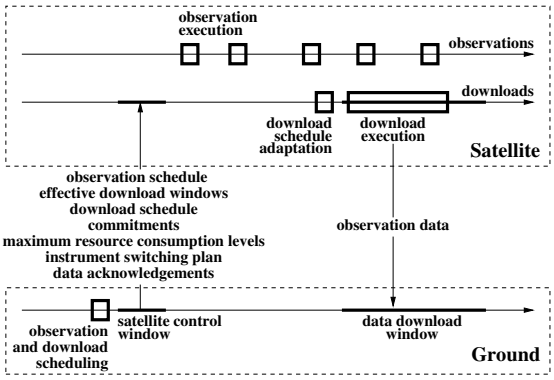


FIGURE 6.4: How decision-making is temporally organized between ground and onboard in the ground-onboard mixed approach.

6.2 Scheduling on the ground

Now we describe more in details each part of the flexible mechanism. Due to the huge number of acquisitions and files to be managed (> 1000 acquisitions, > 5000 files), the usage of exact optimal methods for planning downloads must be left out, even with the computational resources available on the ground. We chose to use the SWO algorithm that performs a sequence of non-chronological greedy searches as explained in Sect. 5.3.

Inconsistencies between ground and board Because plans are computed before a control window, data downloaded by the satellite since the end of the previous control window are not taken into account.

When a production center has successfully received an observation, a signal called *acknowledgment* is sent to the mission center to confirm the reception of the observation. Then, it is sent onboard to allow the satellite to delete the observation from its internal memory. Acknowledgments also allow the ground scheduler to remove the observation from the pool of observations to be downloaded. The signal can take some time to arrive to the mission center because of transmission delays on the ground network. This leads to inconsistencies between the onboard state and the ground state. For example, assume that the satellite has downloaded observation o_1 on station s_1, but the signal is not yet arrived at the mission center when the next download planning happens. o_1 may then be scheduled in the next download plan. More generally, there is an uncertainty about the state of observations (acquired, canceled, deleted, and downloaded) on the ground. Because such inconsistencies may appear, we need to have a decision rule onboard. Here, the onboard software is optimistic and we consider that if a download has been performed, it has succeeded. This way, when the onboard software gets a new download request for an observation that has been already downloaded, this request is ignored, except when a negative acknowledgement has been received.

6.3 Adapting data download schedules on board

In this section, which is the main technical contribution of this chapter, we show how, in the mixed ground-onboard approach, download schedules built on the ground can be adapted on board to take into account new information about the volumes generated by observations. Ground download schedules have been built with the assumption of maximum volume for all the high priority observations and of expected volume for the others. On board, for the observations that are finished at the scheduling time, the actual volume is known: it may be smaller for the high priority observations and smaller or greater for the others. For the observations that are not finished yet, we make the assumption of maximum volumes, whatever the priority level is, in order to guarantee that schedules be anyway executable (see Fig. 6.5 for an example).

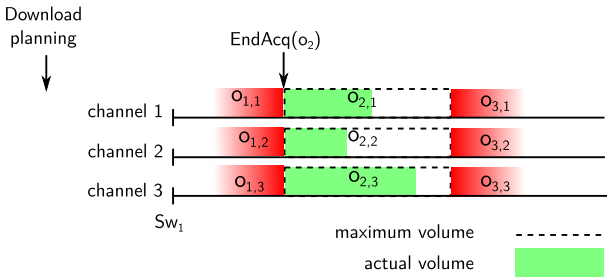


Figure 6.5: Example of an observation finishing during a visibility window and that is immediately downloaded. Because the volume was not known when planning on board, maximum volumes have been considered for all files.

The objective of onboard schedule adaptation is to adapt ground schedules as well and as efficiently as possible to this new data. We describe first how schedule adaptation can be prepared on the ground and then how it can be performed onboard.

6.3.1 Preparing download schedule adaptation on the ground

On the ground, it is possible to build download schedules as described in Chap. 5.

Let $\mathbf{HSeq}$ be the sequence of high priority observation downloads in the download schedule produced on the ground. Let $\mathbf{LHSeq}$ be the last observation download in $\mathbf{HSeq}$. For any high priority observation $o \in \mathbf{HSeq}$, let $\mathbf{NHSeq}_o$ be the next observation download in $\mathbf{HSeq}$ and $\mathbf{NsHSeq}_o$ be the sequence of observation downloads that follow o in $\mathbf{HSeq}$ (o included). Let $\mathbf{Ch}$ be the set of channels, $\mathbf{Mb}$ be the set of memory banks, and $\mathbf{Kct}$ be the key change table. Let $\mathbf{R} = \mathbf{Ch} \cup \mathbf{Mb} \cup \{\mathbf{Kct}\}$ be the set of resources required by any observation download. Channels and memory banks are unsharable renewable resources, whereas the key change table is a consumable/rechargeable resource.

From any ground download schedule, it is possible to compute for each high priority observation $o \in \mathbf{HSeq}$ and each resource $r \in \mathbf{R}$ a maximum resource consumption level $\mathbf{mrc}_{o,r}$ when starting downloading o which allows $\mathbf{NsHSeq}_o$ to be performed (download of o and of all the following high priority observations). If $r \in \mathbf{Ch}$, $\mathbf{mrc}_{o,r}$ represents the latest time at which the download of o can start

on channel r. If $r \in \mathbf{Mb}$, it represents the latest time at which the download of o can start from memory bank r. If $r = \mathbf{Kct}$, $\mathbf{mrc}_{o,r}$ represents the maximum number of key changes performed since the last key change table resetting when starting downloading o.

To compute these maximum resource consumption levels, one first builds the latest schedule **LatSch** including only the high priority observation downloads. The InCELL library can be used for that exactly as it has been used to produce the earliest schedule including all the observation downloads. Then, maximum resource consumption levels can be computed backwardly from the last to the first observation in **HSeq**. For each high priority observation $o \in \mathbf{HSeq}$ and each resource $r \in \mathbf{R}$:

- if $r \in \mathbf{Ch} \cup \mathbf{Mb}$ (channel or memory bank): if downloading o requires r, then $\mathbf{mrc}_{o,r}$ is the (latest) date at which the download of o starts using r in the **LatSch**; else $\mathbf{mrc}_{o,r} = \mathbf{mrc}_{\mathbf{NHSeq}_o,r}$ (latest date associated with the next download), except when $o = \mathbf{LHSeq}$ (the last); in the latter case, $\mathbf{mrc}_{o,r} = \mathbf{Ew}_{\mathbf{dw}_o}$ (end of the download window assigned to o);
- if $r = \mathbf{Kct}$ (key change table): if $o = \mathbf{LHSeq}$ (the last) or if the key change table is reset between o and $\mathbf{NHSeq}_o$, then $\mathbf{mrc}_{o,r} = \mathbf{Nkmax}$; else, if o and $\mathbf{NHSeq}_o$ are required by the same user (no key change), then $\mathbf{mrc}_{o,r} = \mathbf{mrc}_{\mathbf{NHSeq}_o,r}$ (maximum number of key changes associated with the the next download), else (key change) $\mathbf{mrc}_{o,r} = \mathbf{mrc}_{\mathbf{NHSeq}_o,r} - 1$.

Let us assume that, when starting downloading a high priority observation o, on each resource $r \in \mathbf{R}$, the consumption of r is smaller than or equal to $\mathbf{mrc}_{o,r}$. It is possible to follow the latest schedule **LatSch**. As a consequence, it will be possible to perform $\mathbf{NsHSeq}_o$ (download of o and of all the following high priority observations).

If we assume that Fig. 4.2 represents the ground download schedule, Fig. 6.6 shows the latest download start times that can be computed on each channel and each memory bank for the high priority observations C, B, and G. The maximum numbers of key changes performed since the last key change table resetting are respectively $\mathbf{Nkmax} - 1$, **Nkmax**, and **Nkmax** for the high priority observations C, B, and G, because the download of G is the last, the key change table is reset between the downloads of B and G (according to the key change table resetting policy), and C and B are required by two different users.

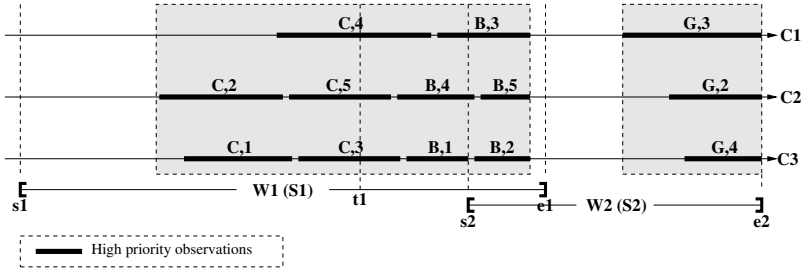


FIGURE 6.6: Latest download start times on each channel and each memory bank computed on the ground for the high priority observations C, B, and G.

Once these maximum resource consumption levels have been computed, the

download schedule sent to the satellite has the form of a sequence of quintuples, each quintuple $q = \langle o, \mathbf{dw}_o, \{\mathbf{ocSeq}_{o,c} | c \in [\![1, \mathbf{Nc}]\!]\}, \mathbf{Ct}_o, \{\mathbf{mrc}_{o,r} | r \in \mathbf{R}\}\rangle$ being made of an observation o to be downloaded, a chosen download window $\mathbf{dw}_o$, a sequence $\mathbf{ocSeq}_{o,c}$ of files to be downloaded on each channel c and, only for high priority observations, a type of commitment $\mathbf{Ct}_o$ and a maximum resource consumption level $\mathbf{mrc}_{o,r}$ on each resource r. Two kinds of commitment are possible: either to download o in $\mathbf{dw}_o$ (type 1) or in $\mathbf{dw}_o$ at the latest (possibly in an earlier window; type 2).

To this schedule, it is possible to add the list $\mathbf{CL}$ of the observations whose download has not been scheduled on the ground, but might be scheduled on board (in case of data volumes lower than expected). This list is ordered from the highest to the lowest priority level and, at the same priority level, from the greatest to the smallest weight.

6.3.2 Performing download schedule adaptation on board

On board, just before any group of overlapping download windows, download schedules must be produced with this group of windows as scheduling horizon. The algorithm we propose is a chronological algorithm, which makes decisions as time goes on by following the ground download schedule as much as possible and by modifying it either when this may be profitable (opportunities to move forward downloads or to add new downloads) or when this is necessary (need to remove downloads).

Let $\mathbf{RL}$ be the list of the observations whose download has been scheduled on the ground, but has been removed on board. This list is initially empty. Then, as downloads are removed on board and added to the list, it is ordered from the highest to the lowest priority level and, at the same priority level, according to the order of insertion in the list. Because, as we will see, high priority observation downloads will be never removed on board, $\mathbf{RL}$ contains only low priority observations.

6.3.2.1 Management of idle periods

Let $q = \langle o, \mathbf{dw}_o, \{\mathbf{ocSeq}_{o,c} | c \in [\![1, \mathbf{Nc}]\!]\}, \mathbf{Ct}_o, \{\mathbf{mrc}_{o,r} | r \in \mathbf{R}\}\rangle$ be the next element in the ground download schedule to be considered, empty if there is no element to be considered next.

Before trying to insert the download of o at the end of the current download schedule, one checks whether or not such an insertion would lead to a so-called idle period. An idle period is a time interval where download is possible, but nothing except antenna moves or key change table resettings has been scheduled on the ground. Three phenomena can lead to an idle period: by following the ground download schedule, one should keep doing nothing by waiting for the beginning of the download window $\mathbf{dw}_o$ assigned to o (case 1), for the end $\mathbf{Eo}_o$ of observation o (case 2), or for the end of the current download window if there is no next element to be considered. Cases 2 and 3 are illustrated in Fig. 6.7. Before inserting the download of observation B, one can observe an idle period due to the fact that one must wait for the end t_1 of observation B (case 2). After inserting the download of observation E (the last in the ground download schedule), one can observe another idle period due to the fact that one must wait for the end e_2 of window 2 (case 3). Idle periods that cannot be exploited on the ground may be exploited on board because of actual volumes that are smaller than expected.

In case of idle period, one tries to insert downloads in it, by considering first the observations in the list $\mathbf{RL}$ of the observations whose download has been removed

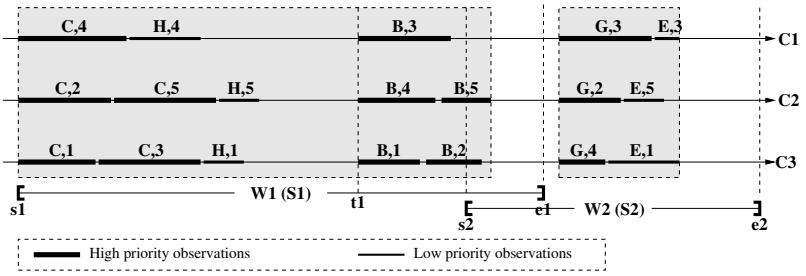


FIGURE 6.7: Two examples of idle period

on board, second the observations in the list **CL** of the observations whose download has not been scheduled on the ground, and third the observations whose download follows o in the ground download schedule (ordered from the highest to the lowest priority level and, at the same priority level, according to the ground schedule order). This is justified by the fact that we want to favor schedule stability and optimality: we want first to add as soon as possible downloads that have been removed on board, and second to add as many as possible downloads that have not been scheduled on the ground. If we insert an observation o' whose download has been previously removed on board, o' is finally moved backward with regard to the ground download schedule. If we insert an observation o' whose download has not been scheduled on the ground, o' is added. If we insert an observation o' whose download follows o in the ground schedule, o' is moved forward with regard to the ground download schedule. In case of a high priority observation, moving it forward is limited to the same window in case of commitment of type 1. Because high priority observation downloads will be never removed on board, only low priority ones can be moved backward in a latest download window.

In case of idle period, insertions are tried until no more insertions are possible. In Fig. 6.8, the download of observation D is added in the first idle period and the download of observation A in the second. When there is no idle period or no more insertions are possible, one tries to insert the download of o. If insertion fails, o is added to the list **RL**.

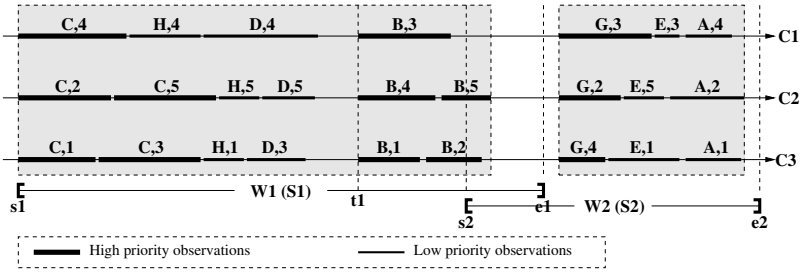


FIGURE 6.8: How idle periods may be exploited to add downloads.

6.3.2.2 Download insertion

When trying to insert the download of any observation o in a window w, one must first insert all its file downloads. If the download of o has been scheduled on the ground, one follows the sequences $\mathbf{ocSeq}_{o,c}$ of file downloads on each channel c that have been computed on the ground. Otherwise, the observation download insertion rule described in Sect. 5.2.1 is used. The earliest times are systematically chosen for file downloads. Then, only in case of low priority observations, two kinds of constraint must be checked: first the physical constraints and then the commitments to high priority observations.

To be sure that physical constraints be satisfied, it suffices to check that all the file downloads of o on every channel end before the end of w, taking into account the antenna move and the key change table resetting that may be necessary before downloading o. Fig. 6.9 shows an example where physical constraints are not satisfied. When trying to insert the download of observation E, it appears that is not possible on channel 3 to end the download of E before the end e_2 of window 2, because the size of file $(E, 1)$ is greater than expected on the ground. E is thus added to $\mathbf{RL}$.

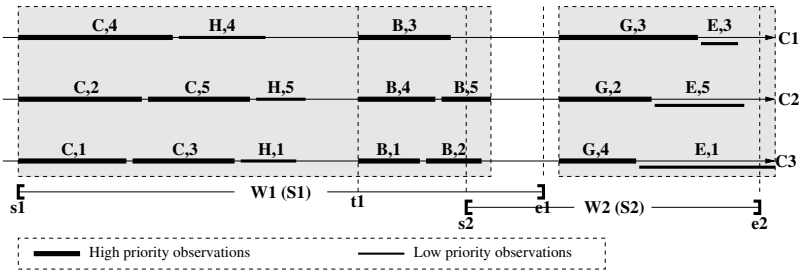


Figure 6.9: How physical constraints may be violated.

To be sure that commitments be met, it is necessary to check that it remains possible to insert all the high priority observation downloads that follow o in the ground download schedule. Checking that may be very time consuming. Fortunately, thanks to the maximum resource consumption levels that have been precomputed on the ground (see Sect. 6.3.1), it suffices to consider the first high priority observation o_1 that follows o in the ground download schedule, and to check that, after inserting the download of o in w, when starting downloading o_1, on each resource $r \in \mathbf{R}$, the consumption of r is smaller than or equal to $\mathbf{mrc}_{o_1,r}$.

In case of positive test result, we are sure that inserting the download of o does not endanger the download of o_1 and of all the following high priority observation downloads. The first reason for that is the existence of the latest schedule $\mathbf{LatSch}$ produced on the ground which guarantees that this is possible even if all the high priority observations produce their maximum volumes. The second reason is that it is always possible to build an earlier schedule which takes adavantage of earlier times and smaller volumes to be downloaded. The only question is about the activities whose duration is time dependent (downloads and antenna moves): if they are started earlier, can they end later? However, the answer to this question is negative: if any of these activities starts earlier, it may take more time but cannot end later. This is

obvious for download durations: if it is possible to download a file f from t_1 to t_2 and if we start downloading f at $t'_1 < t_1$, in the worst case, nothing is downloaded from t'_1 to t_1 and the file f is downloaded from t_1 to t_2; as a consequence, f will be downloaded at t_2 at the latest. This is also true for antenna move durations if we assume that the antenna is more agile than the satellite (able to be trained on a station in spite of any satellite attitude movement): if it is possible to move the antenna from station s_1 to station s_2 from t_1 to t_2 and if we start moving the antenna at $t'_1 < t_1$, in the worst case, the satellite remains trained on station s_1 from t'_1 to t_1 and the antenna movement is performed from t_1 to t_2; as a consequence, the movement ends at t_2 at the latest.

However, in case of negative test result, for two reasons, we are not sure that inserting the download of o really endangers the download of o_1 and of all the following high priority observation downloads. The first reason is that the maximum resource consumption levels have been computed on the ground with the assumption of maximum volumes for all the high priority observations. The second is that some of the high priority observation downloads that follow o in the ground download schedule may have been moved forward and have no longer to be considered. As a consequence, on every resource r, $\mathbf{mrc}_{o_1,r}$ is pessimistic: it is a lower bound on the actual maximum resource consumption levels.

If we really want to know whether or not inserting the download of o allows commitments to be met, an option is to recompute the maximum resource consumption levels for all the future high priority observations with the information about the actual volumes for those that are finished, as well as the information about forward moves. Because this may be time consuming, we propose a kind of lazy lookahead. Let $\mathbf{NsHSeq}_{o_1} = [o_1, \ldots, o_n]$ be the sequence of the high priority observation downloads that follow o in the ground download schedule:

- at the first lookahead step, we check that, by inserting the download of o and o_1 (with the new information about o_1), when starting downloading o_2, on each resource $r \in \mathbf{R}$, the consumption of r is smaller than or equal to $\mathbf{mrc}_{o_2,r}$; in case of positive answer, inserting the download of o is possible and if, on every resource $r \in \mathbf{R}$, the resource consumption level when starting downloading o_1 is greater than $\mathbf{mrc}_{o_1,r}$, it is used to update $\mathbf{mrc}_{o_1,r}$ (less pessimistic lower bound on the actual maximum resource consumption levels); in case of negative answer, we can still check further;
- if necessary, at the ith lookahead step ($i < n$), we check that, by inserting the download of o and o_1, o_2, …, and o_i (with the new information about o_1, o_2, …, and o_i), when starting downloading o_{i+1}, on each resource $r \in \mathbf{R}$, the consumption of r is smaller than or equal to $\mathbf{mrc}_{o_{i+1},r}$; in case of positive answer, inserting the download of o is possible and, for each observation $o_j \mid 1 \leq j \leq i$, if, on every resource $r \in \mathbf{R}$, the resource consumption level when starting downloading o_j is greater than $\mathbf{mrc}_{o_j,r}$, it is used to update $\mathbf{mrc}_{o_j,r}$ (less pessimistic lower bound on the actual maximum resource consumption levels); in case of negative answer, we can still check further by incrementing i;
- if necessary, at the last lookahead step, we check that it is possible to insert o and o_1, o_2, …, and o_n (with the new information about o_1, o_2, …, and o_n); in case of positive answer, inserting o is possible and, for each observation $o_i \mid 1 \leq i \leq n$, if, on every resource $r \in \mathbf{R}$, resource consumption level when starting downloading o_i is greater than $\mathbf{mrc}_{o_i,r}$, it is used to update $\mathbf{mrc}_{o_i,r}$ (less pessimistic lower bound on the actual maximum resource consumption levels); in case of negative answer, inserting the download of o is definitely

impossible and o is added to **RL**.

Performing lookahead until $i = n$ may be time consuming too. It must be however stressed, on the one hand, that lookahead may stop earlier in case of a positive answer and, on the other hand, it may be safely stopped at any step in case of negative answer: the download of o is not inserted and o added to **RL**, even if its insertion would have been finally possible. In any case, high priority downloads are not endangered. If lookahead is never stopped, it is said to be complete. Otherwise, it is said to be incomplete.

Fig. 6.10 shows an example where future high priority observation downloads might be endangered. When trying to insert the download of observation H, it appears that is not possible on channel 2 to start the download of B (the first high priority observation download that follows H in the ground download schedule) before its latest download start time, because the size of file $(H, 5)$ is greater than expected on the ground.

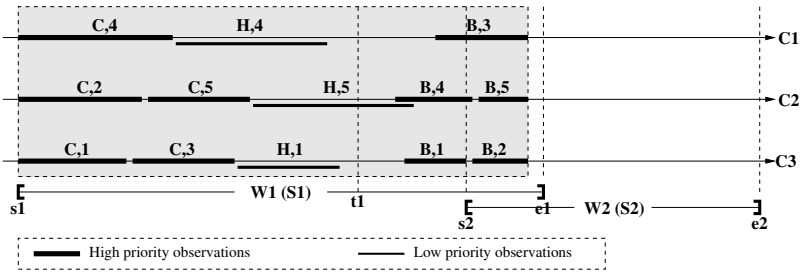


Figure 6.10: How high priority observation downloads might be endangered.

At the first lookahead step, nothing changes because observation B ends during window 1 and maximum volumes are assumed on board as well on the ground (no new information available on board about B). However, at the second and last lookahead step, it appears in Fig. 6.11 that, because of the actual volumes of observation G (the second and last high priority observation download that follows H in the ground download schedule) it is finally possible to insert H, B, and G. The download of H is thus inserted.

Fig. 6.12 gives a more macroscopic view of the onboard repair procedure. There are 8 observations to be downloaded. Only 6 of them have been scheduled in the ground plan $(o_1, o_2, o_3, o_4, o_7, o_8)$. Maximum volumes have been considered for high-priority observations and average for low-priority observations. Only latest start dates are shown at each iteration (and not for each channel). On board, only two volumes are different with respect to the assumptions made on the ground: o_1 is lower than maximum (green) and o_7 is higher than average (red). The onboard procedure follows the original ground plan. It manages to insert the download of observation o_5 by taking advantage of an idle period created by the smaller volume for o_1. At iteration 3, it tries to insert low-priority observation download o_7 but the latest start date check shows that it may endanger future high-priority downloads. Then, at iteration 4, a lookahead is performed and because the volume of o_3 and o_4 have not changed, o_7 is discarded. At iteration 5, the download of o_6 is introduced

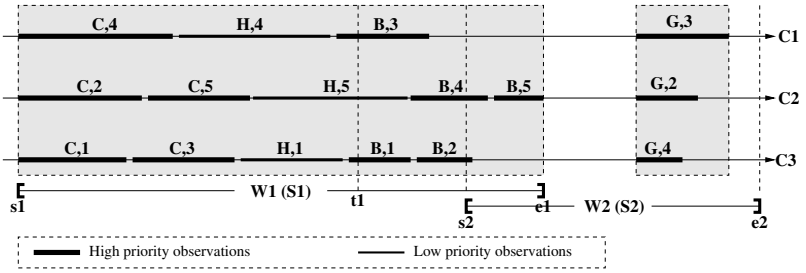


FIGURE 6.11: How high priority observation downloads are finally not endangered (H, B and G can be inserted.

in the idle period before the download of o_3. At iteration 6, the download of o_3 is inserted. Inserting the download of o_8 would endanger the download of o_4 so it is discarded. Iteration 7 presents the actual download plan.

6.3.2.3 Constraint checking

On the ground, the InCELL library is used to handle constraints (temporal or not). Because of the scarce computing resources available on board and of the large size of the InCELL models (need to manage a huge network of temporal constraints over the whole scheduling horizon), we did not consider the option of using InCELL on board. As a consequence, constraint checking is directly implemented in the onboard algorithms without any explicit model of the problem.

6.4 Executing on-board

6.4.1 Temporal flexibility at execution

Onboard planning algorithms build a consistent plan P from scratch or from a flexible plan produced on the ground. Once this consistent plan has been built, the only remaining source of uncertainty is the actual volume of the observations that finish between the end of the onboard planning phase and the end of the onboard planning horizon. To be able to take advantage of actual volumes smaller than maximum for these observations, we do not build a temporally fixed plan, but a temporally flexible plan $Flex(P)$ which takes the form of a *Partial Order Schedule* [Policella et al., 2004a] and more precisely the form of a precedence graph.

To obtain $Flex(P)$, because we use upper bounds on volumes (either maximum volumes for the not yet acquired observations or the real volumes for those that have been acquired), we can neglect exact start and end dates. This approach is close to the one developed in [Policella et al., 2009] where a complete RCPSP solution is first produced and then transformed into a POS for improving flexibility at execution.

6.4.2 Building a flexible plan from a complete plan

Translating a plan P into a temporally flexible plan $Flex(P)$ may be done by keeping only orders between activities [Pralet et al., 2014b]. More precisely, a flexible

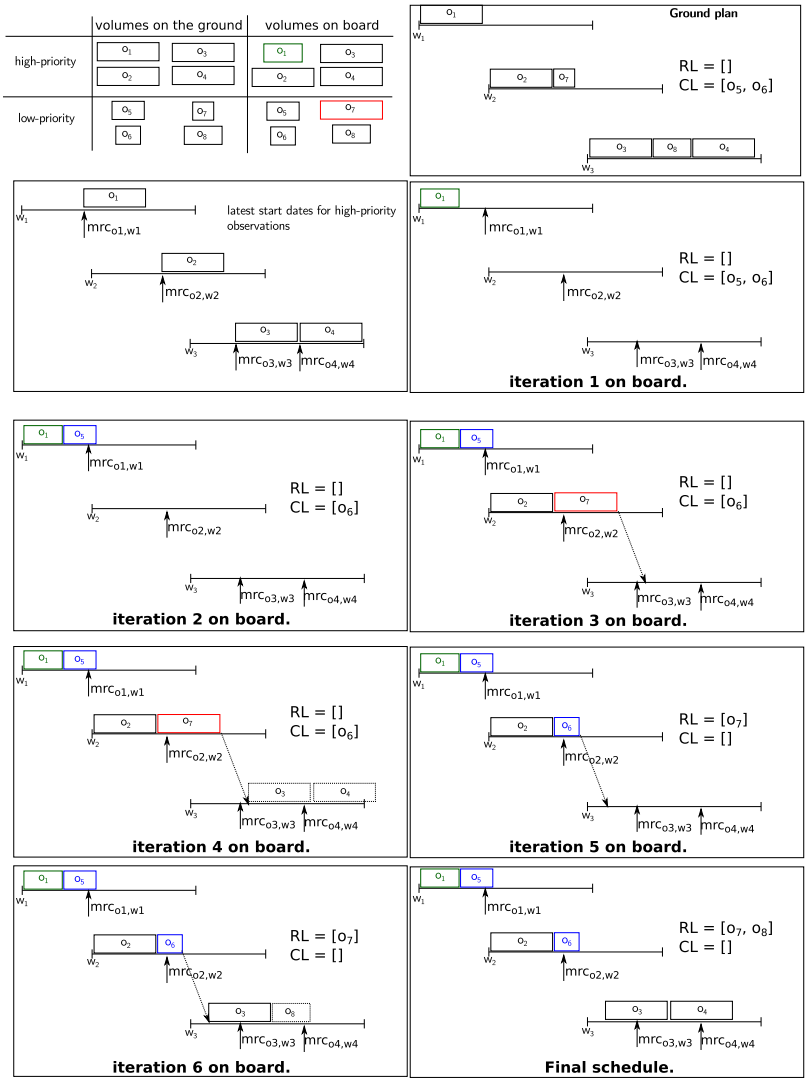


Figure 6.12: Iterations of the onboard greedy adaptation procedure; macroscopic view (without files and channels).

plan is represented as a Directed Acyclic Graph (DAG) where nodes represent all the possible events (start and end of acquisition and file downloads, start and end of download windows, start and end of antenna movements, start and end of key change table resettings, end of acquisitions, and key change times) and where edges represent all the precedences between events. Precedence constraints in the DAG specify that an activity a may be launched if and only if all its predecessors, i.e. all activities a' such that it exists an arc from a' to a in the DAG, are finished. In the following, we define as *key table usage window* the period of time during which downloads happen between two key table changes.

Fig. 6.13 gives an example of an instantiated plan that can be obtained with the algorithms described in previous chapters. There are 5 acquisitions downloaded (A, B, C, D, E) which all have 5 files ($(A,1)$, $(A,2)$, $(A,3)$, $(A,4)$, $(A,5)$ for A, ..., $(E,1)$, $(E,2)$, $(E,3)$, $(E,4)$, $(E,5)$ for E), which are stored on memory banks $M1$ to $M7$. There are two key table usage windows kt and kt' located respectively in two download windows $Vw1$ and $Vw2$. We consider that acquisitions A and B are already stored onboard when onboard planning is performed. The corresponding temporally flexible plan $Flex(P)$ is represented on Fig. 6.14.

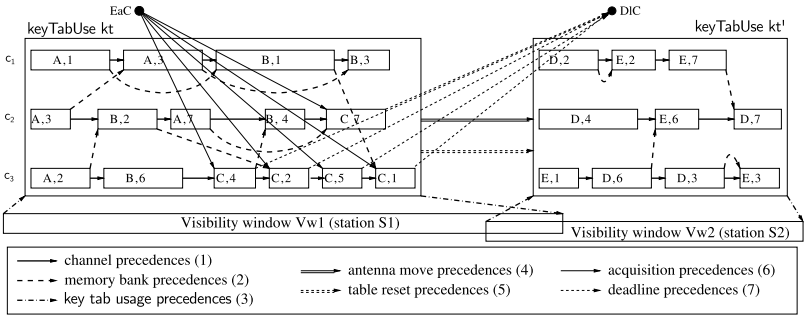


FIGURE 6.13: Completely instantiated plan P.

For convenience, we use the notation $\mathbf{cSeq}_{kt,c}$ as $\mathbf{cSeq}_c$ restricted to key table usage window kt and $\mathbf{mSeq}_{kt,m}$ as $\mathbf{mSeq}_m$ restricted to key table usage window kt. Formally, the flexible plan $Flex(P)$ is a DAG and each node of this DAG is labeled with an activity to perform. The DAG contains :

- nodes $WaitAcq(o)$, representing activities that wait for the end of observation o; a node $WaitAcq(o)$ is created for each observation o downloaded in P (i.e. such that $\mathbf{dw}_o \neq 0$) and not yet stored into the satellite mass memory; Fig. 6.14 shows nodes $WaitAcq(C)$, $WaitAcq(D)$ and $WaitAcq(E)$ for observations C, D, E whose download is planned although these observations are not yet stored on board;
- nodes $WaitVis(w)$, representing activities that wait for the beginning of a download window w; a node $WaitVis(w)$ is created for each download window w used in P for downloading acquisitions, that is for each download window w such that there exists at least one acquisition a whose assigned download window is w ($\mathbf{dw}_a = w$); see for example nodes $WaitVis(Vw1)$ and $WaitVis(Vw2)$ on Fig. 6.14;
- nodes $StaKtabUse(kt)$ and $EndKtabUse(kt)$, representing respectively activities for the start and end of key table usage window kt; a node

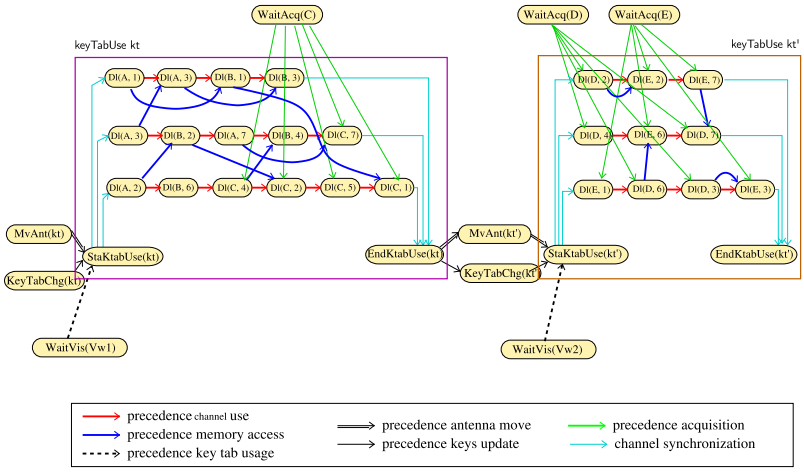


Figure 6.14: Temporally flexible plan $Flex(P)$.

$StaKtabUse(kt)$ and a node $EndKtabUse(kt)$ are created for each key table usage window; see for example nodes $StaKtabUse(kt)$, $EndKtabUse(kt)$, $StaKtabUse(kt')$ and $EndKtabUse(kt')$ associated with key table usage windows kt and kt' on Fig. 6.14;

- nodes $Dl(f)$, representing file download activities for a file f; a node $Dl(f)$ is created for each file f downloaded in plan P; in the example of Fig. 6.14, node $Dl(A, 1)$ represents the download of a file of A stored on memory bank 1;
- nodes $MvAnt(kt)$, representing antenna move activities for pointing towards the station associated with key table usage window kt; a node $MvAnt(kt)$ is created for each key table usage window iff there is actually a pointing operation towards another station between two key table usage windows; see for example nodes $MvAnt(kt)$ and $MvAnt(kt')$ on Fig. 6.14;
- nodes $KeyTabChg(kt)$, representing key change table update activities that happen before key table usage window kt; a node $KeyTabChg(kt)$ is created for each key table usage window; see for example nodes $KeyTabChg(kt)$ and $KeyTabChg(kt')$ on Fig. 6.14;
- nodes $WaitKeyChg(kt, c, k)$ (not represented on the figure for readability), representing key change activities for each key k on channel c in key table usage window kt; a node $WaitKeyChg(kt, c, k)$ is created for each key change; a key change is required for the first download in a key table usage window and between two file downloads that do not have the same key.

Activities in the DAG correspond (1) to event waiting activities (nodes $WaitAcq(a)$, $WaitVis(v)$, $WaitKeyChg(kt, c, k)$), (2) to synchronization activities (nodes $StaKtabUse(kt)$ and $EndKtabUse(kt)$) and (3) to execution activities, which launch a command (activities $Dl(f)$, $KeyTabChg(kt)$ and $MvAnt(kt)$). Internal data related to the activity are associated with each node (such as a file and a download channel for a file download node).

Several types of edges are connecting the nodes in the DAG:

- edges representing precedences on channel usage (red arcs on Fig. 6.14); these

edges between download nodes $Dl(.)$ are obtained directly from sequences $\mathbf{cSeq}_{kt,c}$ for each channel $c \in [\![1, \mathbf{Nc}]\!]$ and each key table usage window kt;

- edges representing precedences for reading access to memory banks (deep blue edges on Fig. 6.14); these edges between download nodes $Dl(.)$ are obtained directly from sequences $\mathbf{mSeq}_{kt,m}$ for each memory bank $m \in [\![1, \mathbf{Nm}]\!]$ and each key table usage window kt;
- edges representing precedences between observation acquisition and download (green edges on Fig. 6.14); for each observation a not yet stored on board and whose download is planned in P, an edge from $WaitAcq(a)$ to $Dl(f)$ is added for each file f of observation a;
- edges representing precedences between start dates of download windows and start dates of key table usage windows (dotted edges on Fig. 6.14); for each key table usage window kt in download window w, a edge is added from node $WaitVis(w)$ to node $StaKtabUse(kt)$;
- edges representing precedences between the start of key table usage windows and the start of file downloads (light blue edges on Fig. 6.14); for each key table usage window kt and each channel $c \in [\![1, \mathbf{Nc}]\!]$, an edge is added from node $StaKtabUse(kt)$ to node $Dl(f)$ with f the first file downloaded in sequence $\mathbf{cSeq}_{kt,c}$ (if it exists);
- edges representing precedences between the end of file downloads on channels and the end of key table usage windows (light blue edges on Fig. 6.14); for each key table usage window kt and each channel $c \in [\![1, \mathbf{Nc}]\!]$, an edge is added from node $Dl(f)$ to node $EndKtabUse(kt)$ with f the last file downloaded in sequence $\mathbf{cSeq}_{kt,c}$ (if it exists);
- edges representing precedences between the end of key table usage windows and the start of antenna movements for pointing towards the station linked to the next key table usage window (double line edges on Fig. 6.14); for each couple of adjacent key table usage windows (kt_{i-1}, kt_i), an edge is added from node $EndKtabUse(kt_{i-1})$ to node $MvAnt(kt_i)$ (if it exists);
- edges representing precedences between the end of antenna movements and the start of key table usage windows (double line edges on Fig. 6.14); for each key table usage window kt, an edge is added from node $MvAnt(kt)$ to node $StaKtabUse(kt)$;
- edges representing precedences between the end of key table usage windows and key change tables updates (black line edges on Fig. 6.14); for each couple of adjacent key table usage windows (kt_{i-1}, kt_i), an edge is added from node $EndKtabUse(kt_{i-1})$ to node $KeyTabChg(kt_i)$;
- edges representing precedences between the end of key change tables updating and the start of key table usage windows (black line on Fig. 6.14); for each key table usage window kt, an edge is added from node $KeyTabChg(kt)$ to node $StaKtabUse(kt)$;
- edges (not shown on the figure) representing precedences between key changes and file downloads; for each key table usage window kt and each channel $c \in [\![1, \mathbf{Nc}]\!]$, an edge is added from node $WaitKeyChg(kt, c, k)$ to node $Dl(f)$ with f the file such that the kth key change has to be done before f is downloaded.

6.4.3 Execution

The precedence graph is represented as a Directed Acyclic Graph $G = (V, E)$ with V a set of nodes and E a set of pairs of nodes called edges. A graph is acyclic iff, starting from any node, there is no path leading to that node. A node x is a predecessor of node y iff there exists an arc (x, y). A node may have several predecessors.

In this graph, we can distinguish controllable events (starts of activities which are under the control of the executive) and uncontrollable events (ends of activities which are under the control of the environment). With each node is associated a counter representing the number of its predecessors that are not finished yet.

This precedence graph is then executed with the following policy: any node is activated as soon as all its parents in the graph are executed, until all the events are executed. This policy may allow observations to be downloaded earlier than expected in the plan.

For three reasons, no event in the precedence graph can be triggered later than expected in the download schedule. The first reason is that the download schedules are pessimistic (assumption of maximum volumes for all the observations that are not finished yet). The second is, even if some activity durations (downloads and antenna moves) are time-dependent, if these activities start earlier, they may take more time but cannot end later (see Sect.6.3.2.2). The third is that each event is triggered as soon as all its predecessors in the graph have been triggered. This guarantees that all the precedence constraints that have been checked in the pessimistic schedule will be satisfied at execution. In particular, no file download will end after the end of the download window it has been assigned to.

Last, to model a download plan without temporal flexibility, the only thing to do is to add temporal waiting nodes in the DAG for each activity. That way, no download or antenna transition is moved forward.

6.5 Experiments

6.5.1 Approaches to be compared

To assess the positive impact of sharing decision-making between ground and onboard, we implemented and compared the following four approaches in the simulator presented in Appendix A:

1. scheduling on the ground (`Ground`) : download scheduling is performed on the ground with maximum volumes for all the observations, using an SWO algorithm built on top of a non-chronological greedy algorithm, as described in Sect. 5.3; this schedule is then executed on board; the only difference with the currently operational approach is that a precedence DAG built from the schedule allows temporal flexibility to be exploited on board and downloads to be started earlier than in the pessimistic ground schedule, as described in Sect. 6.4; this slightly favors the ground approach in terms of information age;
2. scheduling on board (`Board`) : download scheduling is performed on board from scratch before each group of overlapping download windows with exact volumes for the observations that are already finished and maximum volumes for the others, using a chronological greedy algorithm, as described in Sect. 5.2.2; as in the previous approach, a precedence DAG built from the schedule allows temporal flexibility to be exploited;

3. download scheduling is performed on the ground with maximum volumes for the high-priority observations and expected volumes for the others, using an SWO algorithm built on top of a non-chronological greedy algorithm, as in the ground approach; then, download scheduling is performed on board before each group of overlapping download windows with exact volumes for the observations that are already finished and maximum volumes for the others, by following and adapting the ground schedule, as described in Sect. 6.3; as in the previous approaches, a precedence DAG built from the schedule allows temporal flexibility to be exploited; two variants of the onboard adaptation mechanisms have been implemented:
   a) a very simple board adaptation mechanism (`SimpleRepair`): downloads of any priority level are removed from the ground schedule when they are physically unfeasible; this means that only physical constraints are considered; commitments to high priority observations are not considered;
   b) an intelligent board adaptation mechanism (`SmartRepair`): low priority downloads are removed from the schedule either when they are physically unfeasible or when they prevent future high priority downloads from being performed; downloads are also added or moved forward when possible; this means that commitments to high priority observations, as well as physical constraints, are considered; to check that commitments can be met, a complete lookahead mechanism is used; see Sect. 6.3 for details.

6.5.2 Scenarios

We use two realistic scenarios provided by Airbus Defence and Space (see their characteristics in Table 6.1). Both scenarios have been produced from the same actual one-day observation schedule built for a currently operational Earth-observing satellite (about 14 satellite revolutions around Earth over the whole day). They differ from each other according to the number of available ground reception stations: Scenario 1 (the least constrained) involves a large number of ground reception stations (23) resulting in many download windows (115), whereas Scenario 2 (the most constrained) involves a small number of ground reception stations (3) resulting in only few download windows (20). They also differ in the way observations are distributed between users and priority levels: 275 (resp. 247) high priority observations and 1089 (resp. 1117) low priority observations for Scenario 1 (resp. 2). Actual file volumes are randomly generated between $\mathbf{Vmax}/4$ and $\mathbf{Vmax}$, with $\mathbf{Vmax}$ the maximum file volume. 15 instances of actual volumes have been randomly generated and all the results are mean values over these 15 instances. Experiments were run on a Xeon 4-core 3GHz cpu with 8GB RAM, with Ubuntu 12.04 LTS.

6.5.3 Evaluation criteria

Execution results are compared according to several criteria:

1. the number of observation downloads, at each priority level;
2. the mean information age over all the downloaded observations, at each priority level;
3. the global criterion described in Sect. 4.2.4.2, at each priority level, computed with a sharing parameter equal to -3; because this complex criterion aggregates many different characteristics of a schedule (weights of the downloaded

	Scenario 1	Scenario 2
# of memory banks	5	5
# of channels	3	3
# of ground reception stations	23	3
# of download windows	115	20
# of users	5	5
# of priority levels	2	2
# of observations	1364	1364
# of high-priority observations	275	247
# of low-priority observations	1089	1117
# of generated files	6820	6820
# of volume instances	15	15

Table 6.1: Scenario characteristics.

observations, information age, sharing between users), its exact value has no real importance; what is important is only the fact that it may be higher with an approach than with another one;

4. the mean computing time consumed by onboard scheduling, over the successive groups of overlapping download windows;
5. only for the third and fourth approaches, the differences between the ground and onboard schedules: number of downloads that are added, removed, moved forward, and moved backward, at each priority level.

The first three criteria measure the quality of the execution results. The fourth one measures the cost of onboard scheduling. The fifth one measures the stability of the onboard schedules with regard to the ground ones.

6.5.4 Result analysis

Number of observation downloads With regard to the number of downloads (see Fig. 6.15), whatever the scenario is, high priority observations are all downloaded, except with the `SimpleRepair` approach which does not check whether or not future high priority downloads are endangered when inserting low priority ones. Low priority observations are all downloaded on Scenario 1, but not on Scenario 2. On the latter scenario (the most constrained), only the `SmartRepair` and `Board` approaches allow all of them to be downloaded.

Mean information age Significant differences appear in Fig. 6.16 only for low priority observations on Scenario 2 (the most constrained). Mean information age decreases from the `Ground` approach to the `Board` one, suggesting that the higher flexibility in onboard decision-making, the lower resulting information age.

Global criterion With regard to the global criterion, Fig. 6.17 shows that, for high priority observations on Scenario 1, the `Ground` and `SmartRepair` approaches are better than the `SimpleRepair` and `Board` approaches. For the `SimpleRepair` approach, this can be explained by the fact that it removes high priority downloads. For the `Board`

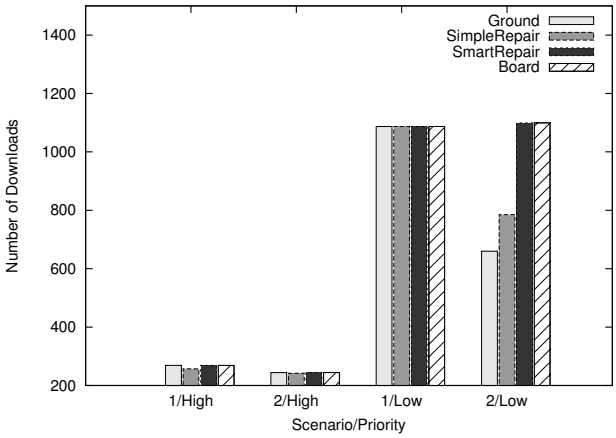


Figure 6.15: Number of downloads. The higher the better.

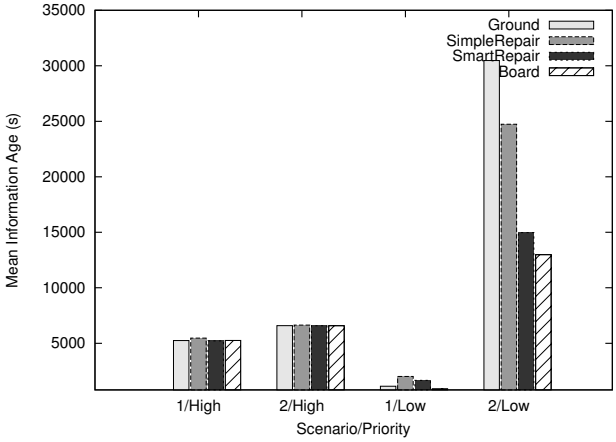


Figure 6.16: Mean information age (in seconds). The lower the better.

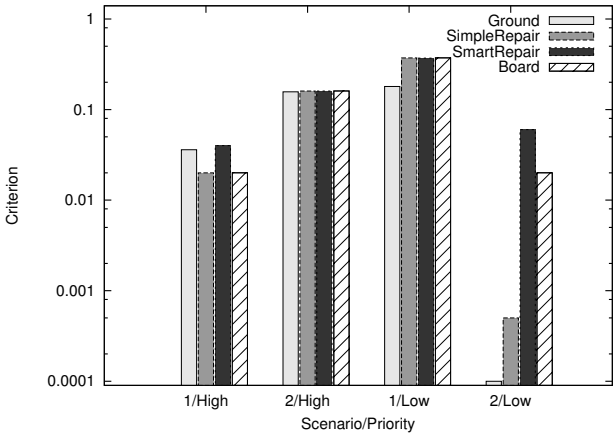


Figure 6.17: Criterion (on a logarithmic scale). The higher the better.

approach, this can be explained by the fact that it makes on board heuristic decisions over successive limited horizons (groups of overlapping download windows), whereas the the `Ground` and `SmartRepair` approaches work on the ground over a larger one-day horizon and explicitly optimize the global criterion thanks to the SWO algorithm. For low priority observations, the `SmartRepair` approach is clearly superior on Scenario 2 (the most constrained), suggesting a positive impact of the combination between a ground scheduling optimization algorithm over a large one-day horizon and an onboard schedule adaptation mechanism over successive limited horizons.

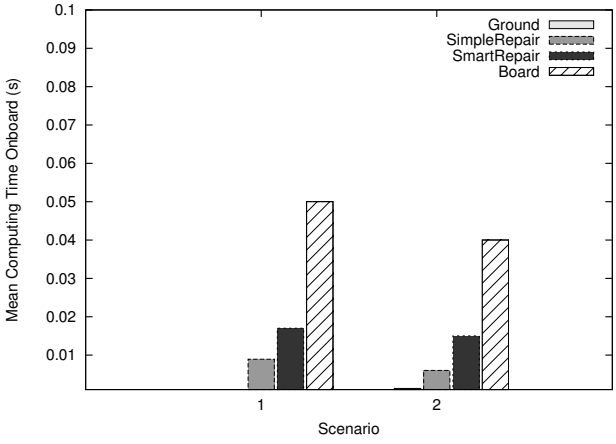


Figure 6.18: Mean computing time on board. The lower the better.

Computing time Unsurprisingly, results in Fig. 6.18 show that the higher flexibility in onboard decision-making, the higher computing time (the mean onboard computing time with the `Ground` approach is too small to be showed). It must be stressed that the `SmartRepair` approach is able to get similar or superior results in terms of number of observation downloads, mean information age, and global criterion with regard to the `Onboard` approach, with 1.5 to 3 times less onboard computing time. Experiments were run on a Xeon 4-core 3GHz cpu with 8GB RAM, with Ubuntu 12.04 LTS.

Stability Tables 6.2 and 6.3 show the mean number of movements (additions, removals, forward and backward movements of downloads) performed by the `SimpleRepair` and `SmartRepair` approaches. No download is added because they are all already present in the ground download schedule (due to expected volumes assumed for all the low priority observations). The `SimpleRepair` approach makes no distinction between high and low priority observations and removes high priority downloads: on average 11 on Scenario 1, and 2 on Scenario 2. On the contrary, the `SmartRepair` approach removes no high priority download. On Scenario 2 (the most constrained), the `SmartRepair` approach removes far fewer low priority observations (4) than the the `SimpleRepair` approach does (72). On both scenarios, the `SmartRepair` approach moves forward or backward many observations. As specified, it moves backward no high priority observation.

	Priority	SimpleRepair	SmartRepair
added	high	/	0
	low	/	0
removed	high	11*	0
	low	0	0
moved forward	high	/	5
	low	/	19
moved backward	high	/	0
	low	/	174

Table 6.2: Onboard movements over a one-day horizon for Scenario 1. Violated commitments are highlighted with *.

	Priority	SimpleRepair	SmartRepair
added	high	/	0
	low	/	0
removed	high	2*	0
	low	72	4
moved forward	high	/	9
	low	/	62
moved backward	high	/	0
	low	/	170

Table 6.3: Onboard movements over a one-day horizon for Scenario 2. Violated commitments are highlighted with *.

6.6 Discussion

6.6.1 Decisional flexibility

Decision	Variants				
	(A)	(B)	(C)	(D)	(E)
Download windows sequencing (1)	•	⋆	⋆	⋆	⋆
Assignment (2)	•	•	⋆	⋆	⋆
Scheduling (3)	•	•	•	⋆	⋆
Temporal problem (4)	•	•	•	•	⋆

Table 6.4: Decisonal flexibility approaches for the data download problem. ⋆ decision made on the ground; • decision made on board.

In Sect. 4.1, the data download problem has been decomposed into three subproblems: an assignment problem, a scheduling problem, and a temporal problem. At first, a fourth subproblem has been spotted, the problem of choosing the sequence of download windows to be successively used given a set of of overlapping visibility windows. Following this analysis and this decomposition, a *hierarchical decision-making* approach has also been explored [Maillard et al., 2014]. It consists in building *partial* plans on the ground and in *completing* them on board. More precisely, in this approach, the ground planner assigns only one or several sets of the decision variables. These sets of decision variables correspond to one or several subproblems (see Tab. 6.4 for the different variants of the flexible approach and Fig. 6.19 for an example of decomposition). This approach is close to the approach described in [Immorlica et al., 2004] which studies a flexible decision framework for stochastic combinatorial problems. Some of the decisions are made offline at a low-cost and some other decisions are made online with a more important cost.

To solve subproblems of the data download problem, we designed linear or quadratic programs from relaxations of the subproblems. These programs are then used in search schemes. Unfortunately, several aspects remain obstacles:

- criteria for evaluating solutions found for each subproblem independently are difficult to find as the impact on other subproblems is not measured;
- computing solutions to subproblems makes the search a bit myopic as it does not take into account other constraints;
- from a partial solution found on the ground, it is difficult to provide any guarantee for commitments on board as partial solutions often require a rebuilding phase where all constraints are taken into account (as the subproblems are relaxations of the real problems).

As a result, this approach has been discarded. However, it could be an interesting perspective for complex combinatorial problems in other industrial settings.

6.6.2 Missing evaluation and future works

In this part, we showed how uncertainty about the amount of data generated by observation can be managed by using a mixed architecture where decision-making about downloads is shared between ground and onboard and by using onboard fast and efficient schedule adaptation mechanisms. We also showed the operational advantages that could be provided by such a mixed approach, when compared to pure

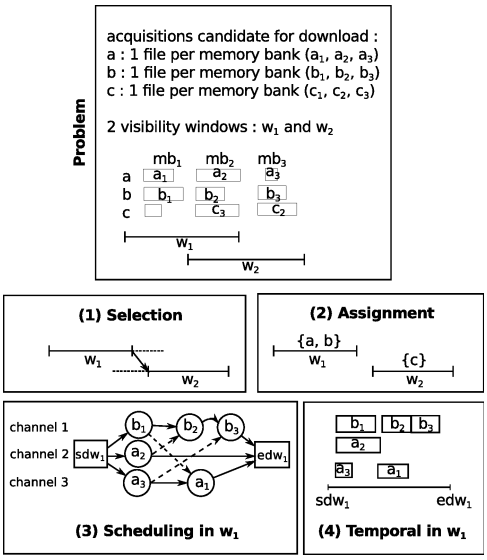


FIGURE 6.19: An example of the hierarchical decision-making approach. On the top part, a small data download problem involving 3 observations, 9 files and 2 visibility windows. On the bottom part, a solution to the 4 subproblems.

ground or pure onboard approaches: increase in the number of downloaded acquisitions, decrease in the age of the downloaded data, and increase in system predictability (stability of the download schedule), everything at an acceptable cost in terms of computing time on board.

The influence of several parameters has not been evaluated. For example, the duration of the planning horizon, on the ground as well as on board, may have an influence on the overall quality of the plans. It is usually good to have a large planning horizon in deterministic cases. In the case of problems with uncertainties, it may be good idea to balance the size of the planning horizon with the amount of uncertainties. Indeed, if there are too many uncertainties, the plans become quickly inconsistent at execution.

Also, we did not evaluate several algorithms (but we provided several heuristics for each algorithm developed) for computing schedules on the ground. This is not really in the scope of this work as we are more interested in looking at decision-making schemes rather than at specific algorithms. More robust methods involving statistical metrics for computing good baseline plans is left for future work.
In this direction, we could also evaluate the influence of the uncertainty model. For now, we assumed that uncertainty about file volumes was uniformly distributed over the interval of uncertainty. Other models could be evaluated, as well as the introduction of a weather forecast model. The latter would allow ground scheduling to be better informed.

Part III

Building conditional acquisition schedules

Introduction

In the previous part, we described how flexible download schedules are produced on the ground when volumes are uncertain, together with commitments to high-priority data downloads. These schedules are then adapted on board when exact volumes are known. The onboard adaptation process guarantees that ground commitments to high-priority downloads be satisfied, but guarantees nothing about low-priority downloads. It only tries to perform the latter as well as possible. With such an approach, the onboard adaptation process remains computationally light and the main computing effort is supported by the ground.

In this chapter, we switch to observation scheduling. More precisely, we present an approach that has been developed to perform observation scheduling under uncertainty about the real energy level available on board at execution time. In this approach, conditional observation schedules are produced on the ground where energy levels are uncertain, together with commitments to high priority observations. The onboard adaptation process guarantees that ground commitments to high-priority observations be satisfied: they will be performed exactly as foreseen in the ground schedule. However, it guarantees nothing about low-priority observations: each of them will be performed if and only if the current level of energy allows it and allows all the future high-priority observations to be performed as well.

This part of the work has been presented at the IJCAI'15 International Workshop on Planning & Scheduling for Space (IWPSS'15) [Maillard et al., 2015b].

Conditional observation scheduling 7

7.1 Informal description of the observation scheduling problem

The observation scheduling problem data consist of a set of observation requests $\{o_1, ..., o_n\}$ that are sent by users. The format of a request, which includes geographical data, has been briefly described in Sect. 2.1.4. The solution of the problem is a sequence of observations $[o_i, ..., o_j]$ such that all physical constraints are satisfied and the quality criterion is maximized. In the non-agile case, it is only a selection problem as the starting time of any candidate observation is fixed and incompatibilities between observations can be pre-computed. In the agile case, starting times of observations are not fixed because of the agility of the satellite. Instead, each observation has a realization time window. Observations that were not physically compatible in the non-agile case may become compatible even if realization windows overlap. In addition, preferences may be expressed for the actual realization time inside the window. These considerations add a highly combinatorial scheduling problem to the observation selection problem.

Fig. 7.1 illustrates the difficulty of computing a valid sequence on a simple example involving 5 observations o_1 to o_5. First we can observe that although observations are geographically ordered with respect to the satellite ground trajectory, it is possible to perform o_3 before o_2 thanks to the agility. In this case, for example, the satellite will point forward with respect to nadir for performing o_3 and then point backward for performing o_2. On Fig. 7.1(b), we note that because two observations are compatible (an arrow is linking them), does not mean they can always be put in an observation sequence with other observations before and after. For example, the realization windows of o_3 and o_2 are overlapping but it is not possible to perform this order if o_1 is performed just before o_3. Indeed, now that performing an observation has a direct influence on the satellite attitude profile, the realization time window available for an observation depends on the already selected observations and on the ability for the satellite to move quickly enough from one attitude to another.

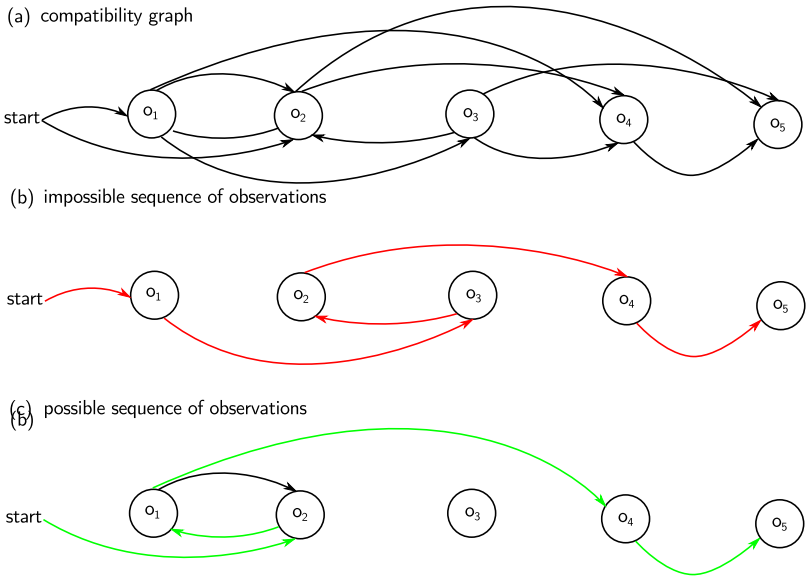


Figure 7.1: Compatibility graph and two explored sequences. An arrow linking two observations o_i and o_j means that the realization of o_i is compatible with the realization of o_j if they are alone in the plan. This illustrates how sequences may be explored and rejected because of agility constraints for example.

7.2 Current approach

In this section we recall some notions that have already been described in previous chapters.

The current approach for managing agile satellites consists in building offline on the ground an observation and download schedule, with precise activity start and end times, which verifies all the constraints related to satellite attitude, to observation and download, and to onboard energy, memory, and temperature. This schedule is then downloaded to the satellite using a ground control station visibility window and executed by the satellite without any change.

The current observation and download scheduling process is illustrated in Fig. 7.3. At the highest level, an observation schedule and then a download schedule are produced, taking into account user request priorities and high-level aggregated constraints such as no more observation time than a given threshold over each satellite revolution or each set of consecutive satellite revolutions. The proposed schedules are then checked according to all the constraints. In case of any constraint violation, they are modified by removing for example some observations or downloads. The process continues until a schedule that satisfies all the constraints is found. This process terminates because an empty schedule (no observation and no download) is always feasible if we assume a nominal satellite state and behavior.

To be sure that the proposed schedule is really executable by the satellite in spite of the numerous uncertainties, safety margins are used when checking constraints. For

example, the expected energy production and consumption rates are replaced by minimum production and maximum consumption rates, resulting in harder constraints. These margins are used in high-level constraint checking phases. In Tab. 7.1, we can see that the proportion of accepted observations over an initial pool of observation requests is low (about 14%). Among the causes of rejection, about 34% are due to energy and temperature high-level constraints. In Fig. 7.2, we can see the great difference between the expected energy profile computed with margins (in green) and the real onboard profile (in black). These facts motivate a more flexible decision-making approach to deal with these constraints. In this chapter, we focus only on the uncertainty about the onboard energy (uncertain productions and consumptions).

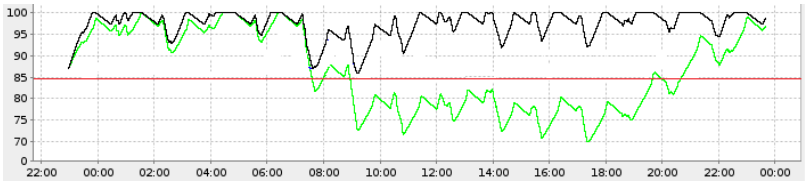


FIGURE 7.2: Satellite energy profiles (in percentage of the maximum energy level $Emax$) over a 24-hour scenario. The green line is the first ground prediction with margins produced by the current approach. The black line represents the actual energy profile resulting from the onboard execution of a conditional schedule. The red line is the minimum level of energy $Emin$, equal to $0.85 \cdot Emax$.

7.3 Some preliminary design choices

In the previous part, a flexible download scheduling approach has been proposed to deal with the uncertainty about the volume of data generated by observations. Roughly speaking, this approach replaces the ground checking of the download and memory constraints by a lighter checking which considers only high priority downloads. For low-priority downloads, decision is delayed online on board and allows the ground download schedule to be modified while guaranteeing ground commitments to high-priority downloads.

Our objective is to extend such an approach by proposing a flexible observation scheduling approach to deal with uncertainty about energy production and consumption. Whereas we focus on energy in this chapter, the approach could be applied to temperature constraints.

accepted	$\sim 14\%$
rejected - kinematic conflicts (agility constraints)	$\sim 51\%$
rejected - energy high-level constraints	$\sim 25\%$
rejected - temperature high-level constraints	$\sim 9\%$
rejected - other constraints	$\sim 1\%$

Table 7.1: Causes of observation rejection in the planning process. Evaluation on six 24-hour scenarios from Pléiades satellites (with one observation scheduling phase per day). The number of candidate observations varies from 1416 to 2560.

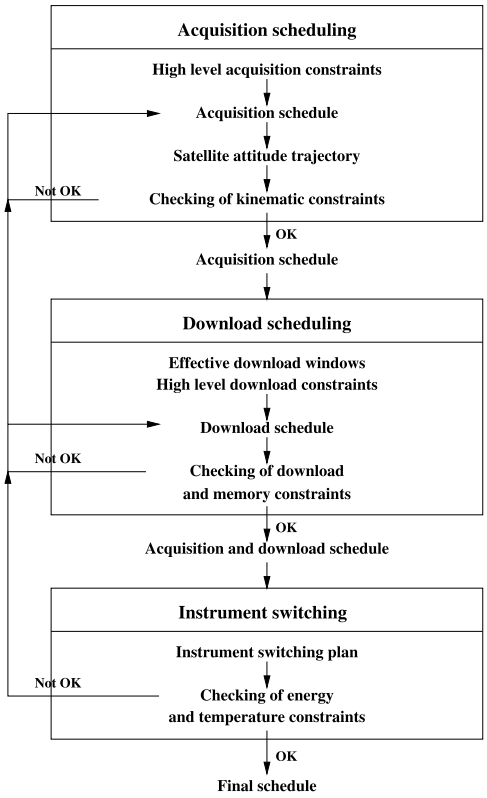


Figure 7.3: High-level view of the observation and download scheduling process.

One key aspect is that observation scheduling is far more complex than download scheduling, mainly because of the satellite attitude trajectories to be built and checked. Nowadays, space engineers consider that it would not be feasible to compute or modify such observation profiles online on board, mainly due to the very limited resources available on board satellites (from 1000 to 10000 times less resources than on a classical ground computer).

In such conditions, we choose not to modify on board the observation schedule that has been built on the ground. The only freedom we grant the onboard software is the ability, for any low observation in the ground observation schedule, to perform it or not. If the onboard software chooses to perform it, the observation instrument must be set ON. If it chooses not to perform it, the observation instrument is set in the SDBY or OFF mode depending on the duration between the current observation and the next one in the ground schedule: SDBY mode when this duration is less than or equal to a given threshold, and OFF mode otherwise. Both modes allow energy to be saved.

7.4 Related works

The problem of selecting and scheduling observations for Earth observation satellites has been extensively studied (see the survey by [Globus et al., 2004a] for example). Most of the time it is an oversubscribed scheduling problem [Barbulescu et al., 2006] where, in addition of a scheduling problem, observations have to be selected. Scheduling observations on agile Earth-observation satellites has been studied by [Grasset-Bourdel et al., 2012], where margins are used on the ground, or by [Beaumet et al., 2011] and [Pralet et al., 2013], where replanning is performed on board. One of the particularities in these studies is the time-dependent constraints induced by the platform agility. Authors have highlighted that this problem has constraints from AI (e.g. for the sequential decision part or the planning problem) and OR (for the pure scheduling part). However, these approaches are often pure ground or pure onboard approaches and while energy is always mentioned, the uncertainty about such parameters is not explicitly taken into account and remains a challenge for AI applications [Bresina et al., 2002a].

In [Schaffer et al., 2005], authors face uncertainty in action duration and resource consumption. They compute probability distributions for time and resource variables based on a user-supplied model of activities and resources. The distributions are then combined during planning to determine the probability distribution on the availability of a resource at any time point, which in turn may be integrated to yield the probability of violating any execution constraints on the resource. This probability is then used to score potential plans and to drive the planner's search toward low-risk actions. This technique makes plans more robust.

Our approach is very close to the approach developed by [Fox and Long, 2002] and [Gough et al., 2004]. In this approach, that is an alternative to the computationally expensive *contingent planning*, deterministic plans made with conservative estimates of a continuous resource are computed. Opportunistic branches are then developed based on the assumption that the continuous resource level will be higher at execution than predicted (see the shape of an opportunistic plan in Fig 7.4). When sufficient resource is available, a branch is developed in the plan. This branch will be used at execution if the level of resource available is sufficient. This approach, like ours, allows to avoid wastage of resource. Compared to this approach, the main difference is that our setting includes mandatory activities (high-priority observations).

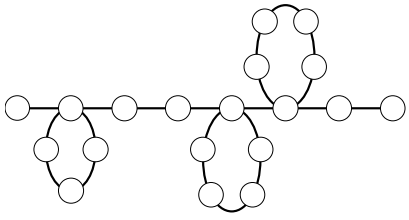


FIGURE 7.4: Opportunistic loop branches in a sequential plan. From [Gough et al., 2004].

7.5 Building a conditional observation schedule on the ground

Here, we assume that a sequence $Seq = [o_i, ..., o_j]$ enforcing agility constraints has been produced. A conditional observation schedule is a classical observation schedule, that is a sequence Seq of observations, where a *required level of energy* is associated with each observation. The required level of energy associated with an observation $i \in Seq$ is the minimum level of energy that is necessary to perform i and all the mandatory activities that follow i, that is all the high-priority observations that follow i in Seq and all the future required activities such as downloads. An example of conditional schedule is given in Fig. 7.5.

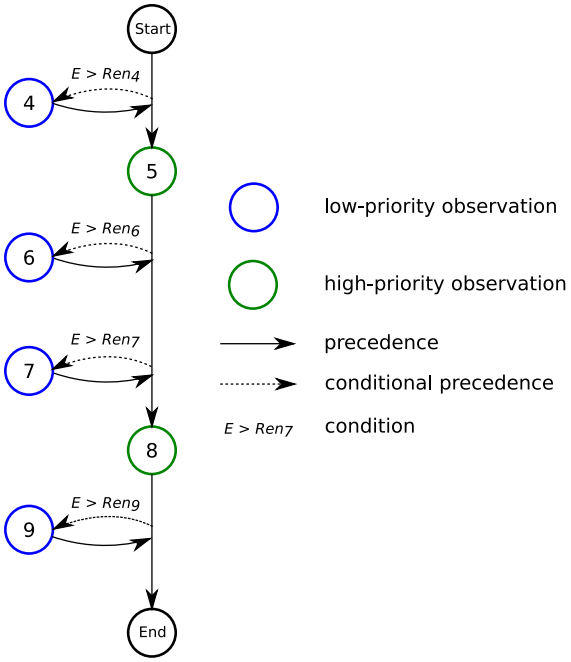


Figure 7.5: Conditional observation schedule. High-priority observations are mandatory and low-priority observations are executed only if the current onboard level of energy E is greater than the required level of energy Ren computed on the ground.

Because the download schedule is flexible (see previous part), we do not exactly know what will be actually downloaded (other than the certainty that high-priority downloads will be executed). To overcome this difficulty, we make a pessimistic assumption: all the download windows or at least those used to download high-priority observations will be completely used; hence, the antenna will be maintained ON over these windows and SDBY or OFF between these windows, according to the rules presented in Sect.7.3. Of course, when several download windows overlap, we assume that the instrument will be maintained ON only during the window resulting from their union.

Required levels of energy can be efficiently computed by a backward procedure which browses the sequence Seq of observations from the last to the first. The pseudo-code of this procedure is given in Algorithm 7 (Function **Flexible-energy-checking**). Its input is made of three elements:

- a sequence Seq of observations over a temporal horizon $[Hs, He]$ to which we add two fictive observations at the beginning and the end of the horizon (Hs and He), whose duration is null and priority is high; observations in Seq, including the two fictive ones, and are numbered from 1 to $|Seq|$; an observation instrument switching plan can be deduced from the sequence Seq by following the rules presented in Sect.7.3;
- a level $EnHs$ of energy at the beginning Hs of the horizon;
- a required level $REnHe$ of energy at the end He of the horizon; this level ensures that the satellite is in a safe state for future orbits.

Algorithm 7: Backward computing of required energy levels

```
1  Flexible-energy-checking(Seq, EnHs, REnHe)
2  begin
3      N ← |Seq|;
4      Ren_N ← REnHe;
5      j ← N;
6      for i = N − 1 to 1 do
7          ⟨Fea_i, Ren_i⟩ ← Required-energy-computing(i, j);
8          if Fea_i = failure then return ⟨failure, i⟩;
9          if Pr_i = high then
10             j ← i;
11         else
12             Mo_i ← Not-performed-mode(i);
13     if Ren_1 > EnHs then
14         return ⟨failure, 0⟩;
15     else
16         return ⟨success, 0⟩;
```

With each observation i in Seq are associated four parameters:

- its priority Pr_i, either high or low;
- its starting time St_i;
- its mode Mo_i, initialized at ON, but possibly modified by the algorithm;
- its required level of energy Ren_i, computed by the algorithm.

An execution of Algorithm 7 is presented in Fig 7.6. A subset of the sequence of observations is represented from Acq. 4 to Acq. 9. High (resp. low) priority observations are represented by bold (resp. thin) rectangles. For example Acq. 8 is of high-priority and Acq. 9 of low priority. The associated observation instrument switching plan is represented with segments of large (resp. intermediate and small) width for ON (resp. SDBY and OFF) modes. For example, the instrument is maintained in the SDBY mode between Acq. 7 and Acq. 8, but in the OFF mode between Acq. 8 and Acq. 9.

At step 1, we assume that the required level of energy at the beginning of Acq. 8 (of high priority) has been already computed and we compute the required level of energy at the beginning of Acq. 7 (of low priority), assuming the execution of Acqs. 7 and 8 (line 7 of Algorithm 7). A failure is returned when a failure occurs while

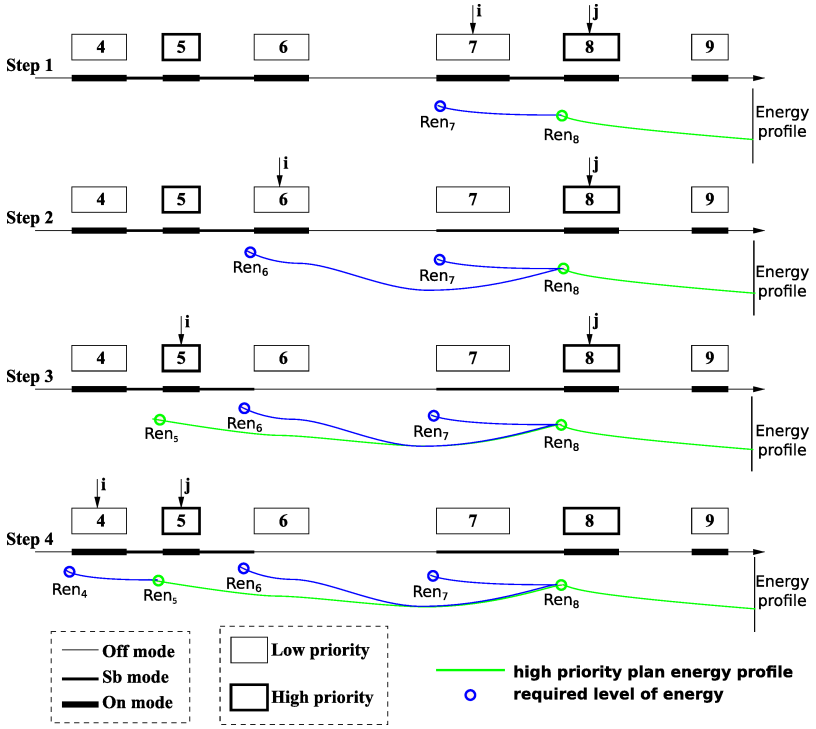


Figure 7.6: Illustration of the backward procedure used to compute required levels of energy.

computing (unfeasible observation; line 8). Then, at step 2, we compute the required level of energy at the beginning of Acq. 6 (of low priority), assuming the execution of Acqs. 6 and 8, but not of Acq. 7 which is of low priority and thus not mandatory (the mode of Acq. 7 is set to SDBY; line 12). The same way, at step 3, we compute the required level at the beginning of Acq. 5 (of high priority), assuming the execution of Acqs. 5 and 8, but not of Acqs. 6 and 7 (the mode of Acq. 6 is set to OFF). Because Acq. 5 is of high priority its required level of energy can be used as a reference level to compute required levels for the previous observations (line 10). This is what is done at step 4 where we compute the required level at the beginning of Acq. 4 (of low priority), assuming the execution of Acqs. 4, 5 and 8. When reaching the beginning of the planning horizon, a failure is returned when the required level of energy is greater than the initial level (unfeasible schedule; line 17). Otherwise, a success is returned.

Function **Not-performed-mode** returns the mode (SDBY or OFF) that has been initially planned just after Acq. i: we assume that, if i is not performed, the observation instrument will be set in that mode from the beginning of Acq. i in order to save energy.

Function **Required-energy-computing** computes the required level of energy at

Algorithm 8: Backward computing of required energy levels

```
1 Required-energy-computing(i, j)
2 begin
3     ren ← Ren_j;
4     t ← St_j;
5     while t > St_i do
6         ren ← max(Emin, ren − Power(t) · Δt);
7         if ren > Emax then return ⟨failure, ren⟩;
8         t ← t − Δt;
9     return ⟨success, ren⟩;
```

the beginning of Acq. i assuming that the required level of energy at the beginning of Acq. j (of high priority) has been already computed. Such a computation is performed backward step-by-step from the beginning of Acq. j to the beginning of Acq. i. It is performed step-by-step because the production and consumption of energy are variable. The production depends on the fact that the satellite is in eclipse or not and, if it is not in eclipse, on the satellite attitude. The consumption depends on the mode of the instruments (observation instrument and emission antenna) which may be ON, SDBY, or OFF. We assume a step of length Δt, small enough to consider that energy production and consumption are constant over any interval of length Δt.

The pseudo-code of this function is given in Algorithm 8. At each step, the required level of energy at time $t - \Delta t$ is computed from the required level of energy at time t already computed. Function **Power** returns the power produced between $t - \Delta t$ and t (assumed to be constant). This power takes into account the power production of the solar panels $\mathbf{Pp}_{solar}(t)$ and the power consumption $\mathbf{Pc}(t)$ of the observation instrument, of the emission antenna, and of the satellite platform. It is defined as:

$$\mathbf{Power}(t) = \mathbf{Pp}_{solar}(t) - \mathbf{Pc}(t) \tag{7.1}$$

Hence, it may be positive or negative. During day periods, power production $\mathbf{Pp}_{solar}(t)$ depends on the satellite attitude $att(t)$ at time t and $hpAtt(t)$ the optimal attitude where solar panels are directed towards the Sun:

$$\mathbf{Pp}_{solar}(t) = \max(0, P_{sun} \cdot \cos(angle(att(t), hpAtt(t)))) \tag{7.2}$$

The forward evolution of the energy level en on board the satellite from $t - \Delta t$ to t is given by Eq. 7.3, where we assume that $Emin \leq en(t - \Delta t) \leq Emax$ with $Emin$ and $Emax$ the minimum and maximum energy levels:

$$en(t) = \min(Emax, en(t - \Delta t) + \mathbf{Power}(t) \cdot \Delta t) \tag{7.3}$$

This equation takes into account that the energy level cannot be greater than $Emax$. A failure occurs when the energy level is lower than $Emin$ (unacceptable level).

From this forward equation, it is possible to deduce an equation which gives the backward evolution of the required level of energy ren on board the satel-

lite from t to $t - \Delta t$. If we assume that $Emin \leq ren(t) \leq Emax$:

$$\begin{aligned} ren(t-\Delta t) &= \min(e \mid & (\min(Emax, e + \textbf{Power}(t) \cdot \Delta t) \\ & & \geq ren(t)) \wedge (e \geq Emin) \\ &= \min(e \mid & (Emax \geq ren(t)) \\ & & \wedge (e + \textbf{Power}(t) \cdot \Delta t \geq ren(t)) \\ & & \wedge (e \geq Emin) \\ &= \min(e \mid & (e \geq ren(t) - \textbf{Power}(t) \cdot \Delta t) \\ & & \wedge (e \geq Emin) \\ &= & \max(Emin, ren(t) - \textbf{Power}(t) \cdot \Delta t) \end{aligned} \tag{7.4}$$

Eq. 7.4 is used at line 6 of Algorithm 8. A failure is returned when the required energy level is greater than $Emax$ (unsatisfiable requirement; line 7).

7.6 Executing a conditional observation schedule on board

On board, all high-priority observations are executed. Before each low-priority observation i, if the current level of energy is greater than or equal to the required level of energy Ren_i computed on the ground, observation i is executed. Otherwise the observation instrument is put into the mode (SDBY or OFF) that has been initially planned just after observation i in order to save energy (see Fig. 7.7 for an example).

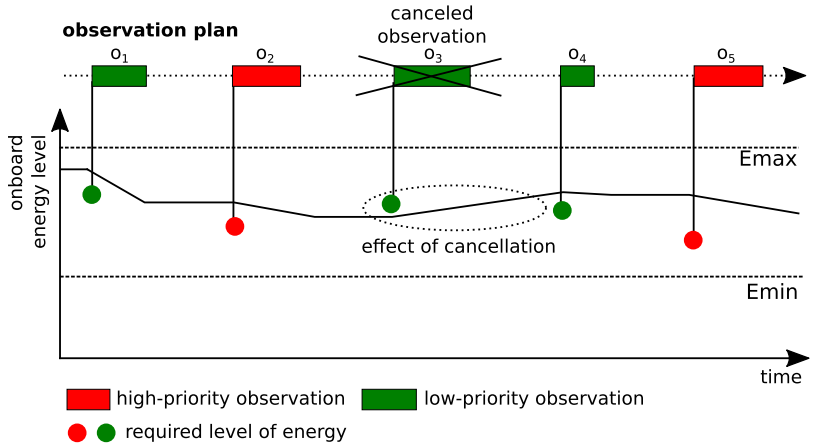


Figure 7.7: Observation cancellation and its effect on energy. On board, there is a required level of energy for each observation o_1, o_2, o_3, o_4, o_5. Before a low-priority observation such as o_3, if the onboard energy level is lower than the required level computed on the ground, the observation is canceled to save energy to ensure that future high-priority observations can still be executed.

As in the flexible approach for data download scheduling, we want to ensure that the mechanism allows all high-priority observations to be performed. First, at the end of the backward procedure, it is sure that the plan made only of high-priority

observations is executable as maximum energy consumptions and minimum energy production have been used for this subset of observations. If observation o_i at index i has high-priority, it is sure that all high-priority observations after o_i can be done. Now let us consider a low-priority observation o_{i-1} at index $i-1$. The required level of energy for o_{i-1} has been computed by going from the required level for o_i and by simulating backward from o_i to the start of o_{i-1}. All download windows have been accounted for. Then, if the onboard energy is greater than Ren_{i-1} at the start of o_{i-1}, it is sure that it is greater than Ren_i at the start of o_i. Then all future high-priority observations can be executed.

7.7 Experiments

In this section we evaluate the flexible observation scheduling mechanism proposed.

7.7.1 Approaches to be compared

To assess the positive impact of sharing decision-making between ground and onboard planning for observation scheduling, we implemented and compared the following three approaches into a satellite simulator we developed (details are given in Appendix A) and already used for experiments on the data download problem:

1. Planning on the ground with coarse energy checking (`GroundHiLevel`): observation planning is performed on the ground with an algorithm which verifies high-level constraints limiting the observation duration over each orbit to indirectly enforce energy constraints. If the schedule is inconsistent with these constraints, the least valuable observations are removed and the process is started again. On board, the plan is executed without any change. This method is close to the currently operational method.
2. Planning on the ground with fine energy checking (`GroundFine`): observation planning is performed on the ground with an algorithm which simulates the execution of the schedule step-by-step with maximum power consumption and minimum power production for all observations (high and low-priority). During simulation, when the onboard level of energy is strictly lower than $Emin$, low-priority observations are removed from the plan and the process is started again. On board, the plan is executed without any change.
3. Conditional planning (`Conditional`): observation planning is performed on the ground with an algorithm which computes backward, for all low-priority observations, the level of energy that is required to perform it and all the future high-priority observations (see Algorithm 7). During backward computing, if the required level of energy for an observation o is greater than $Emax$, meaning that more than the maximum level of energy is needed to perform o and the future high-priority observations, observations are removed from the plan and the process is started again. On board, low-priority observations are executed if and only if the current level of energy is greater than the required level computed on the ground.

7.7.2 Scenarios

We used one realistic scenario produced by Airbus Defence and Space. This scenario covers one day. It involves 5 users, 2 priority levels (high and low), 3 ground

reception stations, 20 visibility windows, and 1364 acquisitions (247 high-priority acquisitions and 1117 low-priority acquisitions).

Uncertainty Attitude $att(t)$ at time t is subject to errors $\epsilon(t) \in \mathbb{R}^2$ on pitch and roll angles. Let $\overline{att(t)}$ be the worst attitude at time t regarding the angle with the optimal attitude $hpAtt(t)$ where solar panels are directed towards the Sun (see Fig 7.8):

$$\overline{att(t)} = \underset{\|a - att(t)\| \leq \epsilon(t)}{\arg\max} (angle(a, hpAtt(t))) \tag{7.5}$$

The minimum power production $\underline{\mathbf{Pp}_{solar}(t)}$ at time t is given by the following equation:

$$\underline{\mathbf{Pp}_{solar}(t)} = \max(0, P_{sun} \cdot \cos(angle(\overline{att(t)}, hpAtt(t)))) \tag{7.6}$$

Given the ϵ uncertainty, $\underline{\mathbf{Pp}_{solar}(t)}$ is estimated to be close to $0.75 \cdot \mathbf{Pp}_{solar}(t)$.

We also consider that nominal power consumption $\mathbf{Pc}(t)$ by the platform, the observation instrument and the emission antenna is uncertain. We evaluated the same scenario with three uncertainty hypotheses: maximum consumptions 10%, 20%, or 30% higher than the nominal consumption:

$$\overline{\mathbf{Pc}(t)} = (1 + \alpha) \cdot \mathbf{Pc}(t), \alpha \in \{.1, .2, .3\} \tag{7.7}$$

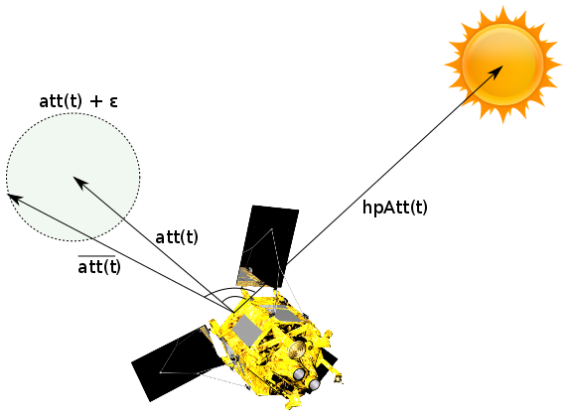


Figure 7.8: An illustration of the attitude uncertainty and of its influence on energy production.

7.7.3 Results

We measured only the number of acquisitions performed. In Tab. 7.2, we see that whatever the approach, all the high-priority acquisitions are executed. Concerning the low-priority acquisitions, we can see that the `Conditional` approach allows the

Priority	Uncertainty (α)	GroundHiLevel	GroundFine	Conditional
High	10%	247	247	247
	20%	247	247	247
	30%	247	247	247
Low	10%	976	1053	1086
	20%	938	1010	1061
	30%	898	927	1040

Table 7.2: Number of acquisitions that are really executed.

highest number of low-priority acquisitions to be executed. Without any surprise the GroundFine approach which simulates finely the energy profile is more effective and allows more low-priority acquisitions to be executed than the GroundHiLevel approach which enforces stronger high-level constraints. Also, we see that the more the uncertainty is large on power production and consumption, the more the conditional approach is relevant.

Fig. 7.9 shows the onboard energy levels during a 24-hour scenario from different perspectives. The GroundFine method computes a first predicted energy profile (dotted line on the figure) based on worst energy consumptions and productions. A procedure then removes acquisitions until this profile does not go under $Emin$, that is 85% in this case. We see that the predicted and real profiles produced by the Conditional approach are quite close at the beginning and at the end of the day because acquisition and download activities mainly occur between 6am and 6pm in this scenario. We also see confirm that the onboard level of energy never goes under the minimum level $Emin$ during the execution of the conditional schedule.

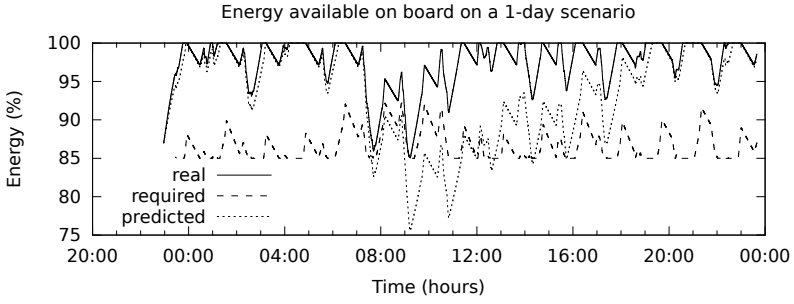


Figure 7.9: Onboard energy levels during a 24-hour scenario.

This evaluation has been done with only one real scenario, it would be interesting to demonstrate the robustness of the approach on other real-world scenarios and with other physical settings. Experiments were run on a Xeon 4-core 3GHz cpu with 8GB RAM, with Ubuntu 12.04 LTS.

7.8 Future works

Coupling with the data download planning The acquisition scheduling problem is strongly connected with the data download problem (see Fig. 7.3). In the flexible approach, we make the strong assumption that all download windows are used for downloading. Whereas this is close to reality in over-subscribed scenarios such as defense scenarios where there are few ground stations and then few visibility windows, it is far from true in civil-oriented scenarios where there are a large number of ground stations that cover all continents and are generally underused. This assumption makes the acquisition schedule very pessimistic. Work remains to be done to use results from the download planning phase during the computation of required energy levels. One possibility would be to consider that downloading high-priority acquisitions is the only mandatory requirement. In that case, only the visibility windows needed for that would be taken into account when simulating the schedule. Following the same `Conditional` approach, required energy levels would be computed on the ground for each group of visibility windows (a group constitutes an onboard download planning horizon). On board, before any of these groups, if the actual level of energy is sufficient, then the whole group could be used for downloading. Otherwise, only high-priority downloads would be executed to save energy.

Temperature As said, this conditional mechanism can be easily extended to take the uncertainty about onboard temperature into account. A fine temperature model is not available on the ground when planning. High-level constraints are then used during the planning process. In Tab. 7.1, we see that 9% of the observations are rejected during the planning phase because of these high-level temperature constraints.

Computing observation profiles on board As already said in a previous chapter, computing observation profiles and more generally attitude trajectories can be modeled as a continuous CSP [Beaumet et al., 2007]. Space engineers consider that the current technology does not allow observation profiles to be efficiently computed on board because of the scarce computing resources. That is why observation scheduling is kept on the ground. Being able to compute these profiles onboard would allow to design the same kind of mixed decision-making mechanism as in the data download problem (as it has recently been done for the EO-1 satellite [Chien and Troesch, 2015]). If onboard computation is not possible, it is still manageable to compute on the ground conditional observation plans with associated profiles for each branch.

Conclusion and perspectives 8

In this manuscript, we have tried to show the possibilities and benefits of a decision-making mechanism that is shared between ground and onboard for an agile Earth-observation satellite.

After having introduced some preliminaries about decision-making under uncertainty, it was observed that the current methods for handling uncertainty in Earth-observation missions was either computing robust low-quality plans on the ground or unpredictable high-quality plans on board. More general methods from combinatorial optimization were shown and their limitations have been pointed out with regard to our problems.

The main contribution of this thesis is the development of flexible decision-making mechanisms. It is inspired by both proactive and reactive methods. In this flexible approach, differently from pure ground or pure onboard approaches, the available computing power on the ground is used to produce high-quality robust plans together with additional features allowing to quickly adapt plans on board where uncertainty is low. Advantages from both planning phases, offline and online, are being used to balance efficiency and predictability.

Such an approach has been first applied to tackle the dynamic data download problem with uncertainty about file volumes. This problem has been modeled as a discrete optimization problem. The latter contains some features which are close other combinatorial problems such as multi-knapsack problems or job-shop scheduling problems but it also contains application features which prevented us from directly reusing existing search schemes.

Exact and approximation algorithms have been used to solve the deterministic version of this data download problem, that is the version without uncertainty. Exact methods has not seemed scalable to real instances and approximation algorithms have been able to produce high-quality solutions.

In the flexible decision-making mechanism developed, a metaheuristic builds a download schedule on the ground as well as bounds on resource consumption for all high-priority observations. This schedule is produced on a deterministic version of the problem where mean volumes are used for low-priority observations, which allows to schedule more downloads than in the pure ground approach, and maximum volumes are used for high-priority observations. An onboard chronological adaptation mechanism is used (1) to quickly repair the schedules if real volumes of low-priority observations endanger future high-priority observations, (2) to insert new observation downloads if possible, (3) to move observation downloads forward in the schedule if possible. Various algorithms have been implemented into a satellite simulator and experiments have been conducted to compare this flexible mechanism with its pure ground and pure onboard counterparts. In the end, flexible scheduling is able to perform as well as pure onboard scheduling on most criteria and to outperform it on the sharing of the system.

The observation planning problem has been investigated as well and the same decision-making sharing mechanism has been applied to tackle uncertainty on energy production and energy consumption on board the satellite. However in this case, plans are not adaptable plans but conditional plans containing branches depending level of energy available at execution time. In the same spirit as in the previous problem, the idea is to perform the maximum number of low-priority activities while ensuring that future high-priority activities will be executable with regard to their

resource consumption. Experiments have showed that the mixed approach is very promising against existing approaches.

Perspectives

Future works have been pointed out in previous chapters. We summarize them and point out other more general perspectives.

Data download scheduling problem

- Approximation algorithms such as the SWO scheme are used to compute data download schedules. Estimating their efficiency by comparing the quality of their solutions to optimal solutions would be interesting. As it seems quite difficult to compute optimal solutions for real-life scenarios, there are two options which are not perfect but satisfying enough to get an idea. A first option would be to try to compute the distance between optimal and approximate solutions on smaller but sufficiently constrained instances. The second option would involve the real scenarios but the approximation algorithms would be compared with optimal solutions for the relaxed subproblems only (such as the assignment or scheduling subproblems) as there are far easier to solve than when they are all integrated.
- We used a very simple model of uncertainty. For now, we assumed that uncertainty about file volumes was uniformly distributed over the interval of uncertainty. Other models could be evaluated as well as the introduction of a forecast model. Such a model would allow ground scheduling to be better informed.
- We could evaluate flexible data download with new metrics, more scenarios with a greater variability of stations and repartition of observations between users. For example, an important issue that has still to be addressed is the design of a flexibility criterion that would assess the flexibility of a ground plan.
- Some algorithms have been proposed to build complete download plans. On the ground, an SWO scheme on top of a non-chronological greedy algorithm is used. Other metaheuristics could be used such as genetic algorithms. In the same spirit, it would be interesting to evaluate the influence of parameters such as the duration of the planning horizon. On this point, it is usually good to have a large planning horizon in deterministic cases. In the case of problem with uncertainties, it may be a good idea to balance the size of the planning horizon with the amount of uncertainties. Indeed, if there are too much uncertainties, the plans become quickly inconsistent at execution.
- One of the temporal phases has not really been investigated yet on board, the execution phase. During this execution, temporal flexibility is provided by the format of the schedule. An onboard reactive rule could be designed to make decisions for observations finishing during the planning horizon. For example, if such an observation has lower volume than maximum, the execution may be waiting for other acquisitions for the next station to be reached. In that case, an onboard reactive decision rule may be able, during execution, to insert new downloads into the schedule to take advantage of this kind of idle period.

Observation scheduling problem

- Concerning the observation scheduling problem, extensive computational experiments should be conducted on other scenarios to demonstrate the benefits of the approach proposed.
- Also, it would be desirable to relax the assumption that all visibility windows are used for downloading. Whereas this is close to reality in over-subscribed scenarios such as defense scenarios where there are few ground stations and then few visibility windows, it is far from true in civil-oriented scenarios where there are a large number of ground stations that cover all continents and are generally underused. This assumption makes the acquisition schedule very pessimistic as the antenna is assumed to be ON most of the time and then consuming a lot of energy. A better option would be to compute on the ground the energy level required for using each group of visibility windows. On board, before any of these groups, if the actual level of energy is sufficient, then the whole group could be used freely for downloading data. Otherwise, only high-priority downloads would be executed to save energy.
- Future research could consist in adapting the exact same conditional mechanism to take the uncertainty about onboard temperature into account. It would allow to relax the high-level constraints used during the ground planning phase that reject 9% of the observations.
- Finally, a significant extension would consist in computing observation profiles on board. This would allow to make the satellite more autonomous and to design an even more flexible decision-making mechanism such as the flexible approach for the data download problem. Since it is not possible to compute these profiles onboard for the time being, it would still be manageable to compute conditional observation plans with a conditional attitude profile (one profile for each branch of the schedule built on the ground).

Experimental platform

An interesting perspective, for both the data download problem and the observation scheduling problem would be, as this is part of early design stage, to fill the gap between theory and practice by performing tests on onboard testbed computers. Implementing the approaches would represent a challenge as it would involve the whole satellite system. Also, specific constraints for onboard algorithms would have to be taken into account with regard to the performance and capabilities of onboard computers.

Generic flexible decision-making

We have seen that *flexible* decision-making may be used in problems involving large horizons and uncertainties on a lot of tasks. When some activities are mandatory such as high-priority observations or downloads, it is still possible to guarantee their realization if it is possible to gather worst-case estimations of their resource consumptions (maximum files volumes, maximum energy consumption, minimum energy productions...). Several frameworks already existed for decision-making under uncertainty but they sometimes lack scalability, such as stochastic programming for example. In this manuscript, we have showed that handling large and complex problems is manageable. One possible direction for future works would be to generalize the framework we have designed for agile Earth-observing satellites to other prob-

lems involving uncertainties. Online stochastic combinatorial optimization proposes the similar idea of unifying two decision-making phases: online fast decision-making with offline methods such as sampling. In our approach, the goal is to focus the computational effort with regard to the decision-making phase and to ease the adaptation of plans computed offline. Finally, we address more the resource constraints and the idea of guaranteeing the realization of mandatory activities.

An experimental environment A

In this appendix, we briefly describe the space mission simulator we have developed for experimenting the flexible approaches against other pure ground or pure onboard approaches.

A.1 Simulator architecture

The simulator has been developed with the JAVA language. Ptolemy has been used to model the system architecture with components representing software or hardware parts of the satellite, and mission elements such as control stations, reception stations, or production centers.

Ptolemy II is an open-source software framework developed at UC Berkeley since 1996 [Eker et al., 2003]. The purpose of this software is to provide a framework for modelling, conception and simulation of dynamical systems. This software is based on software components and heteregenous interaction models already partially or completely implemented between these components. The models may be nested in a hierarchical way to compose a complete heterogenous system. For example, the library allows to model discrete-event subsystems, state machines or signal processing modules. These modules can then be executed on a real-time scale. A Ptolemy model is defined as a set of modules exchanging messages via input/output data ports.

A.1.1 Input data

Baseline observation plan In Fig. 7.3, the building process of the observation plan has been shown. Inputs include a baseline observation plan which has been computed during the *Acquisition scheduling* part of the process. It enforces all constraints but the energy constraints. When applying energy checking or the flexible observation scheduling scheme , some of the observations may be deleted from this baseline plan (as seen in Chap. 7). A list of possible download windows is also available as input too. Indeed, when an observation plan is available, it is possible to deduce the satellite profile and then the possible download windows.

Scenario Inputs include a baseline observation plan which has been computed beforehand by an external planner. A scenario is a set of XML files with :

- a file with the satellite orbit;
- a file with the satellite attitude at all time;
- a file describing ground stations (geographic position, rate, role...);
- a file containing the list of possible visibility windows for each station;

- a file giving the list of users; for each user, the list of authorized control stations and reception stations;
- a file containing the acquisition plan initially computed for the scenario.

Other mission parameters are generic such as the number of memory banks of the central memory, the number of download channels, the TMI antenna angular rotation speed, the characteristics of the key table, the maximum number of files that the memory can contain... These parameters are fixed according to hypothesis given by the space agency.

Planning parameters Finally there are several parameters relative to planning which specify for example :

- the planning approaches used for data download and observation;
- the planning algorithms used on the ground and on board;
- the minimum planning horizon on the ground and on board for data download;
- the minimum duration required x between the end of the onboard planning process and the beginning of its execution. By default, this duration is given by $x = 2 \cdot \delta / \mathbf{Sa}$ seconds with δ the depointing half-angle and $\mathbf{Sa}$ the angular velocity of the antenna; it then allows to schedule downloads on any station because the antenna will be able to cover any angular distance from its current position at the end of the onboard planning process.

A.1.2 Interactions between ground and onboard

Fig. A.3 shows a simplified view of the internal architecture of the simulator for a scenario with 2 ground reception stations, 2 productions centers, and 1 ground control station. There is also a `Satellite` module made of others submodules that we will detail later. The ground segment is made of one ground reception station module for each reception station in the scenario, one production center module for each user, control stations and a single mission center. Interactions between ground and onboard are described in Sect. ??. File downloads are represented as communications between the satellite channels and the ground reception stations. Acknowldegments are modeled by messages sent from ground control stations modules to the `Download` module of the satellite. On the ground segment, production centers receive file downloads from ground reception stations and transmit acknowledgement to control stations. All theses exchanged messages are delayed with specific `delay` generic actors to represent varying distances and link rates between entities.

The behavior of each actor is defined in a Java class. It defines how the actor is processing each message received. It may launch a procedure or send an output message to another module. For example, a production center will send an acknowledgement to control stations (to be then sent on board) once it has received all file downloads for an acquisition. Fig. A.2 shows the Ptolemy networks of connections between ground actors and satellite actors for an example scenario.

A.1.2.1 Decison-making on the ground

On the ground, decision-making for pure ground or flexible scheduling is performed in the `Mission center` module. For data download scheduling, algorithms presented in earlier chapters are implemented on top of a constraint-based model of the data download problem (see Fig. A.3). The library we use for modelling and checking constraints is presented below.

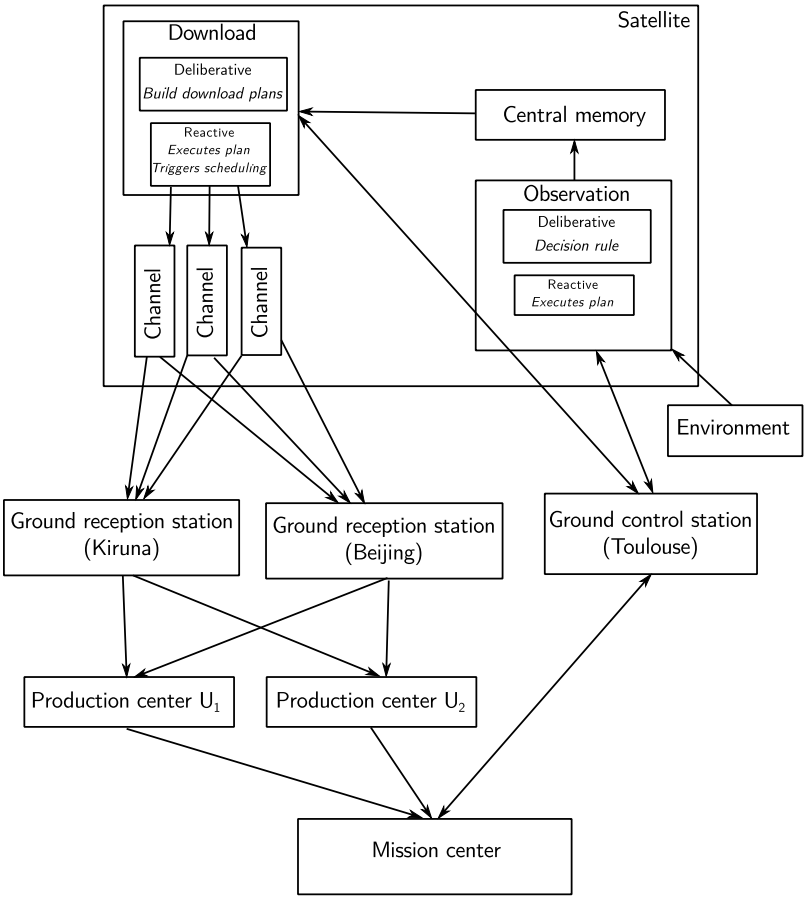


FIGURE A.1: Schematic view of the simulator.

InCELL **I**nvariant-based **C**onstraint **E**va**L**uation **L**ibrary or InCELL is constraint-based local search library developed at ONERA [Pralet and Verfaillie, 2013c] that we use in the ground planning system.

InCELL allows to process *dynamic problems* by using only *static models*, which do not change over the mission horizon. Software embeddedment operational constraints motivates the use of static models. Avoiding dynamic generation of models allows to know beforehand the memory capacity required for planning. It also guarantees, that there will not be any *memory leaks*, any *memory overflow* or wasted time for model generation.

The library is built upon local search techniques. The latter allows to quickly build good quality solutions, using a low amount of memory. Generic techniques of *constraint-based local search*, implemented in tools such as COMET [Michel and Van Hentenryck, 2005] and LocalSolver [Benoist et al., 2011]. The basic element

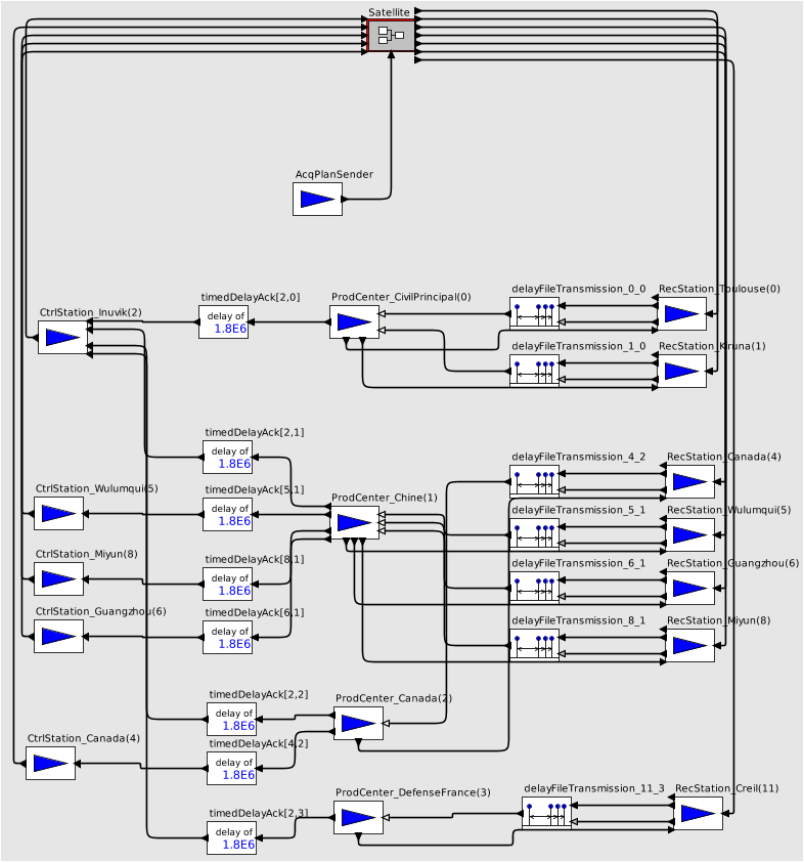


Figure A.2: Ptolemy

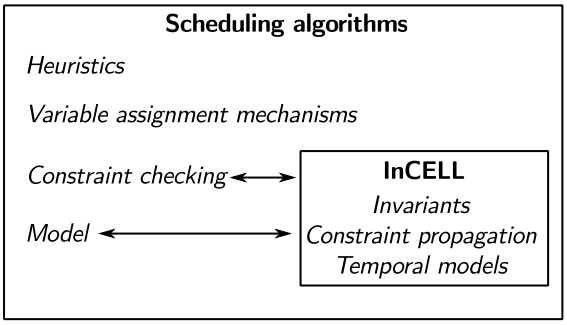


Figure A.3: How scheduling algorithms for the data download problem are built upon a constraint-based model.

handled in these tools is the *invariant*. Invariants allows to model combinatory constraints, temporal constraints, resource constraints, and optimization criterions. They permit to quickly evaluate these values from a model assignement and to incrementally reevaluate them when a change is made to the model. The library also includes techniques to manage Simple Temporal Networks [Dechter et al., 1991] and Time-dependant STN [Pralet and Verfaillie, 2013b].

Invariants are the heart of constraint-based local search. In the COMET tool [Michel and Van Hentenryck, 2005], an invariant is defined as a unidirectionnal constraint $x \leftarrow exp$ with x a variable and exp a functional expression of other variables of the problem, for example $x \leftarrow sum(i \in [1..N])\, y_i$. Invariants are automatically and incrementally maintained through local search movements. Incremental evaluation decreases the amount of work required to recompute output values (left part of the equation) of invariants following local modifications in the assignements of variables contained in the input values (right part of the equation). In the previous example, if y_k changes for $k \in [1..N]$, it is not needed to recompute x from scratch, it simply has to increment x with the difference between the old value of y_k and its new value. The only requirement to allow incremental reevaluation is that invariants must be acyclic, so that a variable is not a function of itself.

In constraint-based local search, the main challenge consists in reevaluating invariants quickly enough to allow many moves to be done. For that, the evaluation process maintains a priority list made of invariants to reevaluate. This list is ordered according the topological position of invariants in the directed acyclic graph of invariants, which allows to reevaluate invariants no more than once. The list of invariants to reevaluate initially contains all successors of modified root variables. When an invariant is reevaluated, all successors are added to the priority list (if they were not in it already). The reevaluation process is stopped as soon as the priority list is empty.

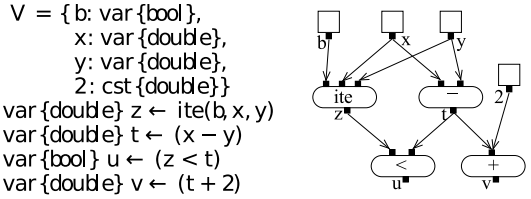


FIGURE A.4: Set of invariants and associated DAG (ite(b; x; y) stands for "if b then x else y"). From [Pralet and Verfaillie, 2013c].

A.1.3 Satellite internal architecture

The `Satellite` module is said to be *composite* because it is made of several low-level modules. Two main modules, `Download` and `Observation`, handle onboard decision-making (when applicable) and action execution . They have a deliberative-reactive behavior [Lemaître and Verfaillie, 2007]. In this kind of behavior, two distinct elements may be distinguished (see Fig. A.5):

– A deliberative submodule, which is called periodically or following an event. For the `Download` module, it is activated before each group of sufficiently close download windows. This submodule builds a temporally flexible plan from a flexible plan as defined in Sect. 6.4.2. It then fills the key change table which

will autonomously change the keys on channels at appropriate times. For the `Observation` module, the deliberative part is minimal because it consists in deciding whether an observation should performed or not.
– A reactive submodule that executes the actions decided by the deliberative part and sends updates to it. The `Download` module executes temporally flexible plan by triggering file downloads on channels and commanding antenna pointings. It also sends updates about new observations to download for example. The `Observation` module executes the observation plan.

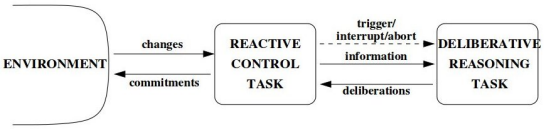


Figure A.5: Deliberative/reactive architecture.

A.2 Execution and outputs

Once the simulation is launched, Ptolemy keeps track of the events (such as observations, downloads, planning phases, download windows...) and moves forward in time until the end of the scenario horizon. Several ground planning phases (~ 4 in a 24-hour scenario), onboard download planning phases (~ 40 in a 24-hour scenario) and observation decision phases (as much as the number of observations in the scenario) are executed (see Fig. A.6). After the simulation has ended, a textual and graphical evaluation is presented to the user. Note that some parameters are required to be uncertain during execution such as file volumes or energy production and consumption. Now, we present how the results are showed.

Textual part

Numeric metrics are computed in the end of the simulation to sum up what happened (see Fig. A.7). These metrics give the number of realized observations, the number of canceled observations due to memory or energy, the number of downloads, the number of observations removed from memory to free space for higher priority observations, the number of observations recorded on board and not downloaded yet, the mean age of observations, and the value of the global criterion such as defined in Sect. 4.2.4.2. Finally, metrics measuring the onboard data download scheduler performance are given such as the mean onboard computing time, the minimum and maximum onboard computing time, and the number of calls to the planner during the simulation.

Graphical part

Executed plans are visulized as activity diagrams with different levels of zoom.

Execution diagram Fig. A.8 shows the execution diagram for a 1-day simulation. The x-axis represents time. The y-axis has several concurrent timelines representing

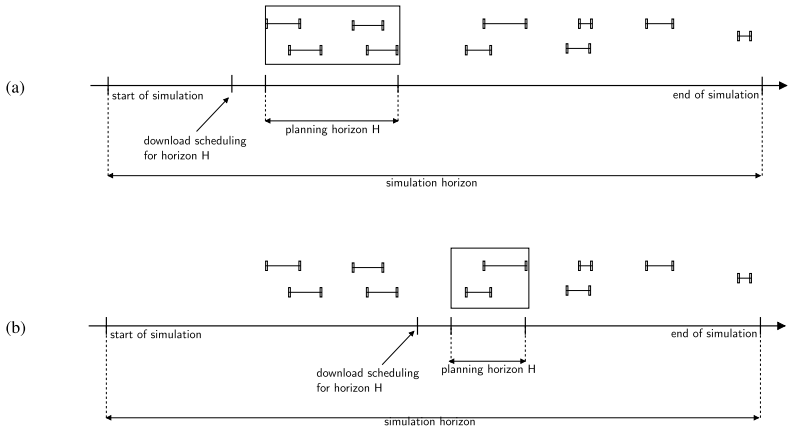


FIGURE A.6: Data download planning horizons during simulation. On the top part, the onboard data download planning horizon for a first set of download windows can be seen. On the bottom part, the same for the next set of download windows.

download channels (`CH_0, ..., CH_4`), instrument switching (`TELESCOPE, ANTENNA`), planning phases (`GRDPLAN, BDPLAN`), download windows (in green with `Vis(Toulouse)` for example), and effective communication windows (in red with `D1(Toulouse)` for example).

Fig. A.9 shows a zoomed version of the previous diagram where we can see more clearly file downloads on channels. Each color is associated with an observation. Observation cancellations due to energy can be seen on the `Telescope` timeline as it is put into SDBY mode (in yellow).

An activity report is also given as a CSV file. It contains details about the status of each observation such as its state (done, downloaded, recorded...), its real volume after random drawing, or other useful data.

Energy diagram Fig. A.10 shows the real onboard energy profile (in black) in percentage of the battery usage. The minimum level of energy to be available onboard is here set to 80%. Concerning the flexible observation scheduling approach, it also shows the simulated energy profile when taking into account worst-case energy consumptions (in green) and the minimum required level of energy computed for ensuring at each instant that the future high-priority observations will be possible (in blue).

Memory diagram Fig. A.11 shows the memory consumption profile for one memory bank. This profile shows the global usage (in black) and the usage per priority level (in red for high-priority observations, in blue for low-priority observations).

```
#done: number of acquisitions done
#canc: number of acquisitions cancelled (not realized due
 to an insufficient memory space available)
#dl: number of acquisitions downloaded
#rem: number of acquisitions removed from the mass memory to free some space
#recnd: number of acquisitions recorded and not downloaded yet at the end of the simulation
meanAge: meanAge in seconds (age = distance between the end
 of acquisition and the delivery time to the production center
util: utility degree (as defined in the PFLEX reports)

*** Priority 0

user              #done    #canc    #dl    #rem    #recnd    meanAge        util
Civil             15       0        9      0       6         6565.684s      0.05000216804076
Defense 1         98       0        92     0       6         7266.85s       0.12202132861121
Defense 2         109      0        93     0       16        6651.067s      0.017968428962817
Defense 3         13       0        12     0       1         5987.557s      0.1693844300566
Defense 4         12       0        7      0       5         10922.015s     0.0020977554552008

GLOBAL            247      0        213    0       34        7016.41s       0.003585114311875


*** Priority 1

user              #done    #canc    #dl    #rem    #recnd    meanAge        util
Civil             950      14       291    0       659       30622.793s     0.003142601016797
Defense 1         11       0        2      0       9         14468.036s     7.678946105072E-4
Defense 2         10       0        5      0       5         15006.02s      0.03398805082631
Defense 3         28       0        5      0       23        14966.403s     0.02197952493272
Defense 4         33       0        10     0       23        23068.495s     0.007416737908159

GLOBAL            1032     14       313    0       719       29778.645s     0.0013062576790861

Total telescope ON or STDBY time : 4766s
Total antenna ON or STDBY time : 8052s
Observations cancelled due to energy 71

=== PLANNING INFOS:
Number of plans built: 43
Total planning time (sec.): 0.082 sec.
Mean planning time (sec.): 0.0019069767441860467 sec.
Max planning time (sec.): 0.014 sec.
```

Figure A.7: Numerical metrics.

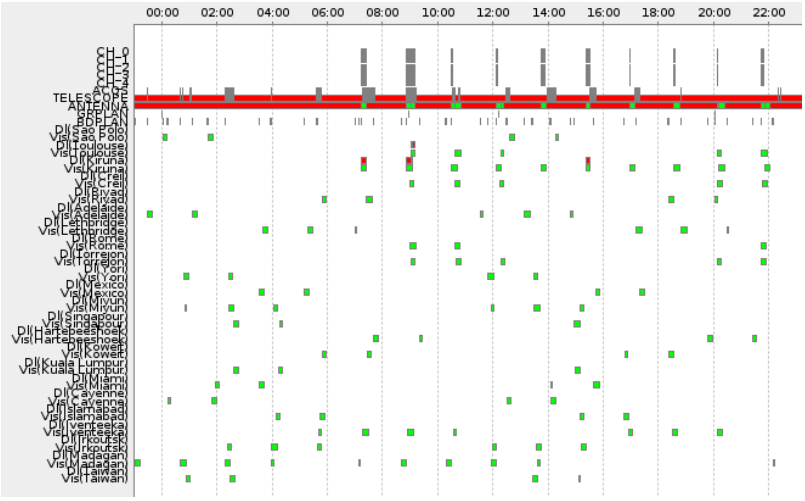


FIGURE A.8: Execution diagram (no zoom) during a 1-day simulation.

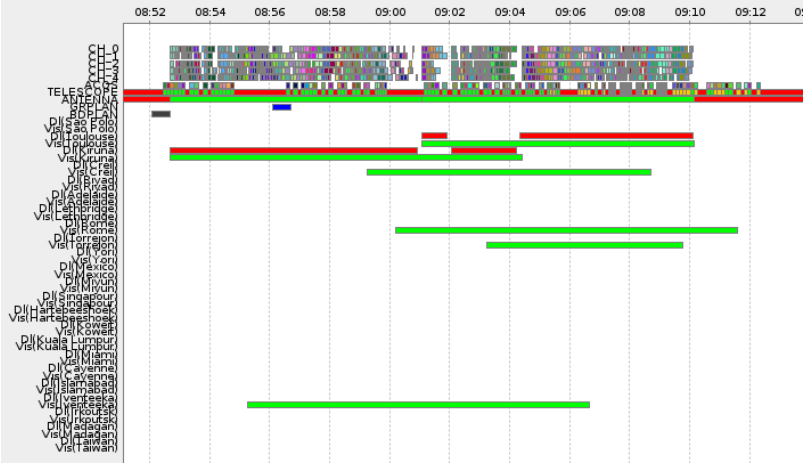


FIGURE A.9: Execution diagram (zoom on a group of activities) during a 1-day simulation.

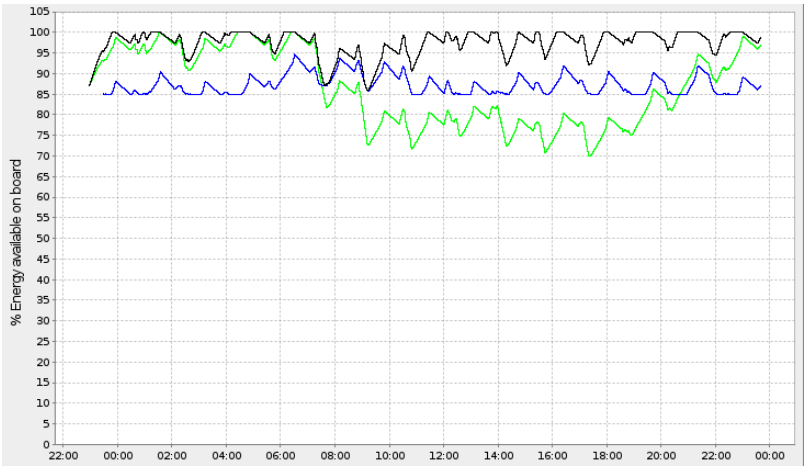


Figure A.10: Energy consumption during a 1-day simulation.

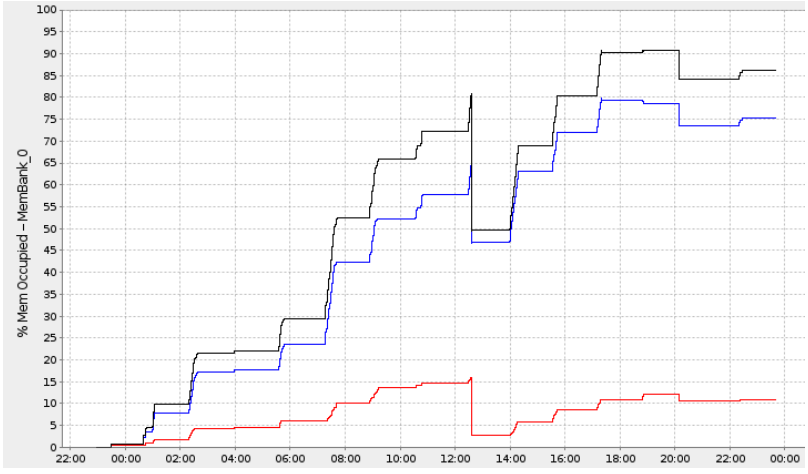


Figure A.11: Memory consumpton during a 1-day simulation.

Guarantees for the R1 insertion rule B

In this appendix, we provide two guarantees concerning the R1 insertion rule which can be seen in Algorithm 3 in Sect. 5.2.1. This procedure is here decomposed into two other procedures : the EnqueueBatch algorithm which is a generalization of the R1 insertion algorithm as no heuristic is used for choosing the next file to schedule and the Enqueue procedure which schedules one file. First we prove that enqueuing a file with algorithm Enqueue returns a schedule without resource conflict. Second, we prove that by using iteratively Enqueue to schedule all files of an observation, EnqueueBatch (and then the R1 insertion algorithm) produce schedules that are somewhat *balanced* on every channels. This work is extracted from a journal paper submitted to Discrete Applied Mathematics [Hebrard et al., 2015].

In this appendix, we use vocabulary from the scheduling literature. Jobs represent files and a batch is a set of files linked to an acquisition, resources correspond to memory banks and parallel machines to telecommunication channels.

Also, we use the following notations and conventions:

n:	the number of jobs, denoted $a_1, \ldots, a_n$
$res(a)$:	the resource required by job a
p_a:	the size of job a, whereas p_{min} and p_{max} stand for the minimum and maximum job sizes, respectively
s_a and e_a:	the start and end time of job a in the schedule, respectively
m:	the number of machines, denoted $M_1, \ldots, M_m$
e_M:	the end time of the last job scheduled on machine M
e_{min} and e_{max}:	the completion time of the last job on the machine finishing first and last, respectively. That is, $e_{min} = \min(\{e_{M_j} \mid 1 \leq j \leq m\})$ and $e_{max} = \max(\{e_{M_j} \mid 1 \leq j \leq m\})$
λ:	the number of resources, denoted $R_1, \ldots, R_\lambda$; Moreover, we also use the symbol R to denote the set of jobs requiring this resource and $L(R)$ stands for the load of this resource, that is, the total processing time of its jobs: $L(R) = \sum_{a \in R} p_a$

B.1 Schedule Density

We consider a greedy heuristic which iteratively inserts a job a into a partial schedule $\mathcal{S}$. We can define an heuristic as an ordering on the jobs and an insertion procedure called Enqueue and shown in Algorithm 9.

The algorithm Enqueue considers a job a and a partial schedule $\mathcal{S}$. We shall see that there is at most one machine whose last job requires the same resource as a. The job a is inserted at the back of such machine if it exists, and at the back of the first available machine otherwise.

Algorithm 9: Enqueue(schedule : $\mathcal{S}$, job : a)

```
if there exists a machine M_j whose last job in S requires res(a) then
    insert a at the back of machine M_j in S;
else
    insert a at the back of the earliest finishing machine in S;
```

We show that any schedule obtained by calling n times the insertion procedure Enqueue on a set of n jobs is valid and dense, that is, there is no idle time between the first and last job on any machine.

Theorem 1. *The schedule computed by n calls to* Enqueue *is valid and dense, and for any machine M, every job a processed on M and such that $e_a \geq e_{min}$ requires the same resource.*

Proof: We prove this proposition by induction on the number of jobs n. For 0 job, it trivially holds. Now we suppose that it holds for n jobs, and show that it also holds for $n+1$ jobs.

Let R_M be the unique resource such that every job scheduled on machine M and finishing later than e_{min} requires R_M (or $\emptyset$ if $e_M = e_{min}$). Observe that since the schedule is valid, there is no two machines M_1, M_2 such that $R_{M_1} = R_{M_2}$ unless $e_{M_1} = e_{M_2} = e_{min}$ (hence for any job a, there is at most one machine M_j such that $R_{M_j} = res(a)$). When inserting the $n+1$-th job a, there are two cases (illustrated in Figure B.1):

Case 1: $\nexists j \in [1, \ldots, m]$ *s.t.,* $res(a) = R_{M_j}$;
Job a is enqueued on machine M whose last job finishes first. It can start immediately after this last, i.e., at time e_{min}, since there is no resource conflict with $res(a)$ on the interval $[e_{min}, e_{max}]$. Now, e_{min} and maybe e_{max} can increase because of this insertion. In all machines but M, this does not invalidate the induction hypothesis. On M, there is now at most a single job on the interval $[e_{min}, e_{max}]$, hence the induction hypothesis is also verified.

Case 2: $\exists j \in [1, \ldots, m]$ *s.t.,* $res(a) = R_{M_j}$;
Job a is enqueued on the machine M_1 such that $res(a) = R_{M_1}$. It can start immediately after the last job of this machine, since $e_{M_1} \geq e_{min}$, and there is no other machine M_2 such that $e_{M_1} \leq e_{M_2}$ and $R_{M_1} = R_{M_2}$. Observe that in this case R_{M_j} remains unchanged for every machine M_j, hence the induction hypothesis is also verified.

□

B.2 Batch Insertion

In the data download problem, jobs are grouped by *batches* of λ jobs, each one requiring a different resource. Indeed, it is preferred to group the downloading of all the files of an acquisition for at least three reasons. First, if for some reason the download is interrupted, any acquisition for which a single file is not yet downloaded is lost. By grouping all the files of an acquisition, we reduce the probability of these occurrences. Second, some acquisitions have higher priority and we might want to

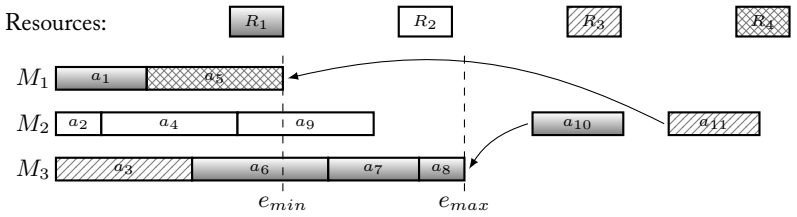


FIGURE B.1: Illustration of the proof of Lemma 1. Job a_{10} falls into the second insertion case whilst job a_{11} falls into the first case.

download them first if possible. Third, all files of a single acquisition must be downloaded in the same time window. When inserting a single new batch in a given time window of the partial schedule (during a local search method or an online scheduling method), it might be costly to re-schedule all the files for this window. If instead, we insist that all the files of the last batch are inserted at the back, the cost is much lower.

It is therefore important to have an efficient method, albeit following a predefined batch order.

The greedy heuristic EnqueueBatch, given in Algorithm 10, follows the given batch order, and uses the Enqueue in any order within a batch.

Algorithm 10: EnqueueBatch(`set of batches` : $\mathcal{B}$)

```
while B ≠ ∅ do
    remove β from B;
    while β ≠ ∅ do
        Pick and remove a job a from β;
        Enqueue(S, a);
return S;
```

We shall assume that we have $n = \alpha\lambda$ jobs each requiring one of λ resources, and each resource supplies for exactly α jobs. In other words, we have α batches of λ jobs. We now want to prove thatthe batch insertion algorithm (the R1 insertion rule) produces a somewhat *balanced* scheduled and that, after inserting a batch, the gap between the least busy channel and the most busy channel is bounded.

Theorem 2. *If there are α batches of λ jobs, the schedule returned by* EnqueueBatch *is such that $e_{max} - e_{min} \leq p_{max}$.*

Proof: We prove this proposition by induction on α: If there is no job at all, this property trivially holds.

Now we suppose that it is true for α batches and we show that it is also true for $\alpha+1$. Let e_j^α be the end time e_{M_j} of machine M_j after inserting the α^{th} batch. Without loss of generality, we suppose $e_1^{\alpha+1} \leq \ldots \leq e_m^{\alpha+1}$, and we assume that the induction hypothesis is falsified at step $\alpha+1$, i.e., $e_m^{\alpha+1} - e_1^{\alpha+1} > p_{max}$. Since $e_m^{\alpha+1} - e_1^{\alpha+1} > p_{max}$ and since no job is larger than p_{max}, it means that the penultimate

job of machine M_m finishes later than the last job of machine M_1. Therefore, the penultimate and last jobs of machine M_m must require the same resource, hence belong to a different batch.

It follows that only one job of batch $\alpha+1$ is on machine M_m. Hence $e_m^{\alpha+1} - p_{max} \leq e_m^{\alpha}$. Therefore, since by induction hypothesis $e_{\max}^{\alpha} - e_{\min}^{\alpha} \leq p_{max}$, the following inequalities hold:

$$\forall 1 \leq i < m,\ e_i^{\alpha} \geq e_m^{\alpha+1} - 2p_{max} \tag{B.1}$$

Moreover, since a single job of batch $\alpha+1$ is allocated to machine M_m, the $\lambda-1$ other jobs must be distributed amongst machines M_1 to M_{m-1}.

If $\lfloor \frac{\lambda-1}{m-1} \rfloor$ jobs or more have been allocated to machine M_1, we have $e_1^{\alpha+1} \geq e_1^{\alpha} + \lfloor \frac{\lambda-1}{m-1} \rfloor p_{min}$. By applying inequality B.1, we have $e_1^{\alpha+1} \geq e_m^{\alpha+1} - 2p_{max} + \lfloor \frac{\lambda-1}{m-1} \rfloor p_{min}$, hence $e_1^{\alpha+1} \geq e_m^{\alpha+1} - p_{max}$. Therefore, at most $\lfloor \frac{\lambda-1}{m-1} \rfloor - 1$ jobs have been allocated to machine M_1. In other words, among the λ jobs of the batch, one has been assigned to machine m and at most $\lfloor \frac{\lambda-1}{m-1} \rfloor - 1$ to machine M_1, so there are $\lambda - \lfloor \frac{\lambda-1}{m-1} \rfloor$ jobs to distribute over the $m-2$ other machines. It follows that at least one machine will be allocated strictly more than $\lfloor \frac{\lambda-1}{m-1} \rfloor$ jobs, because the following relation is a tautology:

$$\lambda - \lfloor \tfrac{\lambda-1}{m-1} \rfloor > \lfloor \tfrac{\lambda-1}{m-1} \rfloor (m-2) \tag{B.2}$$

$$\Leftrightarrow \quad \lambda > \lfloor \tfrac{\lambda-1}{m-1} \rfloor (m-1) \tag{B.3}$$

$$\Leftarrow \quad \lambda > \lambda - 1 \tag{B.4}$$

Now, let M_j be the machine that was allocated at least $\lfloor \frac{\lambda-1}{m-1} \rfloor + 1$ jobs ($1 < j < m$). Consider the schedule when adding the $(\lfloor \frac{\lambda-1}{m-1} \rfloor + 1)^{\text{th}}$ job of batch $\alpha+1$ on machine M_j. Since this is not the first job of the batch to be allocated to machine M_j, it cannot require the same resource as the previous one. Therefore machine M_1 must finish later than the start of this $(\lfloor \frac{\lambda-1}{m-1} \rfloor + 1)^{\text{th}}$ job on machine M_j. In other words, we have $e_1^{\alpha+1} \geq e_j^{\alpha} + \lfloor \frac{\lambda-1}{m-1} \rfloor p_{min}$. By applying inequality B.1 as above, we have $e_1^{\alpha+1} \geq e_m^{\alpha+1} - p_{max}$. In both cases, the assumption that the induction hypothesis does not lift to $\alpha+1$ is contradicted. □

OPL model of the data download problem C

```
using CP;

/*number of channels*/
int Nc = ...;
assert Nc > 0 ;
/*channels*/
range Ncs = 1..Nc ;

/*number of windows*/
int Nw = ... ;
assert Nw > 0 ;
/*windows*/
range Nws = 1..Nw ;

/*Acquisitions*/
int Na = ... ;
assert Na > 0 ;
range Nas = 1..Na ;

/*Number of files*/
int Nf = ...;
/*Files*/
range Nfs = 1..Nf;

/*number of ground stations*/
int Nst = ... ;
assert Nst > 0 ;
/*Stations*/
range Sts = 1..Nst ;
range Stz = 0..Nst ;

/*Number of memory blocks*/
int Nmb = ...;
assert Nmb > 0;
range Nmbs = 1..Nmb;

/*number of entities*/
int Ne = ... ;
assert Ne > 0 ;
/*Entities*/
range Es = 1..Ne ;

/*number of priority levels*/
int Np = ... ;
assert Np > 0 ;
/*Priorités*/
```

```
47 range Ps = 1..Np ;
48
49 /*current priority*/
50 int Pc = ...;
51 assert Pc > 0;
52 assert Pc <=Np;
53 /*previous objectives to be met*/
54 float Op[1..Np] = ...;
55 assert forall(n in 1..Np : n >= Pc) Op[n] == 0;
56
57  /*freshness half period */
58  float DemPer = ... ;
59  assert DemPer > 0 ;
60  int DemPerInt = ftoi(ceil(1000 * DemPer)) ;
61
62 /*sharing parameter to compute the criterion*/
63 float Par = ... ;
64 assert Par <= 1 ;
65
66 /*max number of key-switching*/
67 int Nsw = ...;
68 assert Nsw > 0;
69
70 int NProcC = ...;
71 assert NProcC > 0;
72 range NProcCs = 1..NProcC;
73
74 /*TMI pointing delay*/
75 float Tstsw = ...;
76 int TstswInt = ftoi(ceil(1000*Tstsw));
77
78 /*Download rate*/
79 float Dr = ... ;
80 assert Dr > 0 ;
81
82 /*Informations associated to each visibility window*/
83
84 /*Ground stations*/
85 int Stc[Nws] = ... ;
86 assert forall(w in Nws) 1 <= Stc[w];
87 assert forall(w in Nws) Stc[w]<= Nst ;
88
89 /*start time*/
90 float Wstart[Nws] = ... ;
91 assert forall(c in Nws) Wstart[c] >= 0 ;
92 int WstartInt[w in Nws] = ftoi(ceil(1000 * Wstart[w])) ;
93
94 /*end time*/
95 float Wend[Nws] = ... ;
96 assert forall(w in Nws) Wstart[w] < Wend[w] ;
97 int WendInt[w in Nws] = ftoi(floor(1000 * Wend[w])) ;
```

```
98
99  /*compute the overall horizon*/
100 float WstartMin = min(w in Nws) Wstart[w] ;
101 float WendMax = max(w in Nws) Wend[w] ;
102 int WstartMinInt = min(w in Nws) WstartInt[w] ;
103 int WendMaxInt = max(w in Nws) WendInt[w] ;
104
105 /*Informations associated to each acquisition*/
106 /*start time*/
107 float Astart[Nas] = ... ;
108 assert forall(a in Nas) Astart[a] >= 0 ;
109 /*end time*/
110 float Astop[Nas] = ... ;
111 int AstopInt[a in Nas] = ftoi(ceil(1000 * Astop[a])) ;
112
113 float Adeadline[Nas] = ...;
114 assert forall(a in Nas) Adeadline[a] >= 0 ;
115 int AdeadlineInt[a in Nas] = ftoi(ceil(1000 * Adeadline[a])) ;
116
117 /*priority*/
118 int Pa[Nas] = ... ;
119 assert forall(a in Nas) 1 <= Pa[a] ;
120 assert forall(a in Nas)Pa[a] <= Np ;
121
122 /*normalised weight*/
123 float Wn[Nas] = ... ;
124 assert forall(a in Nas) Wn[a] > 0 ;
125 float Wnmin = min(a in Nas) Wn[a] ;
126 float Umin = 0.0000001;
127
128 /*associated entity*/
129 int Ea[Nas] = ... ;
130 assert forall(a in Nas) Ea[a] <= Ne ;
131 assert forall(a in Nas) 1 <= Ea[a]  ;
132
133 /*Informations for each file*/
134
135 /*memory block*/
136 int Mb[Nfs] = ...;
137 assert forall(f in Nfs) Mb[f] >= 1 ;
138 assert forall(f in Nfs) Mb[f] <= Nmb ;
139
140 /*corresponding acquisition*/
141 int Acqf[Nfs] = ...;
142
143 /*encrypting key*/
144 int Kf[Nfs] = ...;
145
146 /*generated volume*/
147 float V[Nfs] = ... ;
148 assert forall(f in Nfs) V[f] >= 0 ;
```

```
149
150 /*downloading duration calculated from generated volume and debit*/
151 float Dv[f in Nfs] = V[f]/Dr ;
152 assert forall(f in Nfs) Dv[f] >= 0 ;
153 int DvInt[f in Nfs] = ftoi(ceil(1000 * Dv[f])) ;
154
155 /*Informations associated to each entity*/
156
157 /*Quota*/
158 float Q[Es] = ... ;
159 assert forall(e in Es) 0 <= Q[e] ;
160 assert forall(e in Es) Q[e] <= 1 ;
161 assert (sum(e in Es) Q[e]) == 1;
162
163 /*main ground station for this entity*/
164 int Sp[Es] = ... ;
165 assert forall(e in Es) 1 <= Sp[e] ;
166 assert forall(e in Es) Sp[e] <= Nst ;
167
168 /*secondary ground stations*/
169 int Ss[Es][Sts] = ... ;
170 assert forall(e in Es, st in Sts) Ss[e][st] <= 1 ;
171 assert forall(e in Es, st in Sts) 0 <= Ss[e][st] ;
172 assert forall(e in Es) Ss[e][Sp[e]] == 1 ;
173 //assert forall(e in Es) Ss[e][0] == 0 ;
174
175 /*main processing center*/
176 int Pce[Es] = ...;
177 //assert forall(e in Es) Pce[e] > 0;
178 //assert forall(e in Es) Pce[e] <= NProcC;
179
180
181 /*Informations associated to each pair of stations*/
182 /*ground transfer delay*/
183 float Dt[Stz][NProcCs] = ... ;
184 assert forall(st in Stz, pc in NProcCs) Dt[st][pc] >= 0 ;
185
186 int DtInt[st1 in Stz][pc in NProcCs] = ftoi(ceil(1000 * Dt[st1][pc
      ])) ;
187
188 /*TMI antenna transition matrix*/
189 tuple triplet {int id1; int id2; int value;};
190 {triplet} M = {<i,j,TstswInt> | i in Sts, j in Sts};
191
192 // interval of download of each file
193 dvar interval itvs[i in Nfs] optional
194     in WstartMinInt..WendMaxInt
195     size DvInt[i];
196
197 // download alternatives (over each canal)
198 dvar interval altItvChannels[i in Nfs][c in Ncs] optional
```

```
199     in  WstartMinInt..WendMaxInt
200     size DvInt[i];
201
202 // download alternatives (over each window)
203 dvar interval altItvWindows[i in Nfs][w in Nws] optional
204     in WstartInt[w]..WendInt[w]
205     size DvInt[i];
206
207 // sequence of downloads for each canal
208 dvar sequence dlseq[c in Ncs] in
209     all(i in Nfs) altItvChannels[i][c] types
210     all(i in Nfs) Kf[i];
211
212 /*span of all downloads for each window */
213 dvar interval spanOverWindows[w in Nws] optional
214     in  WstartInt[w]..WendInt[w]
215     size 0..WendInt[w] - WstartInt[w];
216
217 //sequence of windows
218 dvar sequence dlseqw in
219     all(w in Nws) spanOverWindows[w] types
220     all(w in Nws) Stc[w];
221
222 // key associated with the download preceding the download of a
      file on the same canal
223 dexpr int keyPrev[i in Nfs] = max(c in Ncs) typeOfPrev(dlseq[c],
      altItvChannels[i][c],0);
224
225 dexpr int nChangeKeys = sum(f in Nfs) (keyPrev[f] != 0 && keyPrev[f
      ] != Kf[f]);
226
227 dexpr int acqdl[i in Nas] =  max(f in Nfs: Acqf[f] == i) presenceOf
      (itvs[f]) ;
228
229 //station where a is downloaded
230 dexpr int dlStation[a in Nas] = max(w in Nws, f in Nfs: Acqf[f] ==
      a) presenceOf(altItvWindows[f][w]) * Stc[w];
231
232 //time when the last file of a is downloaded
233 dexpr int endOfLastFiledl[a in Nas] = max(f in Nfs: Acqf[f] == a)
      endOf(itvs[f]);
234
235 //delay between the end of acquisition a and its download date
236 dexpr float dia[a in Nas] = (endOfLastFiledl[a] + DtInt[dlStation[a
      ]][Pce[Ea[a]]] - AstopInt[a]) ;
237
238 //utility computing
239 dexpr float fresh[a in Nas] = (acqdl[a] == 1)*(2^(-dia[a]/DemPerInt
      ));
240
241 dexpr float wncMin = (min(a in Nas)(Wn[a] *(2^(-(AdeadlineInt[a] -
```

```
        AstopInt[a])/DemPerInt)))) - Umin;
242
243 dexpr int nbAcqPE[p in Ps][e in Es] = sum(a in Nas)((Pa[a] == p) &&
        (Ea[a] == e));
244
245 dexpr float oep[p in Ps][e in Es] =
246     maxl(wncMin, (sum(a in Nas: (Pa[a] == p) && (Ea[a] == e))
247                     (Wn[a] * fresh[a]))) ;
248
249 dexpr float oepMax[p in Ps][e in Es] =
250     (nbAcqPE[p][e] == 0) + ((nbAcqPE[p][e] != 0)*
251     maxl(wncMin, (sum(a in Nas: (Pa[a] == p) && (Ea[a] == e)) Wn[a
            ])));
252
253 /*utility by entity and priority level*/
254 /*Umin is the minimum*/
255 dexpr float oepm[p in Ps][e in Es] =
256 (nbAcqPE[p][e] == 0) + ((nbAcqPE[p][e] != 0) * oep[p][e]) ;
257
258 ///*utility by priority level*/
259 dexpr float op[p in Ps] =
260     ((p < Np) *
261     (((sum(e in Es) (oepm[p][e]^(Par))) / Ne)^(1/Par))) +
262     ((p == Np) *
263     (((sum(e in Es) (Q[e] * ((oepm[p][e] / Q[e])^Par)))^(1/Par))/((
            sum(e in Es) Q[e]*(oepMax[p][e]/Q[e])^Par)^(1/Par))));
264
265 /*downloaded acquisitions by priority and entity*/
266
267 dexpr int nep[p in Ps][e in Es] =
268    sum(a in Nas: (Pa[a] == p) && (Ea[a] == e)) (acqdl[a]==1) ;
269
270  /*********************/
271  /*Search parameters*/
272  /*********************/
273
274  execute {
275     cp.param.relativeoptimalitytolerance = 1.0E-8;
276  } ;
277
278 //Maximize the current priority
279 maximize op[Pc];
280
281 constraints {
282     //two files from the same memory block cannot be downloaded in
            the same time
283     forall(mb in Nmbs){
284         noOverlap(all(f in Nfs: Mb[f] == mb) itvs[f]);
285     }
286
287     //compute actual downloading windows
```

```
288     forall(w in Nws){
289         span(spanOverWindows[w], all(f in Nfs) altItvWindows[f][w])
                ;
290     }
291
292     //no overlap between windows
293     noOverlap(dlseqw, M);
294
295     // no overlap between downloads over the same canal
296     forall(c in Ncs)
297         noOverlap(dlseq[c]);
298
299     nChangeKeys <= Nsw;
300
301     forall(i in Nfs){
302
303         // each file downloaded using at most one of the
                alternative channels
304         alternative(itvs[i], all(c in Ncs) altItvChannels[i][c]);
305
306         // each file downloaded using at most one of the
                alternative windows
307         alternative(itvs[i], all(w in Nws) altItvWindows[i][w]);
308
309         //all files linked to an image should be downloaded in a
                single window
310         forall(j in Nfs : Acqf[i] == Acqf[j] && i< j){
311             forall(w in Nws){
312                 presenceOf(altItvWindows[i][w]) == presenceOf(
                        altItvWindows[j][w]);
313             }
314         }
315
316         /*a file should be downloaded only after the end of the
                acquisition it is part of*/
317         startOf(itvs[i],AstopInt[Acqf[i]]) >= AstopInt[Acqf[i]];
318
319         /*an acquisition should be downloaded on one of the entity
                s station*/
320         forall(w in Nws:(Ss[Ea[Acqf[i]]][Stc[w]] == 0)){
321             presenceOf(altItvWindows[i][w]) == 0   ;
322         }
323
324     }
325
326      /*objectives for higher priorities should be met at least*/
327      forall(p in Ps: p < Pc){
328            op[p] >=  Op[p] ;
329      }
330
331      /*We do not consider files which priority is strictly below Pc
```

```
332         */
333     forall(f in Nfs: Pa[Acqf[f]] > Pc){
334             presenceOf(itvs[f]) == 0 ;
335     }
336
337 }
```

Quatrième partie

Résumé étendu

L'observation de la Terre depuis l'espace est utile dans de nombreux domaines comme la météorologie, la géodésie, la modélisation du climat, la gestion des catastrophes naturelles, ou la reconnaissance militaire. Le système spatial que nous considérons est composé de satellites orbitant autour de la Terre. Ces derniers sont équipés d'instruments optiques à haute résolution, qui permettent d'acquérir des images pour des utilisateurs civils ou militaires, et communiquent avec un réseau de stations sol. Les données acquises sont ensuite compressées, enregistrées, et télé-déchargées (ou vidées) au sol. Les utilisateurs du systèmes soumettent des requêtes d'observation au centre de mission qui produit des plans d'activités qui sont envoyés aux satellites. Ces plans contiennent plusieurs types d'actions tels que les manœuvres orbitales, les acquisitions ou les vidages.

Un aspect à ne pas négliger lors de la planification d'activités pour des satellites d'observation est que ceux-ci évoluent dans un environnement dynamique où des événements imprévus arrivent, comme des changements météorologiques ou des requêtes d'observation urgentes. Plusieurs paramètres sont incertains lorsque la planification des activités est réalisée au sol, comme la couverture nuageuse ou la quantité d'énergie disponible à bord. Jusqu'à aujourd'hui, les plans d'activités ne doivent pas être modifiés à bord. Ils doivent donc être robustes aux incertitudes pour ne pas mettre le satellite dans un état non désiré. Cela implique d'une part, que toutes les décisions sont prises hors-ligne au sol et que le satellite ne fait qu'exécuter strictement le plan sans le modifier d'aucune façon, et d'autre part, que des hypothèses de pire scénario concernant les paramètres incertains sont considérées. Cela rend la planification pessimiste et les plans en résultant sont sous-optimaux. Dans ce contexte, ces incertitudes rendent la planification et l'ordonnancement hors-ligne des activités de plus en plus discutables. L'objectif de ce travail est d'améliorer les performances en donnant plus d'autonomie au satellite pour la prise de décision sans compromettre la prévisibilité requise pour certaines activités. L'idée principale est alors de partager le processus de décision entre le sol et le bord de manière à profiter des capacités de calcul au sol et de la faible incertitude restant à bord. D'abord, on applique cette idée au vidage des acquisitions qui consiste à ordonnancer des télé-déchargements de fichiers durant des fenêtres de visibilité satellite/station sol. Puis, on applique cette idée à la planification des observations.

Contexte industriel : le programme OTOS

En 2011 et 2012, deux satellites agile d'observation de la Terre, Pléiades-1A et Pléiades-1B, ont été mis en orbite par des fusées Soyouz lancées du Centre Spatial Guyanais. Ce programme de coopération entre la France et l'Italie a été géré par le Centre National d'Etudes Spatiales (CNES). Plusieurs pays européens ont contribué à ce programme comme la Suède (calculateurs de bord) ou l'Espagne (antenne bande S).

OTOS est un programme technologique dont le but est d'améliorer les performances de ces systèmes d'observations tout en diminuant leur coût. Les paramètres à améliorer incluent la résolution d'image (la surface réelle couverte par un pixel, la résolution actuelle allant jusqu'à $20cm^2$ par pixel), la précision altimétrique pour mesurer des reliefs, et le nombre de bandes spectrales (le nombre de fréquences auxquelles

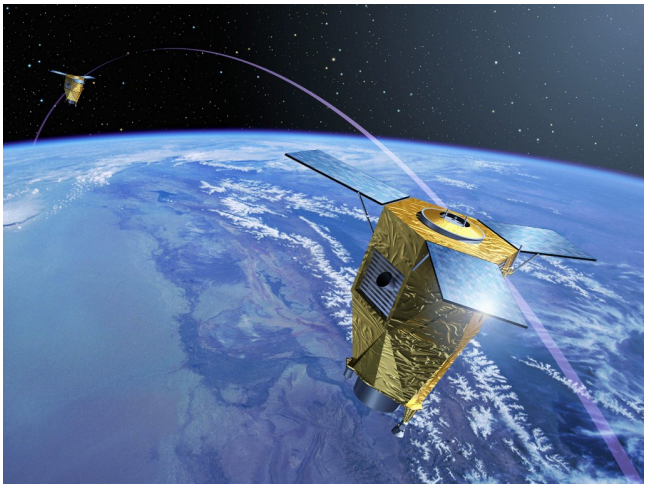


Figure 9.1 – Vue d'artiste d'un satellite Pléiades en orbite autour de la Terre (CNES, 2003).

l'instrument est capable de collecter des données). D'un point de vue opérationnel, il serait souhaitable d'améliorer la *réactivité* (le délai entre la requête d'observation et la livraison de l'image), paramètre important dans des situations comme les feux de forêt ou des inondations, et la *fréquence de revisite* au-dessus d'une même zone géographique, paramètre important pour la surveillance de zone.

Dans ce chapitre, on décrit les éléments d'une mission typique d'observation de la Terre, les différents acteurs impliqués, et le processus de décision permettant la planification des activités.

Le système spatial que l'on considère est constitué d'un satellite agile d'observation de la Terre (voir Section 10.2.2), de plusieurs stations sol de contrôle, de plusieurs stations sol de réception, et de plusieurs centres de traitement des données, chacun associés avec un utilisateur du système. Dans les paragraphes suivants, on décrit les différents éléments du système (voir Figure 10.1).

10.1 Architecture du système

Il y a un segment sol, que l'on décrit ci-dessous, et un satellite d'observation de la Terre, que l'on décrit en Section 10.2.

10.1.1 Centre de mission

Le centre de mission reçoit les requêtes d'observations de la part des utilisateurs du système et produit des plans d'acquisition et de vidage. Ces plans sont ensuite transmis à des stations de contrôle. Le processus de planification est complexe et de nombreux échanges, non détaillés ici, entre les entités peuvent être nécessaire pour valider les plans d'activités.

10.1.2 Stations de contrôle

Les stations de contrôle reçoivent les plans d'activité produits par le centre de mission et les téléchargent à bord du satellite. Elles reçoivent aussi ce qu'on appelle la *télémétrie de servitude*. Cette dernière est un ensemble d'informations envoyé depuis le satellite qui permet de vérifier que tous les systèmes bord fonctionnent correctement. Les stations de contrôle reçoivent aussi la liste des observations effectuées, la liste des observations supprimées de la mémoire, et la liste des observations vidées. Les stations de contrôle sont réparties sur la surface du globe. Par exemple, le CNES

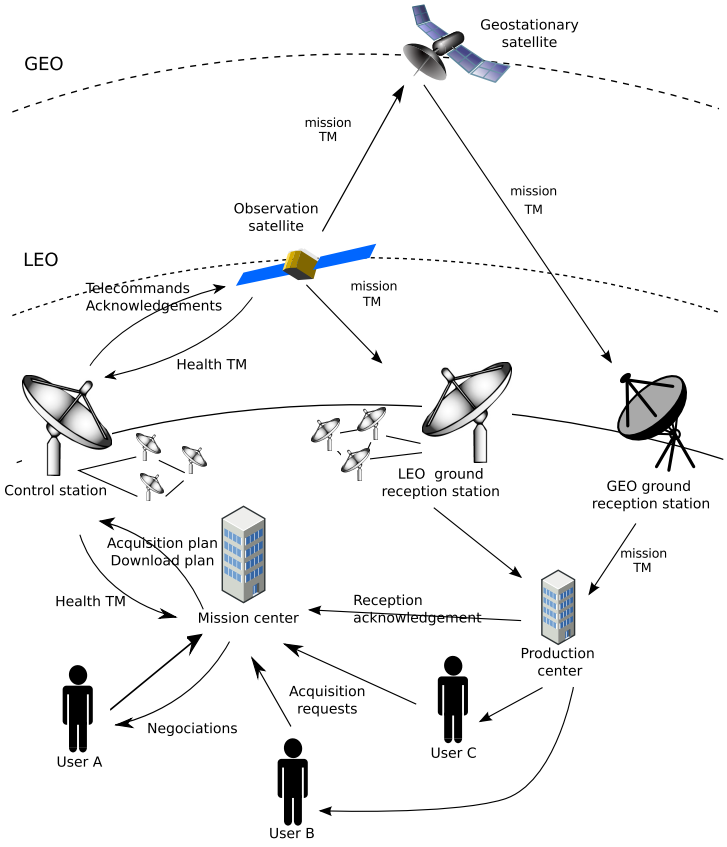


Figure 10.1 – Architecture du système.

a accès au réseau suivant de stations de contrôle : Kerguelen dans les îles Kerguelen ; Kiruna en Suède ; Aussaguel en France ; Kourou en Guyane Française ; Hartebeesthoek en Afrique du Sud ; Perth en Australie. Toutes ces stations sont connectées avec le centre de mission par un lien haut-débit.

10.1.3 Stations de réception

Les stations de réception sont exclusivement utilisées pour recevoir de la *télémétrie mission* qui correspond aux données collectées par les instruments de la charge utile du satellite (des images dans notre cas). Ces données peuvent être transférées quand le satellite est en visibilité, c'est à dire quand son antenne de télémétrie est capable de communiquer avec l'antenne de la station. Chaque utilisateur a une ou plusieurs stations autorisées pour le vidage des données. Les acquisitions reçues dans une station de réception sont ensuite envoyées dans le centre de production associé à l'utilisateur demandeur.

La liste des acquisitions vidées et la liste des acquisitions stockées en mémoire à bord du satellite sont envoyées au centre de mission. Notons qu'il est important de maintenir la cohérence entre l'état de la connaissance entre les stations sol et le satellite de façon à ce que les algorithmes sol puissent s'appuyer sur les données les plus récentes pour produire des plans. Pourtant, ce n'est pas toujours possible, à cause des délais de transfert au sol par exemple. Il est nécessaire de penser à la réparation du plan lors de l'exécution quand le plan produit au sol n'est plus cohérent avec l'état réel du satellite (comme par exemple un plan de vidage qui contient le vidage d'une acquisition qui a déjà été effacée à bord).

10.1.4 Utilisateurs, requêtes d'observation et centres de production

Les requêtes d'observation sont émises par les utilisateurs du système. Chaque observation est définie par :

- une zone géographique à observer qui est un polygone découpé en bandes (voir Figure10.2) ; chaque bande est référencée par ses coordonnées géographiques, une durée d'observation, et un volume de données ; et parfois des prévisions météo pour la zone ; l'observation doit être réalisée dans une des fenêtres de visibilité de la bande ;
- un niveau de priorité ; les priorités sont dites *étanches* car il est toujours préférable de satisfaire une requête de haute priorité plutôt que n'importe quel ensemble de requêtes de plus basse priorité ;
- un poids, permettant de trier les acquisitions qui ont le même niveau de priorité ;
- des angles d'observation maximum : plus l'angle d'observation est grand, moins la qualité est bonne ; la meilleure qualité est donnée par un pointage nadir, c'est à dire quand le satellite pointe vers le centre de la Terre.

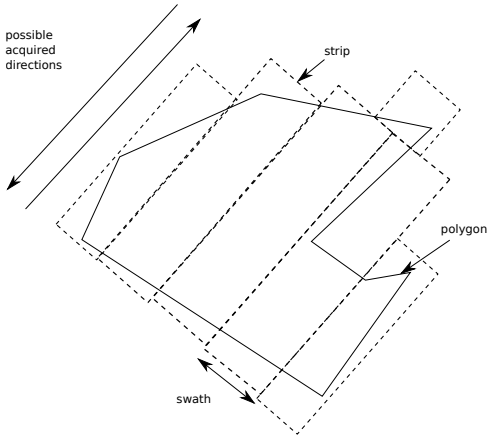


Figure 10.2 – Une zone géographique découpée en bandes.

Une fois que les observations ont été réalisées, les données brutes associées sont envoyées vers les centres de production qui les transforment en images livrables. Chaque utilisateur du système a son centre de production préféré.

10.1.5 Satellite géostationnaire

Le réseau de stations de réception pour les satellites à orbite basse n'est pas assez étendu pour couvrir toute la surface de la Terre. Les satellites géostationnaires ont une altitude d'environ 36000 kilomètres ce qui leur donne une large couverture (1/3 de la surface terrestre). Ce genre de satellite est fréquemment utilisé en télécommunications et en météorologie. Grâce à leur orbite géostationnaire, leur pointage est stationnaire au-dessus d'un point donné. Quand un satellite à orbite basse survole une zone sans station de réception, la seule façon de transférer immédiatement des données au sol est de les transférer via un satellite géostationnaire grâce à un lien radio ou optique. Un satellite géostationnaire, avec sa couverture beaucoup plus large, sera capable de vider ces données sur une station de réception sol inaccessible au satellite à orbite basse. Dans certaines missions, il est possible d'utiliser un tel réseau de satellites géostationnaires, comme le *European Data Relay System*.

10.2 Caractéristiques des satellites d'observation de la Terre

Dans ce qui suit, on décrit les caractéristiques d'un satellite d'observation de la Terre typique. On parle d'*attitude* pour désigner l'orientation du satellite par rapport à son repère inertiel de référence ou par rapport à une autre entité comme par exemple une planète. Un satellite est capable de :

- faire des manœuvres en attitude via des actuateurs gyroscopiques ; voir Section 10.2.2 pour plus de détails ;
- faire des manœuvres orbitales via des réacteurs à ergols ;
- réaliser des acquisitions via l'instrument d'observation ;
- vider les observations via l'antenne de télémétrie ;
- faire des pointages héliocentriques, vers le soleil, pour recharger les batteries avec de l'énergie solaire, et des pointages quasi-héliocentriques, pour favoriser le vidage des données tout en rechargeant les batteries ;
- faire des pointages géocentriques, vers le centre de la Terre, pour rendre les communications avec les stations sol plus faciles.

10.2.1 Orbite

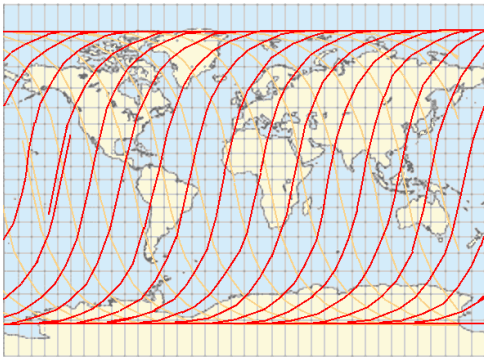


Figure 10.3 – Trace au sol d'une orbite héliosynchrone.

Le satellite est placé sur une orbite géocentrique basse à une altitude de 700 à 1000 kilomètres. Cette orbite est :

- **héliosynchrone**, l'angle entre le plan orbital et la direction du soleil est plus ou moins constant bien que le bourrelet équatorial entraîne une précession de cette orbite ;
- **quasi-polaire**, l'orbite est très légèrement inclinée par rapport à une ligne allant du pôle Nord au pôle Sud ce qui fait que l'orbite passe presque au-dessus des pôles ;
- **de faible périodicité**, le satellite revient fréquemment au-dessus d'un point à la même heure solaire.

L'héliosynchronisme de l'orbite combiné à la rotation de la Terre permet à l'orbite de couvrir presque entièrement la surface de la Terre dans une période donnée (voir Figure 10.3). De plus, ces caractéristiques orbitales favorisent un ensoleillement identique à chaque revisite au-dessus d'un point, ce qui est une caractéristique intéressante pour suivre l'évolution d'un phénomène au sol.

10.2.2 Plateforme

Le satellite considéré dans ce manuscrit sera compatible avec le lanceur européen VEGA, qui est spécialisé dans la mise en orbite basse de charge utile. Le satellite pèsera moins de 1500 kilogrammes, sa consommation d'énergie sera d'environ 2 kilowatts, sa durée de vie sera de 7 à 10 ans, et il sera propulsé électriquement ou chimiquement.

Agilité Les satellites d'observation de la Terre qui ne sont pas *agiles* ont seulement un degré de liberté grâce à un miroir orientable installé en face de chaque instrument optique. Le miroir est actionné sur l'axe de roulis entre les acquisitions. Cela permet au satellite de prendre des images de zones situées à la droite ou à la gauche de sa trace au sol. Une image est produite par le mouvement du satellite sur sa trace au sol (voir Figure 10.4). Les fenêtres temporelles pendant lesquelles une acquisition d'image est possible peuvent être pré-calculée pour chaque requête d'observation. En effet, dans cette configuration, il n'y a qu'une seule opportunité sur chaque orbite pour réaliser une observation donnée (voir à la gauche de la Figure 10.5). Quand deux fenêtres temporelles se chevauchent ou sont trop proches, cela signifie que les deux observations associées ne sont pas compatibles. Un choix doit donc être fait. Donc, le problème de planification des observations inclut un problème de sélection.

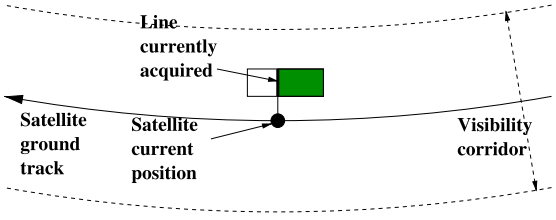


FIGURE 10.4 – Observation de la Terre avec un satellite non agile.

Dans ce manuscrit, on considère la planification et l'ordonnancement des activités pour un satellite *agile* d'observation de la Terre (comme les satellite Pléiades). Avec

ce type de satellite, les instruments et les panneaux solaires sont fixés (*body-mounted*) sur la plateforme (pour éviter les perturbations d'attitude à basse fréquence), et c'est la plateforme qui est mobile sur les trois axes (roulis, tangage et lacet) grâce à des actionneurs gyroscopiques [Chrétien et al., 2004]. Ces deux axes d'agilité supplémentaires permettent de décupler les opportunités pour chaque image. En effet, le satellite peut maintenant pointer en avant ou en arrière de sa position nadir (voir Figure 10.6). Le début d'une acquisition devient libre dans une fenêtre temporelle qui peut être pré-calculée (voir Figure 10.5). Ces fenêtres temporelles sont plus grandes que dans le cas non agile, et certaines acquisitions qui n'étaient pas compatibles deviennent compatibles grâce à cette agilité (voir à la droite de la Figure 10.5). Le problème de planification résultant inclut un problème de sélection et un problème d'ordonnancement, le rendant plus difficile.

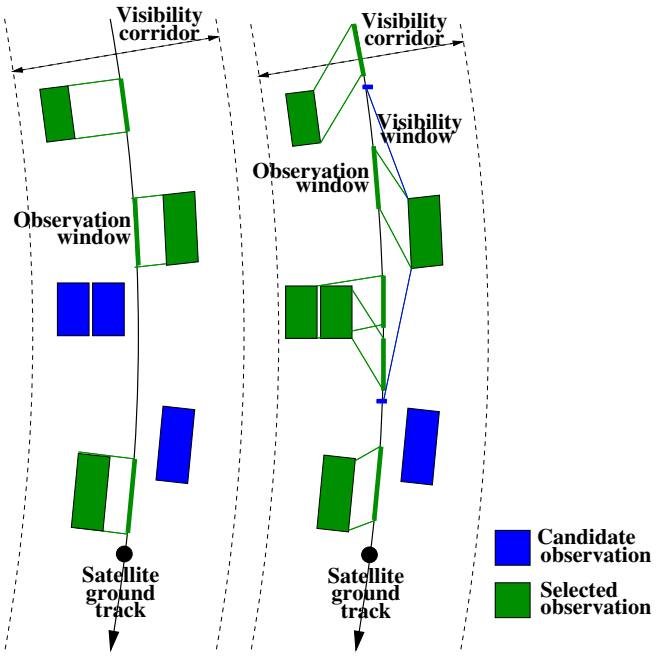


Figure 10.5 – Comparaison des capacités d'observations entre un satellite non agile (gauche) et un satellite agile (droite).

De plus, dans le cas des satellites agiles, la durée minimum de transition entre deux observations successives depend de la date de début de la transition (voir Figure 10.6). Calculer ces durées minimum est un problème d'optimisation continu sous contraintes [Beaumet et al., 2007], qui peut être résolu en utilisant des solveurs tels que RealPaver [Granvilliers and Benhamou, 2006] ou IBEX [Chabert and Jaulin, 2009], ou des techniques d'approximation.

La capacité du satellite à faire des mouvements en attitude combinée à sa trajectoire orbitale permet de réaliser des acquisitions en scannant la zone à acquérir. Les

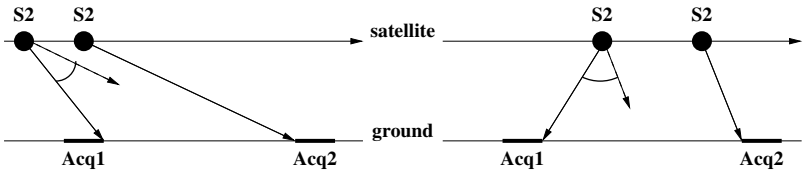


FIGURE 10.6 – La durée de transition entre deux acquisitions dépend de l'angle à couvrir entre la position de départ et la position souhaitée à l'arrivée, et dépend donc de la date de début de la transition.

profils d'observation, c'est à dire les mouvements en attitude réalisés pendant une observation, sont difficiles à calculer et dépendent de la date de début de l'observation.

Manœuvre orbitale Les manœuvres orbitales permettent de changer l'orbite d'un engin spatial grâce à un système de propulsion. Par exemple, les manœuvres orbitales sont nécessaires pour placer des sondes planétaires en orbite autour de planètes. Les satellites d'observation de la Terre dévient régulièrement de leur orbite. Il est alors nécessaire de faire une manœuvre orbitale pour rectifier la trajectoire. Ces manœuvres, qui consomment du carburant, sont réalisées au dessus de zones qui ne sont pas fréquemment observées, comme les pôles.

10.2.3 Chaîne d'acquisition image

On décrit maintenant la chaîne d'acquisition image à bord du satellite. Une illustration est visible en Figure 10.7.

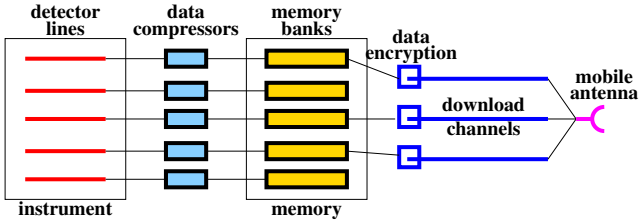


FIGURE 10.7 – Comment les données sont acquises, enregistrées, et vidées.

10.2.3.1 Télescope

Pour réaliser des observations, le satellite est équipé d'un télescope. Son large miroir primaire a une résolution de 20 à 30 centimètres et une fauchée de 15 à 20 kilomètres. Il est possible de prendre un minimum de 250 images par jour (avec un objectif de 500). Le plan focal est constitué de plusieurs lignes de détecteurs, chacune étant dédiée à une gamme de fréquence. Plusieurs lignes forment un canal image. Les contraintes suivantes doivent être considérées :

– un temps minimum de préchauffage est nécessaire avant utilisation du télescope ;

(a) Image très dense (b) Image nuageuse (c) Image peu dense

Figure 10.8 – Trois observations avec différentes caractéristiques. (a) est une image très dense qui ne sera presque pas compressée. (b) inclut des nuages qui seront compressés. (c) est une image peu dense qui sera fortement compressée.

- la température du plan focal augmente linéairement quand l'instrument est activé et décroit exponentiellement quand il est éteint ;
- la température du plan focal ne doit pas dépasser une certaine valeur ce qui limite la durée d'utilisation en continu ;
- il faut minimiser le nombre de cycles ON/OFF car ils dégradent la fiabilité de l'instrument ;
- l'éblouissement par le soleil est interdit : l'angle entre le pointage du satellite et la direction du soleil doit toujours être plus grand ou égal à une valeur minimale.

10.2.3.2 Mémoire de masse à bord

Le satellite possède une mémoire de masse à bord, gérée par un système de fichier standard, conçu pour stocker 2 jours d'acquisitions environ. Plusieurs modes instrument sont disponibles pour réaliser une observation. Chaque mode active certains canaux images. Chaque canal image est relié à une banque mémoire dans la mémoire de masse (voir Figure 10.7). Une observation est alors divisée en plusieurs fichiers, un par canal image. Une banque mémoire peut seulement stocker un nombre limité de fichiers. Supprimer une observation de la mémoire de masse est décidé par le sol après que l'observation ait été validée par les utilisateurs. Il est aussi possible de supprimer une acquisition de faible priorité pour faire de la place pour une acquisition de haute priorité ou quand celle-ci est trop vieille.

Après acquisition, une *compression sélective* est appliquée à l'image pour compresser les parties de l'image telles que les zones à faible densité ou les nuages (voir l'exemple sur la Figure 10.8). Le nombre de bits nécessaires pour encoder les pixels de ces zones peut être réduit de 12 à 3. Le volume de données généré par une acquisition est alors très variable (variant de 1 à 4 fois la taille minimum). Ce volume peut seulement être estimé avant acquisition puis constaté après acquisition. Jusqu'à maintenant, parce que les plans de vidage doivent être exécutables à bord, les volumes maximums (image sans compression sélective) sont considérés durant la phase de planification. Ces plans sont alors sous-optimaux car les vrais volumes sont très souvent plus faibles que le maximum (la compression est fréquente). Cette variabilité est une des motivations pour donner plus d'autonomie au satellite dans la prise de décision.

10.2.3.3 Antenne de télémétrie

Pour vider les données d'observation, la charge utile inclut une antenne montée sur le coté du satellite qui est dirigé vers la Terre et qui permet de couvrir pratiquement tout le demi-espace de vue. Cette antenne peut parcourir 5° par seconde et elle peut transmettre des données à débit de 2 Gbits par seconde. Son cône d'émission est de quelques degrés seulement ce qui veut dire qu'elle ne peut pointer plusieurs stations en même temps.

Une fois que le pointage de l'antenne a été décidé, le suivi satellite-station est automatique et prend en compte les mouvements en attitude. La station de réception et le satellite utilisent les éphémérides pour se suivre mutuellement. Il est aussi possible d'avoir un suivi dynamique par asservissement sur le signal entre le satellite et la station.

La planification des vidages doit prendre en compte la vitesse de rotation de l'antenne, l'attitude du satellite et les coordonnées des stations. Une durée minimum de transition est requise avant de commencer une communication avec une station sol, même si le satellite se situe dans le cône de visibilité de la station (ce qu'on appelle aussi *pull-in time*).

Le débit de l'antenne dépend de la distance entre le satellite et la station. Quand le satellite est loin de la station, la dispersion du signal induit un taux d'erreur important. De plus, à une distance importante, le satellite est rasant, ce qui fait que le signal traverse une couche atmosphérique plus grande que lorsque le satellite est au zénith de la station (voir Figure 10.9). Jusqu'à maintenant, le débit était fixé au pire débit possible, obtenu quand la distance satellite/station est maximum. Dans le système futur, le débit est dit *intelligent* car il varie durant les fenêtres de communication en fonction du taux d'erreur (voir Figure 10.10).

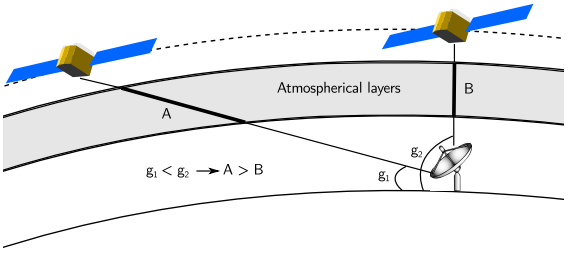


FIGURE 10.9 – Plus le satellite est rasant, plus la couche atmosphérique à traverser est épaisse pour le signal.

10.2.3.4 Vidage des données

Le satellite peut télé-décharger (*vider*) les observations quand il est *visible* depuis une station de réception (quand le satellite est dans une *fenêtre de visibilité*), quand son antenne de télémétrie est orientée vers la station, et quand l'antenne de la station de la réception pointe vers le satellite. A ce moment, la communication effective devient possible pour une certaine durée que l'on nomme *fenêtre de vidage*. Les observations sont alors vidées selon un certain ordre déterminé par le *plan de vidage*.

Le vidage des données se fait sur plusieurs *canaux d'émission* en parallèle (voir Figure 10.7). Chaque fichier résultant d'une observation peut être vidé sur un canal. Il

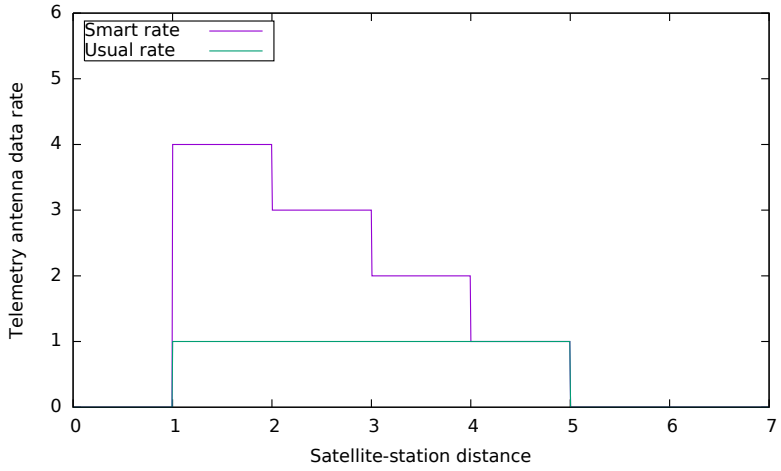


Figure 10.10 – Silhouette du débit de l'antenne de télémétrie en fonction de la distance satellite-station.

n'est pas acceptable d'interrompre un vidage. Étant donné le mouvement du satellite sur son orbite et le mouvement de la Terre sur elle-même, la distance satellite-station évolue et donc la durée de vidage d'un fichier dépend de la date de début de ce vidage. Les fichiers d'une acquisition doivent tous être vidés pendant la même fenêtre de visibilité, ils peuvent toutefois être vidés selon n'importe quel ordre sur n'importe quel canal. Les canaux et les banques mémoire sont des ressources non-partageables ce qui signifie que si deux fichiers sont enregistrés sur la même banque mémoire ou utilisent le même canal, leurs vidages ne peuvent pas se chevaucher temporellement.

Les données sont chiffrées avant vidage. Une clé de chiffrage est associée avec chaque utilisateur (e.g. un pays, un groupe de pays, ou une entreprise). Il y a un composant physique pour chaque canal d'émission ce qui permet à des données destinées à plusieurs utilisateurs d'être transférées de manière concurrente. Changer la clé de chiffrage sur un des composants est immédiat mais il est nécessaire d'inscrire ce changement dans une *table de changements de clés*. Le nombre d'entrées dans cette table est limité. Procéder à des changements dans la table prend une certaine durée, typiquement deux secondes, et nécessite d'arrêter tous les vidages en cours.

10.2.3.5 Allumage des instruments

Pour économiser de l'énergie, garder la température des instruments en dessous d'un certain seuil, et maximiser leur durée de vie, les instruments d'acquisitions et de vidage (l'instrument optique et l'antenne d'émission) ne sont pas toujours allumés. Chacun a trois modes : ON, standby (SDBY), et OFF, en ordre décroissant de consommation d'énergie. Les instruments sont maintenus ON seulement si nécessaire : durant une acquisition ou un vidage. Entre les acquisitions et les vidages, ils sont maintenus en mode SDBY si et seulement si la durée entre deux acquisitions ou

vidages est plus petite ou égale à un certain seuil. Sinon, les instruments sont éteints (en mode OFF).

10.2.3.6 Température des instruments

Si on considère des plans d'acquisition, de vidage, et le plan d'allumage des instruments, on peut calculer une approximation du profil en température pour chaque instrument et vérifier si les niveaux de température se situent dans les limites acceptables. Des marges importantes sont utilisées dans ce processus car l'évolution de la température et les interactions thermiques entre composants sont difficiles à modéliser.

10.2.3.7 Énergie à bord

A bord, de l'énergie est consommée par la plateforme du satellite, ainsi que par les instruments permettant l'acquisition de données et le vidage en fonction du mode dans lequel ils sont (ON, SDBY, ou OFF). Cette énergie est produite par les panneaux solaires qui sont fixés au satellite. Puisque le satellite a une orbite quasi-polaire et une altitude basse et héliosynchrone, il alterne entre des période de jour et de nuit (le soleil est éclipsé par la Terre pendant les périodes de nuit). La production d'énergie est possible seulement pendant les périodes de jour et, à l'intérieur de ces périodes, elle dépend de l'attitude du satellite. L'énergie produite est stockée dans des batteries dont la capacité est limitée. $Emax$ est le niveau d'énergie maximum, égal à la capacité de la batterie : le niveau d'énergie ne peut pas être plus grand que $Emax$. $Emin$ est le niveau minimum acceptable d'énergie : le niveau d'énergie à bord ne peut jamais passer en-dessous de $Emin$. Si on considère les plans d'acquisition, de vidage, ainsi que le plan d'allumage des instruments, on peut calculer une approximation du profil en énergie et vérifier que celui-ci reste toujours au-dessus de $Emin$.

10.3 Processus de décision et incertitudes

Le satellite évolue dans un environnement dynamique et plusieurs sources d'incertitudes existent. En voici quelques unes :

- la couverture nuageuse, conditionnant le succès des observations ; des prévisions météo sont utiles mais souvent imprécises ;
- le volume des fichiers après compression sélective ; étant donné que la couverture nuageuse n'est pas connue à l'avance, le taux de compression n'est pas connu non plus ;
- la fiabilité du lien bord/sol, perturbée par les interférences ; la qualité du lien peut altérer les durées de transfert ;
- l'énergie disponible à bord, difficile à modéliser car elle varie en fonction de la production des panneaux solaires et de la consommation par les systèmes bord ;
- l'arrivée de requêtes d'observation urgentes.

Ces incertitudes influencent la planification d'activité. On dit que l'environnement et le problème de planification en résultant sont *dynamiques* parce que les paramètres du problème évoluent rapidement. Les problèmes de planification et d'ordonnancement sous incertitudes possèdent de grands espaces de recherche (avec plusieurs milliers d'activités dans notre cas). Les plans qui ne prendraient pas en compte ces incertitudes sont destinés à échouer [Bresina et al., 2002b]. L'approche courante consiste à construire des plans hors-ligne au sol (toutes les 8 heures avec un

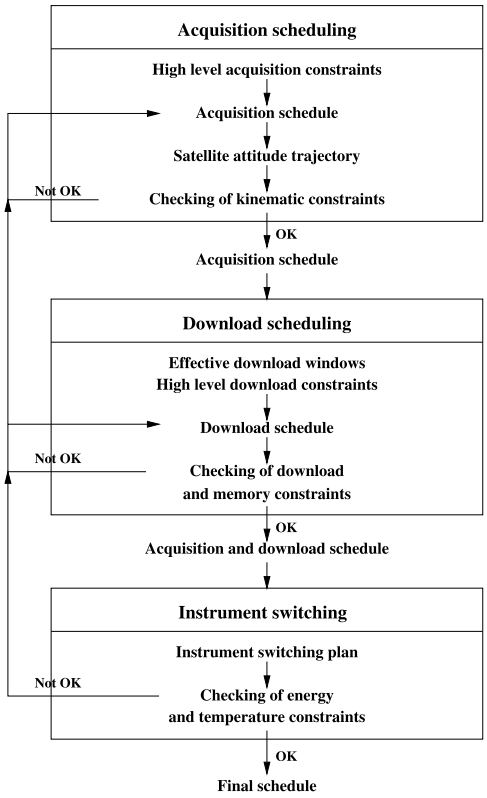


Figure 10.11 – Vue générale du processus de construction des plans d'acquisition, de vidage, et d'allumage des instruments.

horizon glissant de 24 heures), dans lesquels les activités ont des dates de début et de fin fixées. Toutes les contraintes relatives à l'attitude du satellite, à l'acquisition et au vidage, à l'énergie, à la mémoire, et à la température, sont vérifiées au sol. Pour être certain que les plans soient exécutables sans changements, des marges de sureté sont utilisées tout au long du processus. Par exemple la production d'énergie espérée est remplacée par la pire production possible. Cela résulte en un problème encore plus contraint. Les plans sont ensuite envoyés au satellite durant une fenêtre de visibilité de station de contrôle et exécutés strictement par le satellite.

Le but de cette thèse est d'aller au-delà de ce schéma et d'améliorer les performances par une meilleure gestion des incertitudes. Dans le prochain chapitre, nous verrons les différentes façon d'organiser la décision pour parvenir à cela dans le cas de la planification des vidages.

Ordonnancement flexible des vidages 11

Notre première contribution [1] concerne le problème de vidage des acquisitions qui consiste à ordonnancer des vidages de fichiers durant des fenêtres de communication entre le satellite et les stations de réception sol. Des algorithmes sophistiqués compressent les zones de faible intérêt sur les images acquises à bord, comme par exemple les nuages ou les terrains uniformes. Ainsi, la quantité de données résultant d'une acquisition, devant être enregistrée à bord, et ensuite vidée au sol est imprévisible puisque qu'elle dépendra des données acquises et de l'environnement à l'instant de la prise de vue.

Comme il n'est pas possible de communiquer continuellement avec les satellites, les volumes de données générés sont d'abord connus à bord avant de pouvoir être communiqués au sol. Cela mène à penser que les décisions concernant les vidages devraient être prises à bord juste avant les fenêtres de communication avec les stations sol. A ce moment les volumes réel des données sont connus et la planification à bord peut donc produire des plans d'une grande qualité. Cependant la planification à bord prend du temps et des ressources informatiques importantes. De plus, les plans produits ne sont pas connus au sol avant leur exécution et le comportement du satellite est donc imprévisible du point de vue du contrôle opérationnel au sol. Cela est problématique, particulièrement pour les acquisitions de haute priorité car les utilisateurs ayant demandé ces acquisitions peuvent parfois exiger de savoir quand les données seront disponibles. Dans ces conditions, il est toujours possible de construire des plans de vidage au sol avec des hypothèses pire-cas sur les volumes de données, c'est à dire en supposant que tous les volumes sont maximum (cela correspondant à supposer que la compression à bord sera minimale). Ces plans seront toujours exécutables à bord mais largement sous-optimaux étant donné que les fenêtres de vidages seront sous-utilisées en réalité puisque les volumes effectifs sont bien souvent plus faibles que le maximum.

Dans ce travail, nous introduisons un mécanisme qui résulte en plus d'autonomie à bord du satellite mais qui conserve une certaine prévisibilité pour les opérateurs au

1. Cette partie du travail a été présentée aux conférences nationales et internationales suivantes : ICAPS'14 [Pralet et al., 2014c], ICAPS'15 [Maillard et al., 2015a], i-SAIRAS'14 [Maillard et al., 2014], JFPC'14 [Pralet et al., 2014a], ROADEF'14, et ROADEF'15. Un article concernant l'approche mixte sol-bord pour le vidage des acquisitions a été accepté dans AIAA Journal of Aerospace Information Systems. Un article développant des aspects théoriques du sous-problème d'ordonnancement des vidages de fichiers est actuellement revu pour publication dans Discrete Applied Mathematics.

sol. Le problème de vidage est un problème complexe d'ordonnancement avec des contraintes temporelles et non temporelles, proche de problèmes de type *RCPSP/-max* ou *flexible job-shop*. Cependant, aucune technique existante ne peut être directement appliquée à notre problème car celui-ci présente des caractéristiques particulières telles que des contraintes temporelles de type *time-dependent* ou un objectif de juste répartition de la ressource dans le critère d'optimisation. Pour résoudre ce problème, l'idée est de construire des plans de vidages au sol et d'utiliser l'information la plus récente concernant les volumes pour adapter ce plan à bord. Plus précisément, un ordonnancement est produit au sol avec des hypothèses de volumes maximums pour les acquisitions de haute priorité et des hypothèses de volumes moyens pour les acquisitions de faible priorité. La planification au sol sur un large horizon temporel avec des capacités de calcul suffisantes permet de produire des plans de bonne qualité et de prendre en compte un critère d'optimisation complexe incluant la juste répartition de l'utilisation du système parmi les utilisateurs ou la minimisation de l'age de l'information, ce qui ne serait pas possible à bord. Pour faire cela, on utilise un algorithme de type *Squeaky Wheel Optimization* au-dessus d'un algorithme glouton non-chronologique. Les nombreuses contraintes du problème sont vérifiées avec la librairie InCELL. De par l'utilisation des volumes moyens pour certaines acquisitions, le plan n'est pas directement exécutable à bord. Certains volumes peuvent être plus grands que prévus et il faut donc réparer le plan. Une adaptation du plan sol est donc entreprise avant chaque groupe de fenêtres de visibilité. Un algorithme glouton rapide construit un plan exécutable depuis le plan sol en supprimant du plan les vidages de faible priorité s'ils mettent en danger la réalisation future des vidages de haute priorité. Des bornes sur la disponibilité des ressources, calculées au sol, permettent de vérifier rapidement si le volume réel des acquisitions de faible priorité va mettre en danger les vidages futurs. Avec un mécanisme de recherche en avant, on assure que toutes les vidages de haute priorité seront effectués. Quand les volumes réels sont plus faibles que prévus, la procédure d'adaptation essaie d'insérer des vidages qui n'étaient pas prévus par le plan sol ou d'avancer des vidages plus tôt dans le plan. L'adaptation à bord permet de profiter de la connaissance des vrais volumes et de mettre à jour le plan dans un temps acceptable (en comparaison avec la planification totalement à bord). La combinaison de ces deux phases de planification dans un mécanisme flexible rend cette approche très compétitive, parfois même meilleure que la planification purement à bord en terme de critère d'optimisation, et beaucoup plus prévisible que celle-ci.

11.1 Description du problème de vidage

Comme vu précédemment, après que le satellite ait acquis des données, il doit les vider durant une fenêtre de visibilité de station de réception. Dans ce chapitre, on décrit de manière informelle le problème d'ordonnancement des vidages menant à un plan de vidage et les différentes méthodes que nous avons mis en œuvre pour le traiter. Remarquons que les contraintes de ce problème proviennent d'un système réel et qu'elles rendent le problème plus difficile à traiter qu'un problème théorique d'optimisation.

Dans le reste de ce résumé, les mots *observation* et *acquisition* d'un coté, et *fenêtre de vidage* et *fenêtre de visibilité* d'un autre coté, se réfèrent au même concept.

Pour bâtir un plan de vidage, des décisions doivent être prises à trois niveaux différents :

1. la sélection, l'affectation et l'ordonnancement des vidages d'observations : cela consiste à décider quelles observations vont être vidées et à bâtir une séquence de vidages d'observation, chaque vidage étant constitué d'une observation o et d'une fenêtre de vidage w, allouée à o ; ce problème est proche du problème de type multi-sac-à-dos [Kellerer et al., 2004] ou les objets sont les observations et les sacs sont les fenêtres, mais , dans notre cas, les premiers sacs sont préférés puisque la fraicheur de l'information doit être maximisée, et des conflits peuvent exister entre sacs (chevauchements entre fenêtres) ;
2. affectation et ordonnancement des fichiers : chaque observation génère plusieurs fichiers ; une fois qu'une observation a été assignée à une fenêtre de vidage, ce sous-problème consiste à assigner chaque fichier de cette observation à un canal d'émission, et à bâtir pour chaque canal d'émission et chaque banque mémoire, une séquence de vidages de fichiers ; dans chaque fenêtre de vidage, ce problème est proche du *Flexible Open-shop Scheduling problem* [Pinedo, 2012] avec deux types de ressources non-partageables : canaux et banques mémoires ; chaque vidage de fichier a besoin d'une ressource de chaque type, mais le choix du canal est libre alors que la banque est pré-allouée ; l'ordre de vidage sur les canaux et les tranches est aussi libre ;
3. gestion des dates : cela consiste soit à calculer les dates au plus tôt et au plus tard pour tous les évènements, soit à fixer les dates précises ; les événements concernés sont les débuts et fins de vidages de fichiers, de mouvements d'antenne, et de mises à jour de la table des changements de clés ; le problème temporel résultant a la forme d'un *Simple Temporal Network* ; la seule exception concerne les durées de vidages de fichiers et de transitions entre fenêtres de vidage qui sont dépendantes de leurs dates de début (elle sont dites *time-dependent*) ; le résultat est un *Time-dependent STN*, pour lequel les techniques développées sur les STN peuvent être étendues.

Ces décisions doivent satisfaire toutes les contraintes qui sont détaillées ci-après. Beaucoup de celles-ci proviennent des limitations du système physique : par exemple, les durées de vidage. D'autres proviennent de contraintes utilisateur : par exemple, le fait que tous les fichiers générés par une observation doivent être vidés sur une station s et durant une de ses fenêtres de vidage associée. Un autre exemple est la notion d'*engagement* qui peut être associé à certaines observations, empêchant certaines décisions prises au sol d'être modifiées à bord. Plus de détails seront données dans la Section 11.2.3. Beaucoup de contraintes sont temporelles : par exemple, l'absence de chevauchement entre vidages de fichiers sur les canaux et les tranches. Certaines ne sont pas temporelles : par exemple, le fait qu'une observation demandée par l'utilisateur u doive être vidée sur une station autorisée par u.

Les décisions doivent aussi optimiser un critère global qui prend en compte les observations vidées, leurs priorités, l'age de l'information (la distance temporelle entre une observation et la livraison des données à l'utilisateur les ayant demandées), et la juste répartition du système satellite entre les utilisateurs.

Pour illustrer le problème de planification des vidages, voici un exemple. Cet exemple implique deux fenêtres de vidage 1 et 2 se chevauchant respectivement associées avec deux stations de réception 1 et 2, huit observations à vider de A à F, deux niveaux de priorité 1 et 2, trois utilisateurs numérotés de 1 à 3, cinq banques mémoire numérotées de 1 à 5, et trois canaux d'émission numérotés de 1 à 3. Le tableau 11.1 montre les priorités et les utilisateurs associés à chaque observation, ainsi que les fichiers générés par chaque observation et les banques mémoire sur lesquelles ceux-ci sont enregistrés. Par exemple, l'observation A est de priorité 2 (faible prio-

rité), est demandée par l'utilisateur 3, et génère trois fichiers qui sont enregistrés sur les banques mémoire 1, 2, et 4. De plus, on suppose que l'utilisateur 2 autorise seulement la station 2 pour vider ses données, et que toutes les observations sont finies avant le début des fenêtres 1 et 2, exceptée l'observation B qui finit durant la fenêtre 1.

Observations	A	B	C	D	E	F	G	H
Priorités	2	1	1	2	2	2	1	2
Utilisateurs	3	3	1	1	3	2	2	1
Banque 1	x	x	x		x	x		x
Banque 2	x	x	x				x	
Banque 3		x	x	x	x		x	
Banque 4	x	x	x	x			x	x
Banque 5		x	x	x	x	x		x

Table 11.1 – Priorités, utilisateurs, et fichiers générés associés avec chaque observation dans l'exemple.

La figure 11.1 montre un plan de vidage possible et toutes les contraintes temporelles associées. Dans ce plan, la séquence des vidages d'observation est C, H, B, G, E avec les observations C, H, et B vidées dans la fenêtre 1 et les observations G et E vidées dans la fenêtre 2. Les observations A, D, et F ne sont pas vidées (elles le seront plus tard). Si nous allons au niveau des fichiers, on peut voir sur quel canal et dans quel ordre les différents fichiers générés par les observations sont vidés : par exemple pour l'observation C, le fichier enregistré sur la banque mémoire 4 ($(C, 4)$) sur le canal 1, les fichiers enregistrés sur les banques 2 et 5 ($(C, 2)$ et $(C, 5)$) en séquence sur le canal 2...

Toutes les contraintes temporelles sont représentées sur la figure 11.1 :

1. les contraintes de précédence sur les canaux : ces contraintes expriment le fait qu'il ne doit pas y avoir de chevauchement entre les vidages de fichiers sur un même canal ; par exemple, le vidage de $(C, 2)$ doit précéder le vidage de $(C, 5)$ sur le canal 2 ;
2. les contraintes de précédence sur les banques mémoire : elles expriment le fait qu'il ne doit pas y avoir de chevauchement entre les vidages de fichiers enregistrés sur la même banque mémoire, par exemple, le vidage de $(H, 4)$ doit précéder le vidage de $(B, 4)$ parce que tous les deux sont sur la même banque mémoire ;
3. les contraintes de précédence sur les acquisitions : elles expriment le fait qu'une observation ne peut pas être vidée avant d'être terminée ; par exemple le vidage de $(B, 4)$ ne peut pas commencer avant t_1 qui est la fin de l'observation B (le début et la fin des observations sont fournies par le plan d'observation) ;
4. les contraintes de précédence sur les fenêtres de vidage : elles expriment le fait que tous les fichiers associés avec une observation o doivent être vidées dans la fenêtre affectée à o ; par exemple le vidage de $(G, 3)$ doit commencer après le début de s_2 de la fenêtre 2 et le vidage de $(E, 3)$ doit finir avant la fin e_2 de la même fenêtre ;
5. les contraintes sur les transitions de l'antenne : elles expriment le fait que dépointer l'antenne d'une station vers une autre prend du temps durant lequel

aucun vidage n'est possible ; par exemple, une distance minimum est requise entre la fin du vidage de $(B, 5)$ et le début du vidage de $(G, 2)$; une telle distance dépend des stations et de la date à laquelle la transition commence ;

6. les contraintes sur la mise à jour de la table de changements de clés : elles expriment le fait que mettre à jour la table contenant les changements de clé prenne un certain temps durant lequel aucun vidage n'est possible ; par exemple, parce qu'on profite de la transition antenne pour mettre à jour la table, une autre distance constante est nécessaire entre la fin de $(B, 5)$ et le début de $(G, 2)$;
7. les contraintes sur la durée des vidages : elles sont représentées sur la figure par la longueur des rectangles associées avec chaque vidage de fichier ; parce que le débit de vidage n'est pas constant durant une fenêtre de vidage et dépend de la distance satellite-station, les durées de vidage dépendent de leur date de début.

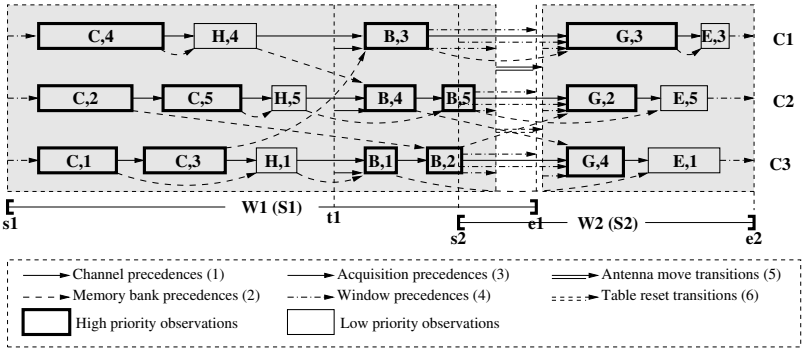


FIGURE 11.1 – Un plan de vidage et ses contraintes temporelles.

La figure 11.2 montre le plan au plus tôt associé ou tous les vidages sont ordonnancés au plus tôt possible.

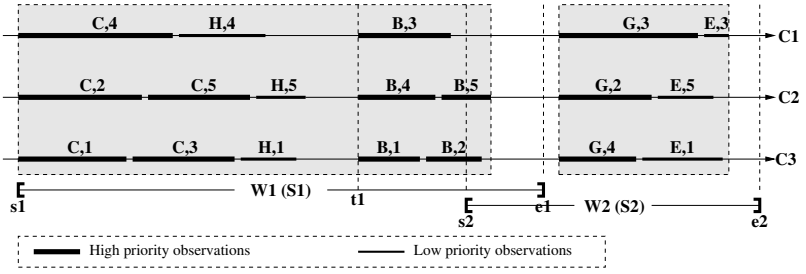


FIGURE 11.2 – L'ordonnancement au plus tôt associé.

11.2 Plusieurs processus de décision

Dans cette partie, on montre comment la prise de décision sur les plans d'observation et de vidage peut être organisée dans toutes les approches que l'on a développées et comparées.

11.2.1 Aperçu de l'approche sol

La figure 11.3 montre l'organisation globale de la prise de décision dans l'approche sol, dans laquelle toutes les décisions sont prises au sol par le centre de mission. Une telle approche est utilisée actuellement pour gérer les satellites d'observation de la Terre Pléiades.

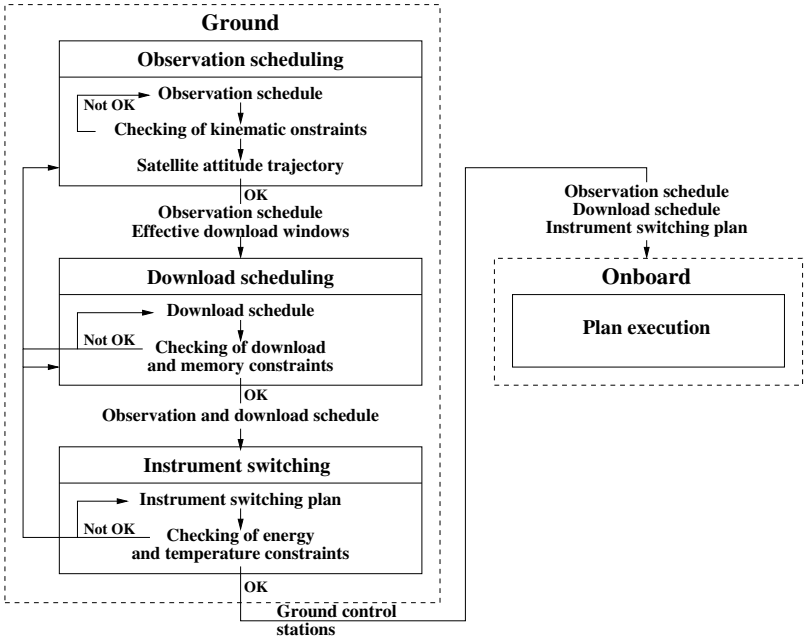


Figure 11.3 – Organisation globale de la prise de décision dans l'approche sol.

Planification des observations au sol Globalement, le sol construit régulièrement (typiquement toutes les huit heures) des plans sur un horizon glissant (typiquement un jour) en sélectionnant et en ordonnançant les observations à réaliser puis en vérifiant les contraintes cinématiques (l'orbite et les mouvement en attitude). Une fois qu'un plan d'observation a été choisi avec des dates fixes pour chaque observation, la trajectoire en attitude du satellite peut être calculée. Cette attitude permet de déterminer ensuite les fenêtres de vidages pour chaque station de réception en enlevant pour chaque fenêtre de visibilité station les sous-fenêtres pendant lesquelles le vidage est impossible à cause de l'attitude du satellite.

Planification des vidages au sol Le sol utilise le plan d'observation et les fenêtres de vidage effectives pour bâtir un plan de vidage sur le même horizon temporel en sélectionnant et en ordonnançant les vidages à faire et en vérifiant les contraintes relatives au vidage et à la mémoire (pas de surcharge). Pour être sûr que les plans de vidages sont exécutables, les volumes maximums sont considérés pour toutes les observations.

Ordonnancement de l'allumage des instruments au sol Une fois que les plans d'observation et de vidage ont été produits avec des dates de début précises pour les observations et les vidages, il est possible de bâtir un plan d'allumage des instruments (pour l'instrument optique et l'antenne de télémétrie) et de vérifier les contraintes d'énergie et de température, ainsi que les limitations relatives aux instruments en terme de durée maximale de mise en route sur l'horizon de planification.

Exécution des plans à bord Une fois que tous ces plans ont été produits au sol, ils sont envoyés au satellite durant une fenêtre de visibilité de station de contrôle, et exécutés tels quels, sans aucun changement.

La première justification pour cette organisation de la prise de décision est qu'il est nécessaire de connaitre le plan d'observation pour construire la trajectoire en attitude du satellite, les fenêtres de vidage effectives, et le plan de vidage, et qu'il est nécessaire de connaitre le plan d'observation et de vidage pour construire le plan d'allumage des instruments. La deuxième justification est qu'il a été observé dans les satellites Pléiades d'observation de la Terre actuellement en vol que les limitations en terme de capacité d'observation sont dans un ordre décroissant d'impact, (1) les contraintes cinématiques en attitude, (2) les contraintes relatives au vidage, (3) les contraintes relatives à l'énergie, (4) les contraintes relatives à la température des instruments, et (5) les contraintes relatives à la mémoire de masse. Vérifier d'abord les contraintes les plus limitantes est connu comme une bonne pratique.

11.2.2 Aperçu de l'approche bord

La figure 11.4 montre l'organisation globale de la prise de décision dans l'approche bord, dans laquelle toutes les décisions sur les vidages sont prises à bord. Cette approche a été explorée dans [Pralet et al., 2014c].

Planification des observation au sol La planification des observations est réalisée au sol dans le centre de mission exactement comme dans l'approche sol.

Ordonnancement de l'allumage des instruments au sol Parce que le sol ne sait pas quel sera le plan de vidage produit à bord, on suppose de manière pessimiste que toutes les fenêtres de vidage seront utilisées pour le vidage. Sur cette base, il est possible de bâtir un plan d'allumage des instruments et de vérifier les contraintes sur l'énergie et la température, ainsi que les limitations relatives aux instruments en terme de durée maximale de mise en route sur l'horizon de planification.

Planification des vidages à bord Le plan d'observation et les fenêtres de vidages effectives sont envoyés au satellite, en utilisant n'importe quelle fenêtre de visibilité de station de contrôle, et utilisés pour la planification des vidages. A bord, parce

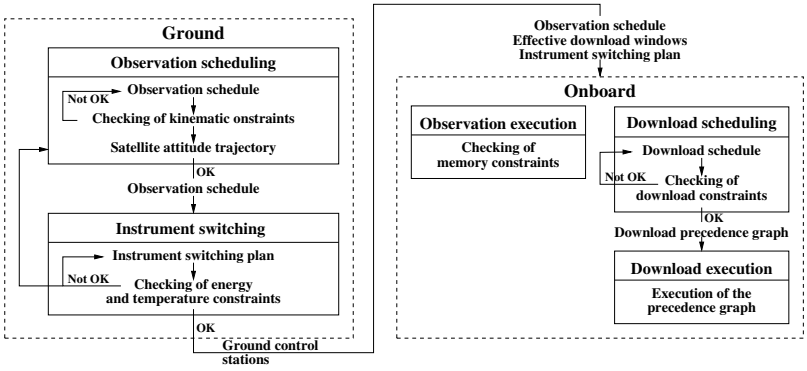


Figure 11.4 – Organisation globale de la prise de décision dans l'approche bord.

que l'information à propos des volumes arrive au moment où les observations se terminent, la planification des vidages est réalisée avant chaque groupe de fenêtres de vidage se chevauchant (au dernier moment, pour profiter du maximum d'informations concernant les volumes), avec ce groupe de fenêtres comme horizon de planification. Deux fenêtres de vidage sont considérées comme se chevauchant si et seulement si elles se chevauchent temporellement ou s'il n'y a pas assez de temps entre la fin de la première et le début de la seconde pour replanifier les vidages. Les volumes exacts sont considérés pour toutes les observations qui sont terminées au moment de la planification et les volumes maximums pour celles qui vont finir entre le début de la planification et la fin de l'horizon de planification, cela pour garantir que le plan est exécutable et n'aura pas besoin de modifications en ligne.

Exécution des plans d'observation et d'allumage à bord Les observations sont réalisées et les instruments sont allumés selon le plan d'allumage produit au sol. Cependant, parce que les contraintes sur la mémoire n'ont pas pu être vérifiées au sol (puisque le plan de vidage est produit à bord), un mécanisme simple est utilisé à bord lors de l'exécution des observations pour éviter le dépassement de la capacité mémoire et l'écrasement de données : si la réalisation de l'observation o peut mener à un écrasement mémoire, soit o n'est pas réalisée, soit une observation de plus faible priorité est effacée de la mémoire pour libérer de l'espace.

Exécution du plan de vidage à bord Les vidages sont exécutés tels que spécifiés par les plans de vidage, au cours des groupes de fenêtres de vidage successifs. Cependant, pour prendre en compte l'incertitude restante concernant le volume des données produit par les observations qui vont finir entre le début de la planification des vidages et la fin de l'horizon de planification, le plan de vidage envoyé pour exécution n'est pas fixe, mais reste temporellement flexible. Concrètement, ces plans ont la forme d'un graphe de précédence (*graphe orienté acyclique*) dans lequel chaque nœud est exécuté dès que tous ses prédécesseurs ont été exécutés. Cela permet de profiter des vidages dont la durée est strictement plus petite que la durée maximum qui a été considérée lors de la planification et ainsi de commencer les vidages plus tôt que prévu dans le plan pessimiste.

Garder la planification des observations au sol est justifié par le fait que cette planification nécessite de nombreux calculs de mouvements en attitude du satellite : des mouvements pour passer d'une observation à une autre et des mouvements précis pour réaliser les observations avec la qualité requise. Ces mouvements sont beaucoup plus complexes à calculer dans le cas des satellites agiles (puisque tout le satellite doit bouger) que dans le cas des satellites d'ancienne génération, comme les premiers satellites Spot (miroir mobile uniquement). Les ingénieurs considèrent que la technologie actuelle ne permet pas le calcul de ces mouvements à bord. Au contraire, déporter la planification des vidages à bord est justifié par le fait qu'elle prend beaucoup moins de temps que la planification des observations. Cette organisation est aussi justifiée par le fait que l'incertitude sur les volumes de données implique directement la planification des vidages, mais seulement indirectement la planification des observations.

11.2.3 Aperçu de l'approche mixte sol-bord

La figure 11.5 montre l'organisation de la prise de décision dans l'approche mixte sol-bord, principale contribution de ce chapitre, dans laquelle les décisions à propos des vidages sont partagées entre le sol et le bord.

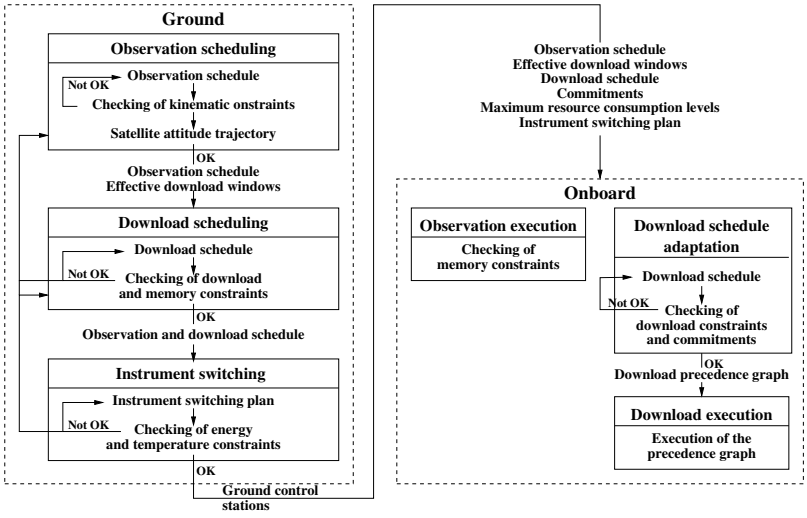


FIGURE 11.5 – Organisation de la prise de décision dans l'approche mixte sol-bord.

Avec chaque observation, est associé un niveau de priorité défini par un entier entre 1 et **Np**, avec 1 la plus haute priorité et **Np** la plus basse. Dans la suite, on se désigne par *observation de haute priorité* les observations de priorité 1 et *observation de basse priorité* toutes les autres, c'est à dire celles dont le niveau de priorité est plus grand ou égal à 2.

Planification des observations au sol La planification des observations est réalisée au sol dans le centre de mission comme dans les deux approches précédentes.

Planification des vidages au sol Le sol utilise le plan d'observation et les fenêtres de vidage effectives pour bâtir un plan de vidage sur le même horizon de planification en sélectionnant et en ordonnançant les vidages à faire et en vérifiant les contraintes concernant le vidage et la mémoire, comme c'est fait dans l'approche sol. La seule différence est que les volumes maximums sont utilisés seulement pour les acquisitions de haute priorité, de manière à garantir qu'elles pourront être vidées. Pour les acquisitions de basse priorité, pour lesquelles le même genre de garantie n'est pas requis, seuls les volumes espérés sont considérés. Avec chaque acquisition de haute priorité o, on associe un *engagement* de type 1 ou 2 défini par l'utilisateur qui a demandé o. Un engagement de type 1 associé avec une observation signifie que celle-ci doit être vidée durant la fenêtre de vidage décidée au sol. En d'autres termes, le vidage peut seulement être avancé durant la même fenêtre. Un engagement de type 2 signifie que l'observation doit être vidée dans la fenêtre décidée au sol ou plus tôt, possiblement dans une fenêtre de vidage antérieure.

Planification de l'allumage des instruments au sol Parce que le sol ne sait pas quel sera le plan de vidage produit à bord (puisqu'il sera adapté à bord), on suppose de manière pessimiste que toutes les fenêtres de vidage seront utilisées pour le vidage (même supposition que dans l'approche bord). Sur cette base, il est possible de bâtir un plan d'allumage des instruments et de vérifier les contraintes sur l'énergie et la température, ainsi que les limitations relatives aux instruments en terme de durée maximale de mise en route sur l'horizon de planification.

Adaptation du plan de vidage à bord Le plan d'observation, les fenêtres de vidages effectives, le plan de vidage (avec les engagements attachés aux acquisitions de haute priorité), et le plan d'allumage des instruments sont envoyés au satellite comme précédemment. Ces éléments sont utilisés pour l'adaptation du plan de vidage à bord. Comme dans l'approche bord, cette adaptation est réalisée avant chaque groupe de fenêtres de vidage se chevauchant avec ce groupe comme horizon de planification. Les volumes exacts sont considérés pour toutes les observations qui sont terminées au début de la planification et les volumes maximums sont considérés pour les observations qui se termineront d'ici la fin de l'horizon. La seule différence est que la planification ne se fait pas à partir de rien mais utilise le plan produit au sol. Elle l'adapte tout en garantissant les engagements relatifs aux acquisitions de haute priorité.

L'algorithme que nous proposons est chronologique et glouton, il construit un plan exécutable en suivant au maximum le plan de vidage produit en sol. Il le modifie parfois, (1) en avançant des acquisitions ou en insérant des acquisitions supplémentaires non présentes dans le plan quand des volumes plus faibles que prévus créent des espaces vides dans le plan, ou (2) en supprimant des vidages qui mettent en danger la réalisation des engagements des vidages liés à des acquisitions de haute priorité. La vérification à bord de la faisabilité des engagements après modification étant une opération potentiellement couteuse en temps, des bornes majorantes sur les ressources sont pré-calculées au sol et permettent des vérifications très rapides à bord.

Exécution des plans à bord Les plans d'observation, d'allumage, et de vidage sont exécutés à bord exactement comme dans l'approche bord précédemment présentée.

Figure 11.6 offre une autre vision de la façon dont la prise de décision est organisée temporellement entre le sol et le bord dans l'approche mixte sol-bord.

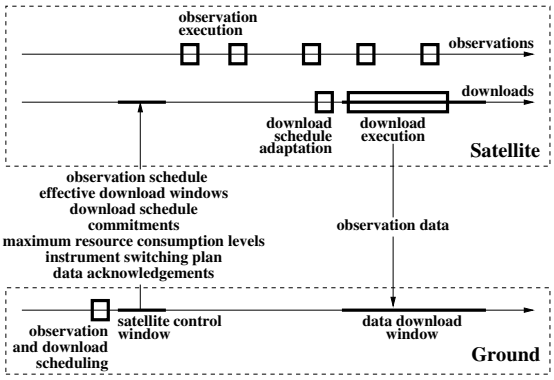


FIGURE 11.6 – Aperçu dont la prise de décision est organisée temporellement entre le sol et le bord dans l'approche mixte sol-bord.

La première justification de cette organisation de la prise de décision par rapport à la précédente est que, bien que la planification des vidages prenne bien moins de temps que la planification des observations, elle reste onéreuse si on considère les ressources de calcul très limitées à bord. Cela peut être une bonne idée de profiter des ressources de calcul disponibles au sol pour bâtir des plans de vidage de qualité et d'utiliser de simples règles de décision pour les adapter à bord. Ces plans sont produits en considérant les volumes maximums pour les acquisitions de haute priorité pour garantir leur vidage, et les volumes espérés pour les acquisitions de basse priorité avec l'idée que les volumes plus importants que les volumes prévus seront compensés par les volumes moins importants que prévus (voir [Bonfietti et al., 2014] pour une étude théorique et empirique de ce phénomène). La seconde justification est que cette organisation augmente la prévisibilité du système, pour les observations de haute priorité en tout cas : pour chacune d'elles, l'utilisateur sait dans quelle fenêtre ou dans quelle fenêtre au plus tard elle sera vidée.

11.3 Expérimentations

11.3.1 Approches à comparer

Pour apprécier l'impact positif d'un partage de la prise de décision entre le sol et le bord, nous avons implémenté et comparé les quatre approches suivantes dans un simulateur de satellite (présenté en anglais en Annexe A) :

1. planification au sol (`Ground`) : la planification des vidages est réalisée au sol à partir des volumes maximums pour toutes les observations, en utilisant un algorithme de type SWO reposant sur un algorithme glouton non-chronologique (décrit en anglais en Section 5.3) ; ce plan est ensuite exécuté à bord ; la seule différence avec l'approche actuellement en utilisation dans les opérations et que le graphe acyclique à bord permet une certaine flexibilité temporelle et permet à certains vidages de commencer plus tôt que dans le plan pessimiste du sol (décrit en anglais en Section 6.4) ; ce dernier mécanisme joue

plutôt en faveur de l'approche sol en terme d'age de l'information par rapport à la réalité ;

2. planification à bord (`Board`) : la planification des vidages est réalisée à bord à partir de rien avant chaque groupe de fenêtres de vidage se chevauchant avec les volumes exacts pour les observations étant déjà terminées et les volumes maximums pour les autres, en utilisant un algorithme glouton chronologique (comme décrit en anglais en Section 5.2.2) ; comme dans l'approche précédente, un graphe acyclique permet une certaine flexibilité temporelle ;
3. la planification des vidages est réalisée au sol avec les volumes maximums pour les observations de haute priorité et les volumes espérés pour les autres, en utilisant un algorithme SWO, comme dans l'approche sol ; ensuite, la planification est réalisée à bord avant chaque groupe de fenêtres se chevauchant avec les volumes exacts pour les observations déjà terminées et les volumes maximums pour les autres, en suivant et en adaptant le plan du sol (comme décrit précisément en anglais en Section 6.3) ; comme dans les approches précédentes, un graphe acyclique est construit et permet une certaine flexibilité temporelle ; deux variantes du mécanisme d'adaptation à bord ont été implémentées :

   a) un mécanisme d'adaptation très simple (`SimpleRepair`) : les vidages de n'importe quelle priorité sont supprimés du plan sol s'ils sont physiquement infaisables ; cela signifie que seules les contraintes physiques sont prises en compte ; les engagements vis-à-vis des observations de haute priorité ne sont pas considérés ;

   b) un mécanisme d'adaptation intelligent (`SmartRepair`) : les vidages de basse priorité sont supprimés du plan sol soit quand ils sont physiquement infaisables, soit quand ils empêchent les vidages de haute priorité futurs ; des vidages sont ajoutés ou avancés quand c'est possible ; cela veut dire que les engagements, ainsi que les contraintes physiques, sont toujours considérés ; pour vérifier que les engagements seront tenus, un mécanisme de recherche en avant est utilisé (voir Section 6.3 en anglais pour des détails).

11.3.2 Scénarios

On utilise deux scénarios réalistes fournis par Airbus Defence and Space. Les deux scénarios sont produits à partir du même plan d'observation de un jour qui a été fait pour un satellite d'observation de la Terre actuellement opérationnel (environ 14 révolutions autour de la Terre au cours de la journée). Ils diffèrent par rapport au nombre de stations sol disponibles pour le vidage : le scénario 1 (le moins contraint) implique un nombre important de stations sol de réception (23) résultant dans de nombreuses fenêtres de vidage (115), tandis que le scénario 2 (le plus contraint) implique un petit nombre de stations de réception (3) résultant en un petit nombre de fenêtres de vidage (20). Ils diffèrent aussi dans la distribution des observations parmi les utilisateurs et les niveaux de priorité : 275 (resp. 247) observations de haute priorité et 1089 (resp. 1117) observations de basse priorité pour le scénario 1 (resp. 2). Les volumes réels des fichiers sont générés de façon aléatoire dans l'intervalle compris entre $\mathbf{Vmax}/4$ et $\mathbf{Vmax}$, avec $\mathbf{Vmax}$ le volume maximum d'un fichier. 15 instances des volumes réels ont été générées et tous les résultats sont des valeurs moyennes sur ces 15 instances. Les expérimentations on été réalisées avec un processeur Xeon 4-coeurs 3GHz avec 8GB RAM, sur Ubuntu 12.04 LTS.

11.3.3 Critères d'évaluation

Les résultats d'exécution sont comparés par rapport à plusieurs critères :

1. le nombre de vidages, pour chaque niveau de priorité ;
2. l'age moyen de l'information sur toutes les observations vidées, pour chaque niveau de priorité ;
3. le critère global (décrit en anglais en Section 4.2.4.2), pour chaque niveau de priorité ; parce que ce critère complexe agrège de nombreuses caractéristiques d'un plan (poids des observations, age de l'information, partage...), sa valeur exacte n'a pas vraiment d'importance ; ce qui est important, c'est qu'une approche en domine une autre ;
4. le temps de calcul moyen de la planification à bord ;

Les trois premiers critères mesurent la qualité des résultats d'exécution et le quatrième mesure le coût de la planification à bord.

11.3.4 Analyse des résultats

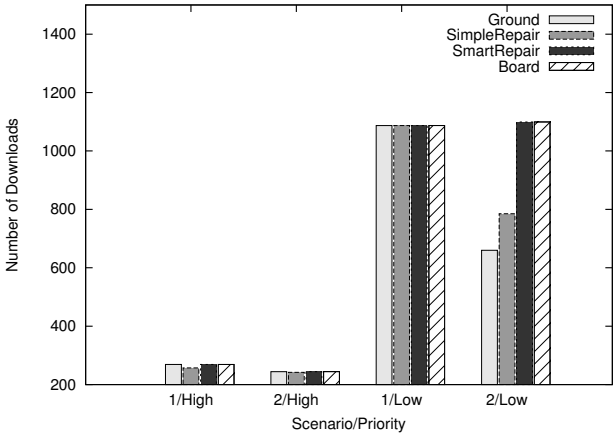


FIGURE 11.7 – Nombre de vidages. Plus il est grand, mieux c'est.

Nombre de vidages Par rapport au nombre de vidages (voir Figure 11.7), quelque soit le scénario, les observations de haute priorité sont toutes vidées, exceptées avec l'approche `SimpleRepair` qui ne vérifie pas que les observations futures soient menacées par l'insertion d'un vidage de faible priorité. Les observations de basse priorité sont toutes vidées dans le scénario 1, mais pas dans le scénario 2. Dans ce dernier scénario (le plus contraint), seules les approches `SmartRepair` et `Board` permettent de les vider.

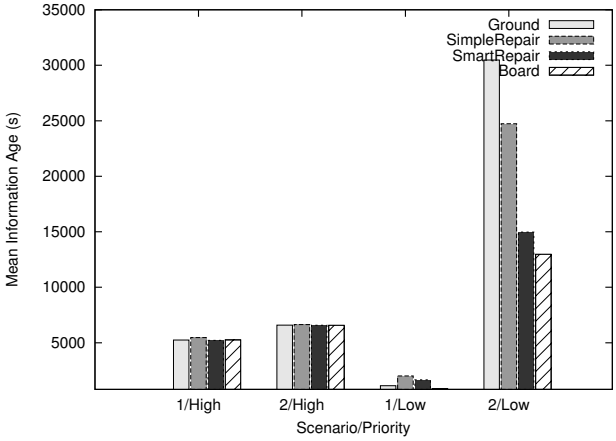


Figure 11.8 – Age moyen de l'information (en secondes). Plus il est petit, mieux c'est.

Age moyen de l'information Des différences importantes apparaissent sur la Figure 11.8, mais seulement pour les observations de basse priorité sur le scénario 2 (le plus contraint). L'age moyen de l'information diminue à mesure que la prise de décision est déportée à bord.

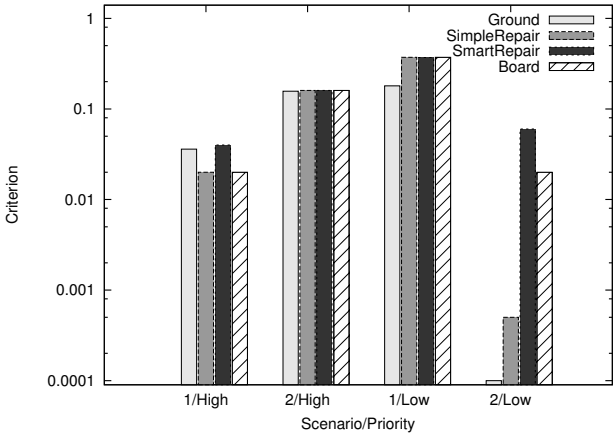


Figure 11.9 – Critère global (sur une échelle logarithmique). Plus il est grand, mieux c'est.

Critère global Par rapport au critère global, la Figure 11.9 montre que, pour les observations de haute priorité sur le scénario 1, les approches `Ground` et `SmartRepair`

sont meilleures que les approches `SimpleRepair` et `Board`. Pour l'approche `SimpleRepair`, cela peut être expliqué par le fait que l'algorithme d'adaptation supprime des vidages de haute priorité. Pour l'approche `Board`, cela peut être expliqué par le fait que les planifications bord sont effectuées successivement sur des horizons très limités, tandis que les approches `Ground` et `SmartRepair` travaillent sur un grand horizon de planification et optimisent explicitement le critère global avec l'algorithme SWO. Pour les observations de basse priorité, l'approche `SmartRepair` est clairement supérieure sur le scénario 2, suggérant l'impact positif d'une combinaison entre un algorithme d'optimisation au sol sur un large horizon et une adaptation du plan à bord sur des petits horizons successifs.

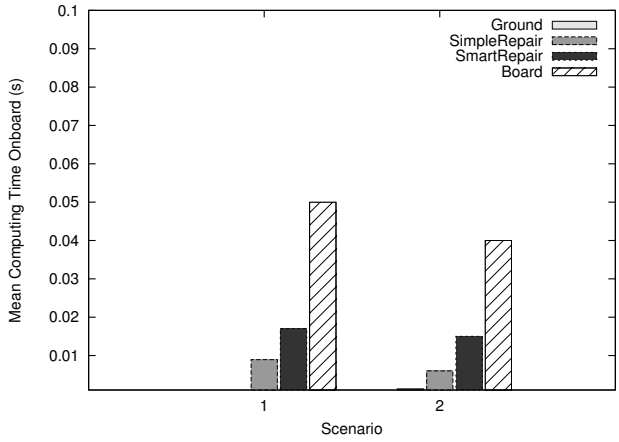


FIGURE 11.10 – Temps moyen de calcul à bord. Plus il est petit, mieux c'est.

Temps moyen de calcul à bord Sans surprise, les résultats sur la Figure 11.10 montrent que plus la prise de décision est déportée à bord, plus le temps moyen de calcul à bord est important (ce temps est trop faible pour être montré pour l'approche `Ground`). Signalons que l'approche `SmartRepair` est capable d'obtenir des résultats similaires ou supérieurs en terme de nombre de vidages, d'age de l'information, et de critère global, par rapport à l'approche bord avec 1.5 à 3 fois moins de temps de calcul à bord.

Planification flexible des acquisitions 12

Nous avons aussi examiné le problème de planification des acquisitions qui consiste à sélectionner et ordonnancer les observations que le satellite réalisera [1]. De nos jours, les plans d'observation sont produits au sol. Un plan de vidage est ensuite construit et le tout est simulé pour vérifier que les contraintes sur l'énergie, la température et la mémoire sont bien respectées par le plan. Si certaines contraintes sont violées, des acquisitions de faible priorité sont supprimées du plan d'observation et de vidage et le processus de simulation est relancé. Parce que la simulation de l'évolution de l'énergie ou de la température est une opération nécessitant de fortes ressources de calcul, cela ne peut pas être fait finement durant la planification. Cette simulation fine est donc remplacée par des contraintes de haut niveau ajoutées au problème. Ces contraintes sont très margées par rapport à la réalité pour assurer la sûreté du satellite. Les paramètres affectés par ces marges sont la consommation et la production d'énergie, l'évolution de la température et le volumes des observations. La consommation d'énergie réelle est souvent moins importante que celle prévue par les marges, la production d'énergie réelle est souvent plus importante que celle prévue par les marges, et ainsi de suite. Le profil d'énergie est donc toujours plus haut que prévu et pourtant, un nombre important d'acquisitions qui auraient pu être réalisées et vidées sont éliminées de la planification à cause des contraintes de haut niveau. Pour améliorer la capacité du système, l'idée développée ici est de supprimer ces contraintes de haut niveau pour les acquisitions de faible priorité. Cependant, et contrairement au problème de vidage, il n'est pas possible d'adapter les plans à bord à cause des faibles capacités de calcul disponibles à bord et de la difficulté d'adaptation qui est plus beaucoup grande. Cela signifie que le bord peut seulement exécuter les plans. Dans une nouvelle approche, les plans d'observation produits au sol sont des plans conditionnels contenant des conditions pour déclencher les acquisitions de faible priorité. Bâtir ces plans conditionnels implique le remplacement des contraintes de haut niveau par un mécanisme dans lequel on calcule d'abord les niveaux minimums d'énergie devant être présents à bord au début de chaque acquisition de faible priorité de manière à assurer que si on réalise cette acquisition, toutes les futures acquisitions de haute priorité pourront être effectuées. Ces niveaux sont produits grâce à une procédure de simulation à rebours qui commence à la fin de l'horizon et qui calcule des bornes de réalisation sur les niveaux d'énergie. Une fois à bord, avant chaque acquisition de faible priorité, si le niveau d'énergie réel est plus important que le niveau requis calculé au sol, l'acquisition est réalisée. Sinon, l'instrument d'observation n'est pas allumé et

1. Cette partie du travail a été présentée au IJCAI'15 International Workshop on Planning & Scheduling for Space (IWPSS'15) [Maillard et al., 2015b].

de l'énergie est donc économisée. Comparée avec l'approche actuelle, cette approche évite le gaspillage d'énergie et permet d'augmenter le nombre d'acquisitions réalisées.

12.1 Quelques choix de conception

Dans la partie précédente, une approche flexible sol-bord a été proposée pour traiter l'incertitude sur le volume des données généré par les observations lors du vidage. Globalement, l'idée est de remplacer les vérifications sur le vidage et la mémoire par des vérifications plus légères qui ne concernent que les observations de haute priorité. Pour celles de basse priorité, la décision est reportée à bord et permet au plan de vidage produit au sol d'être modifié tout en garantissant que les engagements sont tenus concernant les acquisitions de haute priorité.

Dans cette partie, on étend cette approche pour traiter l'incertitude sur la production et la consommation d'énergie à bord. Tandis que l'on se concentre sur l'énergie ici, la même approche peut être appliquée à l'incertitude sur la température des instruments.

planifiées	$\sim 14\%$
rejetées - conflits cinématiques (contraintes d'agilité)	$\sim 51\%$
rejetées - contraintes de haut niveau sur l'énergie	$\sim 25\%$
rejetées - contraintes de haut niveau sur la température	$\sim 9\%$
rejetées - autres contraintes	$\sim 1\%$

Table 12.1 – Causes du rejet des observations durant la planification. Évaluation sur six scénarios Pléiades de 24 heures (avec une planification des observations par jour). Le nombre d'observations candidates varie de 1416 à 2560.

Un aspect crucial est que, comme cela l'a déjà été dit, la planification des observations est beaucoup plus complexe que celle des vidages. Il n'est pas envisageable de modifier les profils d'observation à bord du satellite. Dans ces conditions, on choisit de ne pas modifier le plan d'observation à bord. La seule liberté accordée au logiciel bord est la capacité, pour les acquisitions de basse priorité dans les plans d'observation, de les réaliser ou pas. Si le logiciel bord choisit de la réaliser, l'instrument optique est allumé. Sinon l'instrument est mis en mode SDBY ou OFF selon le délai avant la prochaine observation dans le plan (comme vu précédemment).

12.2 Production d'un plan d'observation conditionnel au sol

Ici, on suppose qu'une séquence $Seq = [o_i, ..., o_j]$ satisfaisant les contraintes en agilité a été produite. Un plan conditionnel est un plan d'observation classique, c'est à dire une séquence Seq d'observations, dans lequel un *niveau requis d'énergie* est associé avec chaque observation. Ce niveau requis d'énergie associé à l'observation $i \in Seq$ est le niveau minimum qui est nécessaire pour réaliser i et toutes les observations *obligatoires* qui suivent i dans Seq, c'est à dire toutes les acquisitions de haute priorité et toutes les activités telles que les périodes de vidage. Un exemple de plan conditionnel est donné en Figure 12.1.

Parce que le plan de vidage est flexible (voir l'approche précédente), on ne sait pas exactement ce qui va être vidé. Pour surmonter cette difficulté, on fait une supposition pessimiste : toutes les fenêtres de vidage, ou au moins celles utilisées pour vider

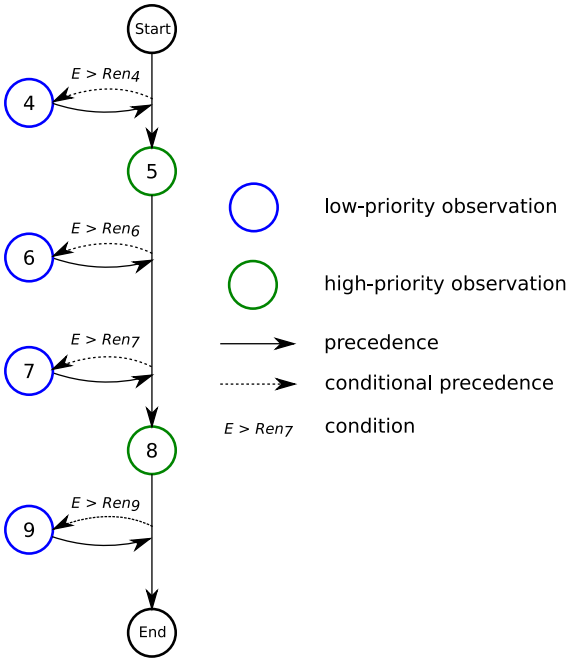


FIGURE 12.1 – Plan d'observation conditionnel. Les observations de haute priorité sont obligatoires et celles de basse priorité sont exécutées seulement si le niveau courant d'énergie à bord E est plus grand que le niveau requis d'énergie Ren calculé au sol.

les observations de haute priorité, seront utilisées ; ainsi, l'antenne va être maintenue allumée sur ces fenêtres et SDBY ou OFF entre ces fenêtres, suivant les règles présentées en Section12.1. Bien sûr, si elles se chevauchent, on suppose que l'instrument sera maintenu allumé seulement durant l'union des intervalles.

Les niveaux requis d'énergie peuvent être efficacement calculés par une procédure à rebours qui parcourt la séquence Seq d'observations de la dernière à la première. Les entrées de cette procédure sont :

- une séquence Seq d'observations sur un horizon temporel $[Hs, He]$ dans laquelle nous ajoutons deux observations fictives au début et à la fin de l'horizon, dont la durée est nulle et la priorité haute ; les observations dans Seq, ainsi que les fictives, sont numérotées de 1 à $|Seq|$; le plan d'allumage peut être calculé grâce aux règles présentées précédemment en Section12.1 ;
- un niveau $EnHs$ d'énergie au début de l'horizon Hs ;
- un niveau d'énergie requis $REnHe$ à la fin He de l'horizon ; ce niveau permet d'être certain que le satellite est dans un état sûr pour les prochaines orbites.

Une exécution de la procédure est présenté en Fig 12.2. Un sous-ensemble de la séquence d'observations est réprésenté de l'Acq. 4 à l'Acq. 9. Les observations de haute (resp. basse) priorité sont représentées en rectangles gras (resp. mince). Par exemple Acq. 8 est de haute priorité et Acq. 9 est de basse priorité. Le plan d'allumage

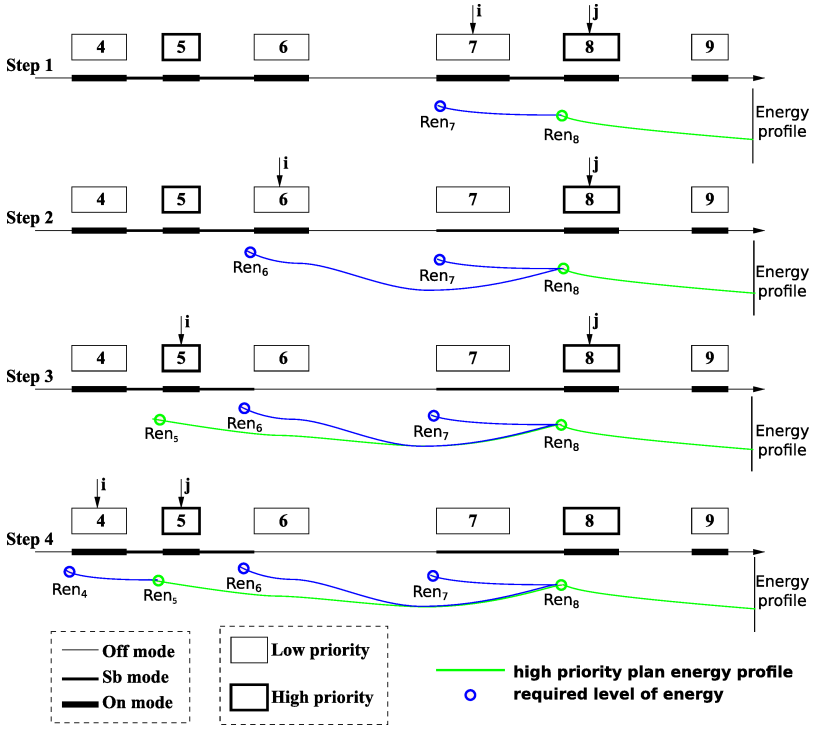


Figure 12.2 – Illustration de la procédure à rebours utilisée pour calculer les niveaux requis d'énergie.

correspondant est représenté par des segments larges (resp. fins) pour les modes pour ON (resp. SDBY et OFF). Par exemple, l'instrument est maintenu en mode SDBY entre Acq. 7 et Acq. 8, mais en mode OFF entre Acq. 8 et Acq. 9.

A l'étape 1, on suppose que le niveau requis d'énergie au début de Acq. 8 (haute priorité) a déjà été calculé et on calcule le niveau requis d'énergie au début de Acq. 7 (basse priorité), en supposant l'exécution de Acq. 7 et 8. Une erreur est retournée lorsque le calcul échoue. Ensuite, à l'étape 2, on calcule le niveau requis d'énergie au début de Acq. 6 (basse priorité), en supposant l'exécution de Acq. 6 et 8, mais pas de Acq. 7 puisqu'elle est de basse priorité et donc non obligatoire (le mode de l'instrument pendant Acq. 7 est mis à SDBY). De la même façon, à l'étape 3, on calcule le niveau requis d'énergie au début de Acq. 5 (haute priorité), en supposant l'exécution de Acq. 5 et 8, mas pas de Acq. 6 et 7 (le mode de l'instrument pendant Acq. 6 est mis à OFF). Parce que Acq. 5 est de haute priorité, son niveau requis d'énergie peut être utilisé comme un niveau de référence pour calculer les niveaux des observations précédentes dans la liste. C'est ce qui est fait à l'étape 4 dans laquelle on calcule le niveau d'énergie requis au début de Acq. 4 (basse priorité), en supposant l'exécution de Acq. 4, 5 et 8. En arrivant au début de l'horizon de planification, une erreur est retournée si le niveau requis d'énergie est plus grand que le niveau initial (ce

qui voudrait dire que le plan est infaisable). Sinon, l'algorithme termine correctement.

12.3 Exécution d'un plan d'observation conditionnel à bord

A bord, toutes les acquisitions de haute priorité sont exécutées. Avant chaque acquisition de basse priorité i, si le niveau d'énergie à bord et plus grand ou égal au niveau requis d'énergie Ren_i calculé au sol, l'acquisition i est réalisée. Sinon, l'instrument d'observation est mis dans le mode initialement prévu juste après l'observation i pour économiser de l'énergie (voir Figure 12.3 pour un exemple).

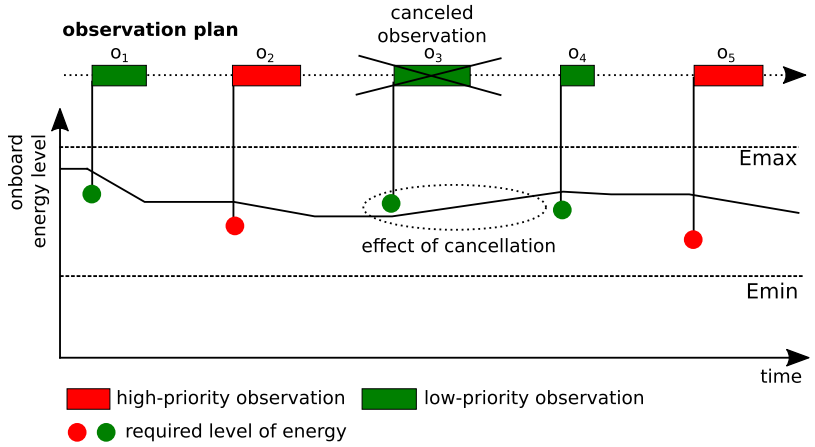


FIGURE 12.3 – Annulation d'une observation et son effet sur l'énergie à bord. Il y a un niveau requis d'énergie pour chaque observation o_1, o_2, o_3, o_4, o_5. Avant chaque observation de basse priorité comme o_3, si le niveau d'énergie courant est plus petit que le niveau requis calculé au sol, l'observation est annulée pour assurer la réalisation des toutes les futures acquisitions de haute priorité.

Comme dans l'approche flexible pour le vidage des données, on veut que ce mécanisme assure la réalisation des observations de haute priorité. Premièrement, si la procédure à rebours termine normalement, on est sûr que le sous-plan constitué seulement des observations de haute priorité est exécutable puisque les consommations d'énergie maximum et les productions d'énergie minimum ont été utilisées pour ce sous-ensemble d'observations. Si l'observation o_i à l'indice i est de haute priorité, il est sûr que toutes les observations de haute priorité situées après o_i peuvent être réalisées. Maintenant, on considère une observation de basse priorité o_{i-1} à l'index $i-1$. Le niveau requis d'énergie pour o_{i-1} a été calculé en partant du niveau requis pour o_i et en simulant à rebours depuis o_i jusqu'au début de o_{i-1}. Toutes les fenêtres de vidages ont été prises en compte. Alors si le niveau d'énergie à bord est plus grand que Ren_{i-1} au début de o_{i-1}, on est sûr qu'il est plus grand que Ren_i au début de o_i. Alors, toutes les observations de haute priorité futures peuvent être réalisées.

12.4 Expérimentations

Dans cette section, on évalue le mécanisme flexible de planification des observations que nous avons proposé.

12.4.1 Approches à comparer

Pour évaluer l'impact positif que peut avoir un partage de la prise de décision entre le sol et le bord pour la planification des observations, nous avons implémenté et comparé les trois approches suivantes dans un simulateur de mission satellite que nous avons développé et déjà utilisé pour des expérimentations dans le chapitre précédent :

1. Planification au sol avec vérification grossière des contraintes portant sur l'énergie (`GroundHiLevel`) : planification des observations au sol avec un algorithme qui vérifie les contraintes de haut niveau limitant la durée d'observation sur chaque orbite pour valider indirectement les contraintes sur l'énergie. Si le plan est invalide avec ces contraintes, les observations les moins importantes (par rapport au critère) sont supprimées du plan et le processus est relancé. A bord, le plan est exécuté sans aucun changement. Cette méthode est la plus proche de ce qui est fait actuellement en opérations.
2. Planification au sol avec vérification fine des contraintes portant sur l'énergie (`GroundFine`) : planification des observations au sol avec un algorithme qui simule l'exécution du plan étape par étape avec la consommation d'énergie maximum et la production d'énergie minimum pour toutes les observations (haute et basse priorité). Durant la simulation, quand le niveau d'énergie est en dessous de $Emin$, des observations sont supprimées du plan et le processus est relancé. A bord, le plan est exécuté tel quel.
3. Planification conditionnelle (`Conditional`) : planification des observations au sol avec un algorithmes qui calcule, pour chaque observation de basse priorité, le niveau d'énergie requis pour la réaliser elle ainsi que toutes les futures observations de haute priorité. Durant ce calcul, si le niveau requis d'énergie pour une observation o est plus grand que $Emax$, ce qui veut dire qu'un niveau d'énergie plus grand que le maximum est requis pour réaliser o et les futures observations de haute priorité, des observations sont supprimées du plan et le processus est relancé. A bord, les observations de basse priorité sont réalisées si et seulement si le niveau courant d'énergie est plus grand que le niveau requis calculé au sol.

12.4.2 Scenarios

Nous avons utilisé un scénario réaliste produit par Airbus Defence and Space qui couvre un jour. Il implique 5 utilisateurs, 3 stations sol de réception, 20 fenêtres de visibilité, et 1364 observations (247 de haute priorité et 1117 de basse priorité).

Incertitude L'attitude du satellite $att(t)$ à l'instant t est sujet à l'erreur $\epsilon(t) \in \mathbb{R}^2$ sur les axes de roulis et tangage. Soit $\overline{att(t)}$ la pire attitude possible à l'instant t par rapport à l'attitude optimale pour la recharge $hpAtt(t)$ dans laquelle les panneaux solaires sont pointés vers le soleil (voir Fig 12.4) :

$$\overline{att(t)} = \underset{\|a - att(t)\| \leq \epsilon(t)}{\arg\max} \; (angle(a, hpAtt(t))) \tag{12.1}$$

La production minimum d'énergie $\underline{\mathbf{Pp}_{solar}(t)}$ à l'instant t est donnée par l'équation suivante :

$$\underline{\mathbf{Pp}_{solar}(t)} = \max(0, P_{sun} \cdot \cos(angle(\overline{att(t)}, hpAtt(t)))) \tag{12.2}$$

Vu l'incertitude ϵ, $\underline{\mathbf{Pp}_{solar}(t)}$ est estimée proche de $0.75 \cdot \mathbf{Pp}_{solar}(t)$.

On considère aussi que la consommation d'énergie normale $\mathbf{Pc}(t)$ par la plateforme, l'instrument d'observation et l'antenne d'émission est incertaine. On évalue le même scénario avec trois hypothèses d'incertitude : consommation maximale à 10%, 20%, ou 30% au-dessus de la consommation nominale :

$$\overline{\mathbf{Pc}(t)} = (1 + \alpha) \cdot \mathbf{Pc}(t), \alpha \in \{.1, .2, .3\} \tag{12.3}$$

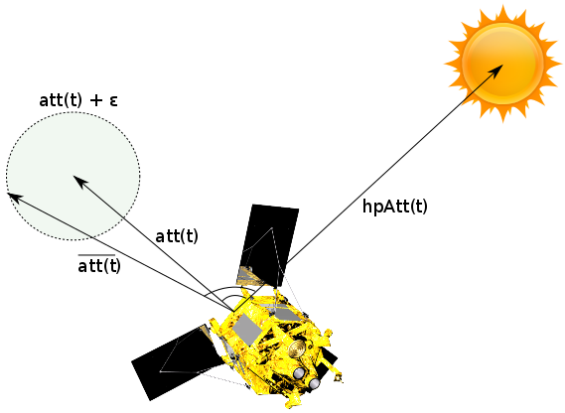


FIGURE 12.4 – Incertitude sur l'attitude et son influence sur la production d'énergie.

12.4.3 Résultats

Priorité	Incertitude sur la production (α)	GroundHiLevel	GroundFine	Conditional
Haute	10%	247	247	247
	20%	247	247	247
	30%	247	247	247
Basse	10%	976	1053	1086
	20%	938	1010	1061
	30%	898	927	1040

TABLE 12.2 – Nombre d'observations réellement réalisées à bord selon l'approche et l'incertitude.

Nous avons seulement mesuré le nombre d'observations réalisées. Dans le Tableau 12.2, on voit que quelque soit l'approche, toutes les observations de haute priorité sont réalisées. Concernant les observations de basse priorité, on peut voir que l'approche `Conditional` permet la réalisation du plus grand nombre d'acquisitions de

basse priorité. Sans surprise, l'approche `GroundFine` qui simule finement le profil en énergie est plus efficace et permet la réalisation de plus d'observations de basse priorité que l'approche `GroundHiLevel` qui valide les contraintes de haut niveau seulement. De plus, on peut voir que plus l'incertitude est grande sur la production et la consommation d'énergie, plus l'approche conditionnelle est intéressante.

La Figure 12.5 montre le niveau d'énergie à bord durant l'exécution d'un scénario de 24 heures à travers différentes perspectives. La méthode `GroundFine` calcule d'abord un profil en énergie simulé (pointillés sur la figure) en se basant sur les pires productions et consommations d'énergie. Une procédure supprime des observation jusqu'à ce que ce profil ne passe plus en-dessous de $Emin$, c'est à dire 85% de $Emax$ dans ce cas. On peut voir que le profil prédit et le profil réel produit par l'approche `Conditional` sont assez proches au début et à la fin du jour parce que les activités d'observation et de vidage sont réalisées principalement entre 6 heures et 18 heures. On peut aussi confirmer que le niveau d'énergie à bord ne tombe jamais en dessous du niveau minimum $Emin$ durant l'exécution du plan conditionnel.

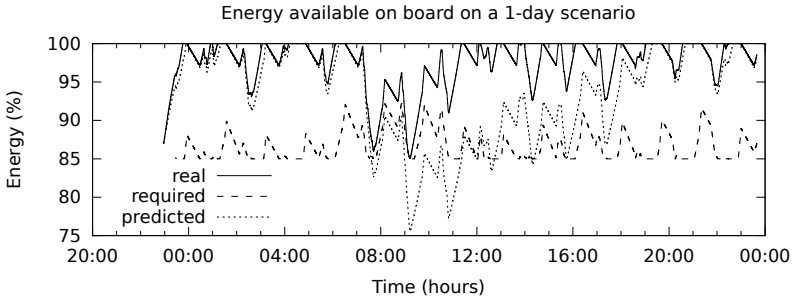


Figure 12.5 – Niveau d'énergie à bord au cours d'un scénario de 24 heures.

Cette évaluation a été réalisées en utilisant un seul scénario, il aurait été intéressant de montrer la robustesse de l'approche sur d'autres scénarios réalistes et avec d'autres paramètres physiques. Les expérimentations ont été faites sur un processeur Xeon 4-coeur 3GHz avec 8GB RAM, sur Ubuntu 12.04 LTS.

Perspectives 13

Dans ce manuscrit, nous avons tenté de montrer les propriétés et les avantages d'un mécanisme de prise de décision qui serait partagé entre le sol et le bord pour un satellite agile d'observation de la Terre. On décrit maintenant quelques perspectives.

Problème de planification du vidage des observations

- Des algorithmes d'approximation sont utilisées pour produire des plans de vidages, tels que l'algorithme de type SWO. Il serait intéressant de comparer la qualité des solutions trouvées avec les solutions optimales. Cela parait bien difficile de calculer ces solutions optimales dans le cas des scénarios réalistes. Il y a pourtant deux options qui ne sont pas parfaites mais assez satisfaisantes pour avoir une idée à ce propos. Une première option serait de calculer la distance entre les solutions optimales et approximées sur des instances plus petites mais suffisamment contraintes du problème. La seconde option impliquerait les scénarios réalistes mais les approximations seraient comparés avec les solutions optimales sur des relaxations des sous-problèmes (comme les sous-problèmes d'allocation ou d'ordonnancement) puisque ceux-ci sont beaucoup plus simples à résoudre séparément que lorsqu'ils sont tous intégrés.
- Nous avons utilisé un modèle d'incertitude très simplifié. Pour l'instant, nous avons supposé que les volumes réels des fichiers étaient distribués de manière uniforme sur l'intervalle d'incertitude. D'autres modèles pourraient être considérés. Un modèle météo pourrait permettre une planification plus informée aussi.
- Nous pourrions évaluer la planification flexible des vidages avec de nouvelles métriques, avec d'autres scénarios possédant une plus grande variabilité en terme de nombre de stations et de répartition des observations parmi les utilisateurs. Par exemple, un des points qui reste à aborder est la conception d'un critère permettant d'optimiser la flexibilité d'un plan de vidage au sol.
- Des algorithmes ont été proposés pour bâtir des plans de vidage complets. Au sol, un algorithme SWO bâti au-dessus d'un glouton non-chronologique est utilisé. D'autre métaheuristiques pourraient être utilisées, comme les algorithmes génétiques par exemple. Dans le même esprit, il serait intéressant d'évaluer l'influence des paramètres, comme la taille de l'horizon de planification. Sur ce point, il est habituellement positif d'avoir un grand horizon de planification dans le cas déterministe. Dans le cas d'un problème avec incertitude, il faut certainement équilibrer la taille de l'horizon de planification en fonction de la quantité d'incertitudes. En effet, s'il y a trop d'incertitudes sur le futur, les plans deviennent vite invalides à l'exécution.
- La prise de décision lors de la phase d'exécution du plan à bord n'a pas été encore explorée. Durant cette phase, de la flexibilité temporelle est fournie par le format du plan. Une règle réactive pourrait être conçue pour prendre des décisions pour les observations se terminant pendant cette phase. Par exemple, si une de ces observations possède un volume plus faible que prévu (plus faible que maximum), et que l'exécution attend la fin d'une autre observation, une règle de décision peut être capable d'insérer de nouveaux vidages dans le plan pour profiter de ces espaces inutilisés.

Problème de planification des observations

- Un nombre plus important d'expérimentations devraient être réalisées sur une plus large gamme de scénarios pour démontrer la viabilité de l'approche proposée.
- Il serait désirable de relâcher l'hypothèse que toutes les fenêtres de vidages sont utilisées pour le vidage. Tandis que cette hypothèse est proche de la réalité dans les scénarios sur-contraints, dans lesquels il y a peu de stations sol et donc peu de fenêtres de vidage, l'hypothèse est bien loin de la réalité dans les scénarios civils dans lesquels il y a de nombreuses stations couvrant tous les continents et dans lesquels les fenêtres de vidages sont sous-utilisées. Cette hypothèse rend la planification des vidages très pessimiste puisque il est supposé que l'antenne d'émission est pratiquement toujours allumée. Une meilleure option consisterait à calculer au sol le niveau requis pour utiliser un groupe de fenêtres de vidage. A bord, avant chacun de ces groupes, si le niveau d'énergie à nord est suffisant, alors le groupe peut être utilisé librement pour vider des données. Sinon, seules les observations de hautes priorité seraient vidées, cela pour économiser de l'énergie.
- Des recherches futures pourraient consister à adapter le même mécanisme de planification conditionnelle pour traiter l'incertitude sur la température des instruments. Cela permettrait de relâcher les contraintes de haut niveau utilisées lors de la planification, contraintes qui conduisent à rejeter 9% des observations.
- Finalement, une extension importante consisterait à calculer les profils d'observation à bord. Cela permettrait de rendre le satellite plus autonome et de concevoir un mécanisme de prise de décision encore plus flexible, tel que celui pour la planification des vidages. Bien qu'il ne soit pas possible de calculer ces profils à bord pour l'instant, il serait faisable de calculer des plans avec des profils d'attitude conditionnels (un profil pour chaque branche du plan).

Plateforme expérimentale

Une perspective intéressante, que ce soit pour le problème de planification des vidages ou de planification des observations, serait de combler le fossé entre la théorie et la pratique en menant des expérimentations sur des ordinateurs de bord. Implémenter ces approches représenteraient certainement un défi puisqu'ils impliquent tout le système sol et bord. De plus, certaines contraintes relatives aux algorithmes bord devraient être considérées, comme par exemple le temps de calcul et les performances.

Un cadre de prise de décision générique

Nous avons vu que la prise de décision *flexible* peut être utilisée pour des problèmes impliquant de larges horizons de planification et des incertitudes sur beaucoup de tâches. Quand certaines tâches sont obligatoires, comme les observations de haute priorité ou leurs vidages, il est possible de garantir leurs réalisations s'il est possible de récupérer des estimations pire-cas de leurs consommations en ressources (volumes de fichiers maximums, productions et consommations d'énergie maximums...). Plusieurs cadres existent déjà pour la décision dans l'incertain mais la mise à l'échelle se révèle souvent difficile (typiquement le cadre *stochastic programming*). Dans ce manuscrit, on a montré qu'il est possible de prendre en compte des problèmes complexes de taille importante. Généraliser le cadre qui a été développé pour la planification des activités de satellites d'observation de la Terre permettrait de traiter d'autres problèmes

de la même façon. *Online stochastic combinatorial optimization* propose un peu l'idée similaire qui consiste à unifier deux phases de prise de décision : une phase de décision rapide en-ligne avec des méthodes habituellement utilisées hors-ligne comme l'échantillonnage. Dans notre approche, le but est de spécialiser l'effort calculatoire en fonction de la phase de décision et de faciliter l'adaptation en-ligne des plans calculés hors-ligne. Finalement, nous nous concentrons plus sur les contraintes relatives aux ressources et à l'idée de garantir la réalisation de certaines activités obligatoires.

Bibliography

P. Aguiar Melgarejo, P. Laborie, and C. Solnon. A Time-Dependent No-Overlap Constraint: Application to Urban Delivery Problems. In *12th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CP-AI-OR)*, LNCS, pages 1–17. Springer, May 2015.

M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, et al. Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission. *Intelligent Systems, IEEE*, 19 (1):8–12, 2004.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.

P. Awasthi and T. Sandholm. Online stochastic optimization in the large: Application to kidney exchange. In *Proc. of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA*, pages 405–411, 2009.

L. Barbulescu, A. E. Howe, M. Roberts, and L. D. Whitley. Understanding Algorithm Performance on an Oversubscribed Scheduling Application. *Journal of Artificial Intelligence Research*, 27:577–615, 2006.

R. Barták and M. A. Salido. Constraint satisfaction for planning and scheduling problems. *Constraints*, 16(3):223–227, 2011.

M. Bartusch, R. H. Möhring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of operations Research*, 16(1): 199–240, 1988.

G. Beaumet, G. Verfaillie, and M. Charmeau. 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)*, pages 3–17, Providence, RI, USA, 2007.

G. Beaumet, G. Verfaillie, and M. Charmeau. Feasibility of Autonomous Decision Making on board an Agile Earth-observing Satellite. *Computational Intelligence*, 27(1):123–139, 2011.

G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In *Proc. of the 19th International Conference on Computer Aided Verification (CAV)*, pages 121–125, Berlin, Germany, 2007.

T. Benoist, B. Estellon, F. Gardi, R. Megel, and K. Nouioua. Localsolver 1.x: a black-box local-search solver for 0-1 programming. *4OR*, 9(3):299–316, 2011.

D. Bernard, G. A. Dorais, C. Fry, E. B. Gamble, B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. Nayak, B. Pell, K. Rajan, N. Rouquette, B. Smith, and B. C. Williams. Design of the remote agent experiment for spacecraft autonomy. In *Aerospace Conference*, volume 2, pages 259–281. IEEE, 1998.

P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Mbp: a model based planner. In *Proc. of the IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information*, 2001.

J.-C. Billaut, A. Moukrim, and E. Sanlaville. *Flexibility and robustness in scheduling*. John Wiley & Sons, 2013.

R. Bjarnason, A. Fern, and P. Tadepalli. Lower bounding klondike solitaire with monte-carlo planning. In *Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09)*, 2009.

A. Bonfietti, M. Lombardi, and M. Milano. Disregarding Duration Uncertainty in Partial Order Schedules? Yes, We Can! In *Proc. of the International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR-14)*, pages 210–225, Cork, Ireland, 2014.

J. Bresina, R. Dearden, N. Meuleau, S. Ramkrishnan, D. Smith, and R. Washington. Planning under Continuous Time and Resource Uncertainty: A Challenge for AI. In *Proc. of the 18th International Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 77–84, Edmonton, Alberta, Canada, 2002a.

J. Bresina. Heuristic-biased stochastic sampling. In *Proc. of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI, IAAI, Portland, Oregon*, pages 271–278, 1996.

J. L. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. E. Smith, and R. Washington. Planning under continuous time and resource uncertainty: A challenge for AI. In *Proc. of the 18th Conference in Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada*, pages 77–84, 2002b.

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):1–43, 2012.

R. Castano, T. Estlin, R. Anderson, D. Gaines, A. Castano, B. Bornstein, C. Chouinard, and M. Judd. OASIS: Onboard Autonomous Science Investigation System for Opportunistic Rover Science. *Journal of Field Robotics*, 24(5): 379–397, 2007.

A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, and N. Policella. An Innovative Product for Space Mission Planning: An A Posteriori Evaluation. In *Proc. of the 17th International Conference on Automated Planning and Scheduling (ICAPS-07)*, Providence, RI, USA, 2007.

G. Chabert and L. Jaulin. Contractor programming. *Artificial Intelligence*, 173(11): 1079–1100, 2009.

G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-carlo tree search: A new framework for game AI. In *Proc. of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, October 22-24, 2008, Stanford, California, USA*, 2008.

Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaitre, N. Maudet, J. Padget, S. Phelps, J. A. Rodriguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica (Slovenia)*, 30(1):3–31, 2006.

S. Chien, R. Knight, A. Stechert, R.Sherwood, and G. Rabideau. 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)*, pages 300–307, Breckenridge, CO, USA, 2000.

S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, R. Lee, D. Mandl, S. Frye, B. Trout, J. Hengemihle, J. D'Agostino, S. Shulman, S. Ungar, T. Brakke, D. Boyer, J. VanGaasbeck, R. Greeley, T. Doggett, V. Baker, J. Dohm, and F. Ip. The EO-1 Autonomous Science Agent. In *Proc. of the 3rd Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*, pages 420–427, New York City, USA, 2004a.

S. Chien and M. Troesch. Heuristic onboard pointing re-scheduling for an earth observing spacecraft. *Proc of the ICAPS'15 Scheduling and Planning Applications Workshop (SPARK)*, 2015.

S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, R. Lee, D. Mandl, S. Frye, et al. The eo-1 autonomous science agent. In *Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 420–427. IEEE Computer Society, 2004b.

J.-P. Chrétien, M. Llibre, and F. Jouhaud. Agilité en attitude par actionneurs gyroscopiques: génération de consignes et commande boucle fermée. In *Atelier CNES "Commande optimale, Commande boucle fermée"*, Toulouse, France, 2004.

B. Cichy, S. Chien, S. Schaffer, D. Tran, G. Rabideau, and R. Sherwood. Validating the autonomous eo-1 science agent. *WS5*, page 39, 2005.

F. de Novaes Kucinskis, M. G. V. Ferreira, and R. Arias. Increasing the autonomy of scientific satellites to deal with short-duration phenomena. In *Aerospace Conference, 2007 IEEE*, pages 1–12. IEEE, 2007.

R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. Incremental contingency planning. In *ICAPS'03 Workshop on Planning under Uncertainty and Incomplete Information*, pages 38–47, 2003.

R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49:61–95, 1991.

M. Drummond, J. L. Bresina, and K. Swanson. Just-in-case scheduling. In *Proc. of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 1098–1104, Seattle, WA, USA, 1994.

J. Eker, J. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Sachs, Y. Xiong, and S. Neuendorffer. Taming heterogeneity - the ptolemy approach. *Proc. of the IEEE*, 91(1):127–144, 2003.

H. Fargier and J. Lang. Uncertainty in Constraint Satisfaction Problems: A Probabilistic Approach. In *Proc. of the European Conference on Symbolic and Quantitavive Approaches of Reasoning under Uncertainty (ECSQARU-93)*, pages 97–104, Grenade, Spain, 1993.

H. Fargier, J. Lang, R. Martin-Clouaire, and T. Schiex. Mixed Constraint Satisfaction : a Framework for Decision Problems under Uncertainty. In *Proc. of the 11th International Conference on Uncertainty in Artificial Intelligence (UAI-95)*, Montréal, Canada, 1995.

G. Fohler. Realizing changes of operational modes with a pre run-time scheduled hard real-time system. In *Responsive Computer Systems*, pages 287–300. Springer, 1993.

M. Fox and D. Long. Single-trajectory opportunistic planning under uncertainty. In *Proc. of the UK Planning and Scheduling SIG meeting*, Delft, The Netherlands, 2002.

M. Fox, A. Gerevini, D. Long, and I. Serina. Plan stability: Replanning versus plan repair. *Proc. of the 16th International Conference on Automated Planning and Scheduling (ICAPS-06)*, 2006a.

M. Fox, R. Howey, and D. Long. Exploration of the robustness of plans. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 834–839, 2006b.

J. Frank, A. Jonsson, R. Morris, and D. Smith. Planning and scheduling for fleets of earth observing satellites. In *Proc. of the 6th international symposium on artificial intelligence, robotics, automation and space*, 2001.

N. Fu, H. C. Lau, P. Varakantham, and F. Xiao. Robust local search for solving rcpsp/max with durational uncertainty. *Journal of Artificial Intelligence Research*, 43 (1):43–86, 2012.

R. Gerber, W. Pugh, and M. Saksena. Parametric dispatching of hard real-time tasks. *Computers, IEEE Transactions on*, 44(3):471 –479, March 1995.

M. L. Ginsberg, A. J. Parkes, and A. Roy. Supermodels and robustness. In *Proc. of the 15th National Conference on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA.*, pages 334–339, 1998.

A. Globus, J. Crawford, J. Lohn, and R. Morris. A Comparison of Techniques for Scheduling Earth Observing Satellites. In *Proc. of the 16th Conference on Innovative Applications of Artificial Intelligence (IAAI-04)*, San Jose, CA, USA, 2004a.

A. Globus, J. Crawford, J. D. Lohn, and A. Pryor. A comparison of techniques for scheduling earth observing satellites. In *Proc. of the 19th National Conference on Artificial Intelligence, 16th Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 836–843, 2004b.

J. Gough, M. Fox, and D. Long. Plan execution under resource consumption uncertainty. In *Proc. of the ICAPS Workshop on Connecting Planning Theory with Practice*, Whistler, British Columbia, Canada, 2004.

L. Granvilliers and F. Benhamou. Algorithm 852: Realpaver: an interval solver using constraint satisfaction techniques. *ACM Trans. Math. Softw.*, 32(1):138–156, 2006.

R. Grasset-Bourdel, G. Verfaillie, and A. Flipo. Action and Motion Planning for Agile Earth-observing Satellites. *Acta Futura*, 5:121–131, 2012.

L. Greenwald and T. Dean. A conditional scheduling approach to designing real-time systems. In *Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems, Pittsburgh, Pennsylvania, USA*, pages 224–231, 1998.

E. Hebrard, M.-J. Huguet, N. Jozefowiez, A. Maillard, C. Pralet, and G. Verfaillie. Approximation of the parallel machine scheduling problem with additional unit resources. *Discrete Applied Mathematics*, Submitted, 2015.

P. V. Hentenryck and L. Michel. *Constraint-based Local Search*. MIT Press, 2005.

P. V. Hentenryck and R. Bent. *Online stochastic combinatorial optimization*. MIT Press, 2009.

J. Hoffmann and R. I. Brafman. Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence*, 170(6-7):507–541, 2006.

L. Hunsberger. A faster algorithm for checking the dynamic controllability of simple temporal networks with uncertainty. In *Proc. of the 6th International Conference on Agents and Artificial Intelligence, Volume 1, ESEO, Angers, Loire Valley, France, 6-8 March, 2014*, pages 63–73, 2014.

N. Immorlica, D. Karger, M. Minkoff, and V. S. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *Proc. of the 15th annual ACM-SIAM symposium on Discrete algorithms*, pages 691–700. Society for Industrial and Applied Mathematics, 2004.

M. T. Jensen. Generating robust and flexible job shop schedules using genetic algorithms. *IEEE Trans. Evolutionary Computation*, 7(3):275–288, 2003.

D. Joslin and D. P. Clements. Squeaky wheel optimization. *Journal of Artificial Intelligence Research (JAIR)*, 10:353–373, 1999.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

S. Kiesel and W. Ruml. Planning under temporal uncertainty using hindsight optimization. In *Proc. of the ICAPS'14 Workshop on Planning and Robotics*, Portsmouth, NH, USA, 2014.

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

S. Lemai and F. Ingrand. Interleaving Temporal Planning and Execution in Robotics Domains. In *Proc. of the 19th National Conference on Artificial Intelligence (AAAI-04)*, San Jose, CA, USA, 2004.

M. Lemaître, G. Verfaillie, H. Fargier, J. Lang, N. Bataille, and J. Lachiver. Equitable allocation of Earth observing satellites resources. In *Proc. of the 5th ONERA-DLR Aerospace Symposium*, 2003.

M. Lemaître, G. Verfaillie, and N. Bataille. Exploiting a common property resource under a fairness constraint: A case study. In *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 206–211, Stockholm, Sweden, 1999.

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

M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6 (5):367 – 381, 2002.

M. L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research (JAIR)*, 9:1–36, 1998.

I. Mahdavi, B. Shirazi, and M. Solimanpur. Development of a simulation-based decision support system for controlling stochastic flexible job shop manufacturing systems. *Simulation Modelling Practice and Theory*, 18(6):768–786, 2010.

A. Maillard, C. Pralet, G. Verfaillie, J. Jaubert, and T. Desmousceaux. Building Flexible Download Plans for Agile Earth-Observing Satellites. In *Proc. of International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS-14)*, Montreal, Canada, 2014.

A. Maillard, C. Pralet, J. Jaubert, I. Sebbag, F. Fontanari, and J. L'Hermitte. Ground and Board Decision-Making on Data Downloads. In *Proc. of the 25th International Conference on Automated Planning and Scheduling (ICAPS-15)*, Jerusalem, Israël, 2015a.

A. Maillard, C. Pralet, G. Verfaillie, J. Jaubert, I. Sebbag, and F. Fontanari. Postponing Decision-Making to Deal with Resource Uncertainty on Earth-Observation Satellites. In *Proc. of 9th International Workshop on Planning and Scheduling for Space*, Buenos Aires, Argentina, 2015b.

L. Michel and P. Van Hentenryck. The comet programming language and system. In *Principles and Practice of Constraint Programming-CP 2005*, pages 881–881. Springer, 2005.

A. Oddi and N. Policella. A Max-Flow Approach for Improving Robustness in a Spacecraft Downlink Schedule. In *Proc. of the 4th International Workshop on Planning and Scheduling for Space (IWPSS-04)*, Darmstadt, Germany, 2004.

A. Oddi, N. Policella, A. Cesta, and G. Cortellesa. Generating High Quality Schedules for a Spacecraft Memory Downlink Problem. In *Proc. of the 9th International Conference on Principles and Practice of Constraint Programming (CP-03)*, pages 570–584, Cork, Ireland, 2003.

A. Orlandini, A. Finzi, A. Cesta, and S. Fratini. Tga-based controllers for flexible plan execution. In J. Bach and S. Edelkamp, editors, *KI 2011: Advances in Artificial Intelligence*, volume 7006 of *Lecture Notes in Computer Science*, pages 233–245. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-24454-4.

H. Palacios and H. Geffner. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research*, 35(2):623, 2009.

J. Pemberton and L. Greenwald. On the need for dynamic scheduling of imaging satellites. *International Archives Of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(1):165–171, 2002.

M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 2012.

N. Policella, S. Smith, A. Cesta, and A. Oddi. Generating Robust Schedules through Temporal Flexibility. In *Proc. of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 209–218, Whistler, Canada, 2004a.

N. Policella, S. F. Smith, A. Cesta, and A. Oddi. Generating robust schedules through temporal flexibility. In *Proc. of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada*, pages 209–218, 2004b.

N. Policella, A. Cesta, A. Oddi, and S. F. Smith. Solve-and-robustify. *J. Scheduling*, 12(3):299–314, 2009.

C. Pralet and G. Verfaillie. Dynamic Online Planning and Scheduling using a Static Invariant-based Evaluation Model. In *Proc. of the 23rd International Conference on Automated Planning and Scheduling (ICAPS-13)*, Rome, Italy, 2013a.

C. Pralet and G. Verfaillie. Time-dependent Simple Temporal Networks: Properties and Algorithms. *RAIRO Operations Research*, 47(2):173–198, 2013b.

C. Pralet and G. Verfaillie. Time-Dependent Simple Temporal Networks. In *Proc. of the 18th International Conference on Principles and Practice of Constraint Programming (CP-12)*, pages 322–338, Quebec City, Canada, 2012.

C. Pralet, A. Maillard, G. Verfaillie, E. Hébrard, N. Jozefowiez, M.-J. Huguet, T. Desmousceaux, P. Blanc-Paques, and J. Jaubert. Gestion du vidage de données satellite avec incertitude sur les volumes. In *10èmes Journées Francophones de Programmation par Contraintes - JFPC 2014*, Angers, France, 2014a.

C. Pralet, G. Verfaillie, E. Hebrard, M.-J. Huguet, and N. Jozefowiez. Planification mission flexible. Technical report 2/21065 DCSD, ONERA, 2014b.

C. Pralet, G. Verfaillie, A. Maillard, E. Hebrard, N. Jozefowiez, M.-J. Huguet, T. Desmousceaux, P. Blanc-Paques, and J. Jaubert. Satellite Data Download Management with Uncertainty about the Generated Volumes. In *Proc. of the 24th International Conference on Automated Planning and Scheduling (ICAPS-14)*, Portsmouth, NH, USA, 2014c.

C. Pralet and G. Verfaillie. 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)*, 2008.

C. Pralet and G. Verfaillie. Dynamic online planning and scheduling using a static invariant-based evaluation model. In *Proc. of the Twenty-Third International Conference on Automated Planning and Scheduling, ICAPS 2013, Rome, Italy*, 2013c.

C. Pralet, G. Infantes, and G. Verfaillie. A generic constraint-based local search library for the management of an electromagnetic surveillance space mission. In *Proc. of ICAPS'13 Scheduling and Planning Applications Workshop (SPARK)*, 2013.

M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

G. Rabideau, R. Knight, S. Chien, A. Fukunaga, and A. Govindjee. Iterative repair planning for spacecraft operations using the aspen system. In *Proc. of i-SAIRAS'99*, volume 440, page 99, 1999.

R. Rasconi, A. Cesta, and N. Policella. Validating scheduling approaches against executional uncertainty. *Journal of Intelligent Manufacturing*, 21(1):49–64, 2010.

G. Righini and E. Tresoldi. A Mathematical Programming Solution to the Mars Express Memory Dumping Problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 40(3):268–277, 2010.

F. Rossi, P. Van Beek, and T. Walsh. *Handbook of constraint programming*. Elsevier, 2006.

H. E. Sakkout and M. Wallace. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints*, 5(4):359–388, 2000.

S. R. Schaffer, B. J. Clement, and S. A. Chien. Probabilistic reasoning for plan robustness. In *IJCAI-05, Proc. of the 19th International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK*, pages 1266–1271, 2005.

M. Sevaux and K. Sörensen. A genetic algorithm for robust schedules in a just-in-time environment. Technical report, 2002.

S. F. Smith. Reactive scheduling systems. In *Intelligent scheduling systems*, pages 155–192. Springer, 1995.

Y. N. Sotskov, V. S. Tanaev, and F. Werner. Stability radius of an optimal schedule: A survey and recent developments. 1998.

K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120(1):81–117, 2000.

F. Stork. *Stochastic resource-constrained project scheduling*. PhD thesis, Universitätsbibliothek, 2001.

F. Teston, R. Creasey, J. Bermyn, D. Bemaerts, and K. Mellab. Proba: Esa's autonomy and technology demonstration mission. Technical report, 1999.

D. Thompson, T. Smith, and D. Wettergreen. Information-Optimal Selective Data Return for Autonomous Rover Traverse Science and Survey. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA-08)*, pages 968–973, 2008.

G. Verfaillie, G. Infantes, M. Lemaître, N. Théret, and T. Natolot. On-board Decision-making on Data Downloads. In *Proc. of the 7th International Workshop on Planning and Scheduling for Space (IWPSS-11)*, Darmstadt, Germany, 2011.

G. Verfaillie and M. Lemaître. Selecting and scheduling observations for agile satellites: some lessons from the constraint reasoning community point of view. In *Proc. of Principles and Practice of Constraint Programming (CP'01)*, pages 670–684. Springer, 2001.

T. Vidal and M. Ghallab. Dealing with uncertain durations in temporal constraint networks dedicated to planning. In *Proc of the 12th European Conference on Artificial Intelligence, Budapest, Hungary*, pages 48–54, 1996.

T. Walsh. Stochastic Constraint Programming. In *Proc. of the 15th European Conference on Artificial Intelligence (ECAI-02)*, pages 111–115, Lyon, France, 2002.

M. Wilson, T. Klos, C. Witteveen, and B. Huisman. Flexibility and decoupling in simple temporal networks. *Artificial Intelligence*, 214:26–44, 2014.

W. J. Wolfe and S. E. Sorensen. Three scheduling algorithms applied to the earth observing systems domain. *Management Science*, 46(1):148–166, 2000.

M. Woods, A. Shaw, D. Barnes, D. Price, D. Long, and D. Pullan. Autonomous Science for an ExoMars Rover-Like Mission. *Journal of Field Robotics*, 26(4): 358–390, 2009a.

M. Woods, A. Shaw, D. Barnes, D. Price, D. Long, and D. Pullan. Autonomous science for an exomars rover-like mission. *Journal of Field Robotics*, 26(4):358–390, 2009b.

G. Wu, E. K. Chong, and R. Givan. Burst-level congestion control using hindsight optimization. *Automatic Control, IEEE Transactions on*, 47(6):979–991, 2002.

S. D. Wu, E.-S. Byeon, and R. H. Storer. A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47(1):113–124, 1999.

S. W. Yoon, A. Fern, R. Givan, and S. Kambhampati. Probabilistic planning via determinization in hindsight. In *Proc. of the Twenty-Third Conference on Artificial Intelligence, AAAI, Chicago, Illinois, USA,*, pages 1010–1016, 2008.

List of Figures

List of Tables

Earth-observation satellites are space sensors which acquire data, compress and record it on board, and then download it to the ground. They evolve in a dynamic environment which implies several uncertain parameters such as cloud cover or available energy on board. These uncertainties make planning and scheduling satellite activities offline on the ground more and more arguable. Until now, plans are often built on the ground and are not to be modified on board. It means that in face of uncertainties, current plans need to be robust. It implies that on one hand, all the decisions are made offline on the ground and the satellite is a simple executive which neither makes, nor changes any decision and on the other hand, worst-case assumptions are made about uncertain parameters. This makes planning very pessimistic and plans suboptimal. This dissertation details our efforts at designing a flexible decision-making scheme that allows to profit from the realization of uncertain parameters on board while keeping a fair level of predictability on the ground.

Our first contribution concerns the data download problem. Sophisticated onboard algorithms compress low-interest zones of acquired images such as cloudy zones. The amount of data resulting from an acquisition is then unpredictable on the ground. Until now, all decisions regarding downloads are made on the ground, and data volumes are assumed to be maximum to make the plan robust. This makes plans highly suboptimal because real volumes are almost always lower than maximum and download windows are then underused. A flexible decision-making mechanism has been designed where only high-priority acquisitions are scheduled with worst-case assumptions. Other acquisition downloads are scheduled with expected volumes and conditioned by resource availability. The plan is then adapted on board: if acquisitions have a lower volume than predicted, acquisition downloads are added to the plan. If a low-priority acquisition has a higher volume than predicted, a look-ahead is performed to prevent future high-priority downloads to be endangered. This decision-making mechanism has been evaluated against pure ground and pure onboard approaches and showed great properties, ensuring predictability for the high-priority acquisitions and efficiency for the low-priority acquisitions while being computationally light on board. Our second contribution concerns the acquisition planning problem. When an acquisition plan is computed on the ground, resource availability such as memory or energy must be checked to ensure that the execution of the plan will not in endanger the satellite. From a technical point of view, because simulating the evolution of onboard energy and temperature is computationally expensive, it cannot be done finely during the planning process. It is replaced by high-level constraints with safety margins. The negative side of such a strategy is that the real energy profile is always better than predicted, and a lot of acquisitions that could have been done are eliminated when planning because of these high-level constraints. In a new decision-making scheme, these high-level constraints are removed for low-priority acquisitions. Observation plans produced on the ground are conditional plans involving conditions for triggering low-priority acquisitions. Compared with the current approach, this approach avoids wastage of resource and allows more acquisitions to be executed.

Les satellites d'observation de la Terre sont des senseurs qui acquièrent des données, les compressent et les mémorisent à bord, puis les vident vers le sol. Ils évoluent dans un environnement dynamique ce qui implique que certains paramètres comme la couverture nuageuse ou l'énergie disponible à bord sont incertains. Ces incertitudes rendent la planification des activités au sol de plus en plus discutable. Jusqu'à maintenant, les plans d'activité sont produits au sol et exécutés strictement à bord. Les paramètres incertains sont réglés aux pires valeurs possibles pour que les plans soient robustes aux incertitudes. Les plans sont alors largement sous-optimaux. Cette thèse détaille la conception d'une plan-ification mixte qui permet de profiter de la réalisation des paramètres incertains à bord tout en préservant la prévisibilité de l'exécution pour les opérateurs au sol. Notre première contribution concerne le problème de planification des vidages. A bord, des algorithmes compressent les zones de faible intérêt, telles que les zones nuageuses, après acquisition des images. La quantité de données résultant d'une acquisition est donc imprévisible au sol. Jusqu'à maintenant, la planifi-cation des vidages est réalisée au sol avec des volumes maximums pour permettre l'exécution robuste du plan. Les plans sont alors largement sous-optimaux puisque les volumes réels sont presque tout le temps plus faibles. Un mécanisme de planification flexible a été conçu dans lequel seules les acquisitions de haute priorité sont planifiées avec les volumes maximums. Les autres sont planifiées avec des volumes espérés. A bord, un algorithme adapte le plan en fonction des volumes réels, en s'assurant que le vidage des acquisitions de haute priorité est toujours garanti, et insère des nouveaux vidages si possible. Comparé avec des mécanismes de planification purement bord ou sol, ce mécanisme flexible permet d'exécuter des plans plus efficaces, prévisibles pour les acquisitions de haute priorité, tout en consommant un minimum de ressources informatiques à bord du satellite. Notre deuxième contribution concerne le problème de planification des acquisitions. Quand un plan d'acquisition est calculé au sol, il faut vérifier que l'exécution du plan ne dépasse pas la capacité des ressources, telles que la mémoire et l'énergie. Puisque simuler finement l'évolution de la température ou de l'énergie sur un horizon temporel important n'est pas faisable, des contraintes de haut niveau avec des marges sur l'utilisation des ressources sont en pratique ajoutées au problème. Au sol, ces contraintes contribuent à éliminer du plan de nombreuses acquisitions qui auraient pu être réalisées car les niveaux d'énergie à bord sont souvent plus hauts que ceux prévus par ces contraintes. Dans un nouveau mécanisme de décision, ces contraintes sont supprimées pour les acquisitions de basse priorité. Le sol produit des plans conditionnels dans lesquels la réalisation des acquisitions de basse priorité est conditionnée par des niveaux d'énergie requis. Comparé à d'autres mécanismes, cette approche permet d'éviter le gaspillage d'énergie et permet de réaliser plus d'acquisitions.