ELSEVIER

Aerospace Science and Technology 6 (2002) 367–381

Aerospace Science and Technology

www.elsevier.com/locate/aescte

Selecting and scheduling observations of agile satellites

Michel Lemaître [a,*], Gérard Verfaillie [a], Frank Jouhaud [a], Jean-Michel Lachiver [b], Nicolas Bataille [b]

[a] *ONERA, Department of Systems Control and Flight Dynamics, 2, avenue Édouard Belin, B.P. 4025, 31055 Toulouse Cedex 4, France*
[b] *CNES, Advanced Observation Systems Department, 18, avenue Édouard Belin, 31401 Toulouse Cedex 4, France*

Received 21 March 2001; accepted 1 July 2002

Abstract

This article concerns the problem of managing the new generation of Agile Earth Observing Satellites (AEOS). This kind of satellites is presently studied by the French *Centre National d'Études Spatiales* (PLEIADES project). The mission of an Earth Observing Satellite is to acquire images of specified areas on the Earth surface, in response to observation requests from customers. Whereas non-agile satellites such as SPOT5 have only one degree of freedom for acquiring images, the new generation satellites have three, giving opportunities for a more efficient use of the satellite imaging capabilities. Counterwise to this advantage, the selection and scheduling of observations becomes significantly more difficult, due to the larger search space for potential solutions. Hence, selecting and scheduling observations of agile satellites is a highly combinatorial problem. This article sets out the overall problem and analyses its difficulties. Then it presents different methods which have been investigated in order to solve a simplified version of the complete problem: a greedy algorithm, a dynamic programming algorithm, a constraint programming approach and a local search method.

Keywords: Earth observing satellite; Agile satellite; Mission management; Scheduling

1. Introduction

The mission of an Earth Observing Satellite (EOS) is to acquire images of specified areas on the Earth surface, in response to observation requests from customers.

The current generation of EOS, like those belonging to the SPOT system of satellites,[1] have only one degree of freedom, provided by a mobile mirror in front of each observation instrument. The mirror, operated on the roll axis, is only moved during transitions between images, and remains fixed during an image acquisition. An image is produced by the movement of the satellite on its track. There is only one possible azimuth for imaging: the one parallel to the track of the satellite. The starting time of any candidate image is fixed: it is the exact time at which the satellite fligths over the beginning of the area to be acquired. Consequently, the order between any candidate image cannot change, and compatibilities between images can be pre-computed. The problem to be solved is only a selection problem, and not a true scheduling problem.

The new generation of EOS, like those studied in the French PLEIADES project, are *Agile* Earth Observing Satellites (AEOS). This means that, while the unique on-board instrument is not mobile and remains fixed on the satellite, the whole satellite is mobile on the three axes (roll, pitch and yaw), thanks to its Attitude and Orbit Control System, thus allowing manoeuvrability for image acquisitions as well as for transitions between images. The advantages provided by these new agility capabilities are the following: there is a potentially infinite number of ways of acquiring a given area on the Earth surface, since the azimuth and the starting time of the image acquisition are now free (within given limits). These two additional degrees of freedom (starting times and azimuths of imaging) give rise to a potentially better efficiency of the whole system. On the other hand, the selection and scheduling of observations for an AEOS becomes significantly more difficult, because the search space is considerably larger.

This article reports a study on the mission management of these AEOS. The Section 2 states the overall problem. After a reminder on heliosynchronous satellites, and agile

* Corresponding author.
E-mail address: Michel.Lemaitre@cert.fr (M. Lemaître).

[1] See the URL http://www.cnes.fr/WEB_UK/activites/programmes/1index.html.

1270-9638/02/$ – see front matter
PII: S1270-9638(02)01173-2

satellites, the general selection and scheduling problem is loosely described, and its difficulties are pointed out. Then we show how the overall problem was simplified and cast into a simpler problem named the AEOS Track Selection and Scheduling Problem, in order to be actually solved.

Even this simplified problem is highly combinatorial. The Section 3 presents four different approaches to the resolution of the Track Selection and Scheduling Problem, and then compares the results obtained with these approaches.

Finally, we will state our conclusions and present directions for future work.

2. The problem

2.1. Managing observations of an AEOS: the full problem

2.1.1. General operation of an Earth Observing Satellite

An Earth Observing Satellite is operated by an Image Programming and Processing Center, which collects observation requests from customers and distributors, builds daily the imaging workload of the satellite, receives back the data associated with acquired images, processes and evaluates them, and finally sends back the results to the customers (see Fig. 1).

A request generally involves several contiguous images (this will be explained later). Due to the large number of requests concerning some zones, in general all of these requests cannot be satisfied on one day. Choices have to be made between observations covering the requests. Accordingly, the overall problem consists of selecting, for each period of time (typically a day), a feasible sequence of images that will be acquired by the satellite over the period (see Fig. 3), with the objective of providing maximum satisfaction to the customers.

2.1.2. A heliosynchronous, polar, circular and phased orbit

In order to achieve repetitive observations under comparable light conditions, whichever area is observed and whatever the observation day is, the orbit of the satellite must be heliosynchronous, that is, it must display a constant angle with the direction of the Sun. A heliosynchronous orbit is nearly a polar orbit. This allows the satellite to take advantage of the Earth rotation to cover potentially the whole Earth surface. A circular orbit is chosen, in order to maintain a constant altitude of the satellite. Finally, the orbit is phased: the satellite, after a cycle of a fixed number of revolutions, goes back exactly to its previous positions with respect to Earth.

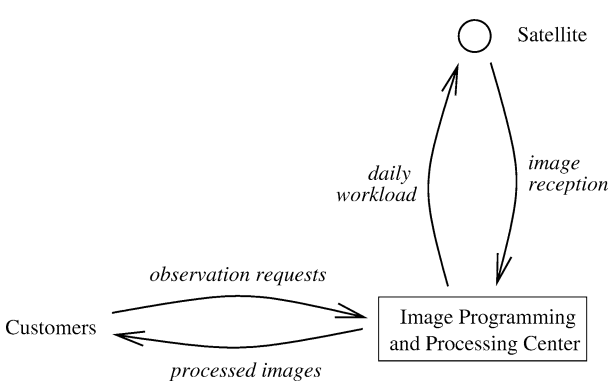

Fig. 1. Operating an Earth Observing Satellite.

2.1.3. Acquisition of images

As stated in the introduction, on an AEOS, the imaging instrument (a telescope) is not mobile and is fixed to the satellite, but the whole satellite can move on all axes, hence the name “agile”.

The telescope concentrates the light from the Earth on an array of aligned photo-diodes, thus enabling the acquisition of a landscape line in a single stroke. The combined movements of the satellite on its three axes (roll, pitch and yaw) and on its orbit, allow the formation of an image as a sequence of contiguous lines of pixels (see the Fig. 2). To facilitate their processing, images are constrained to be *strips* of rectangular shape (see Figs. 3 and 4). The width of a strip depends on the design of the instrument; as an

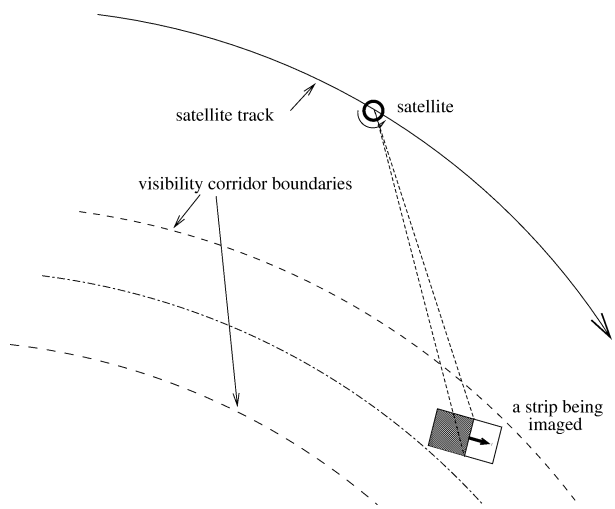

Fig. 2. An image acquisition. An Agile Earth Observing Satellite (AEOS) can image strips from a wide visibility corridor (typically 1000 km wide) under its track, enabling any aim (front, back or side) on the Earth surface, as well as any azimuth for the strip axis.

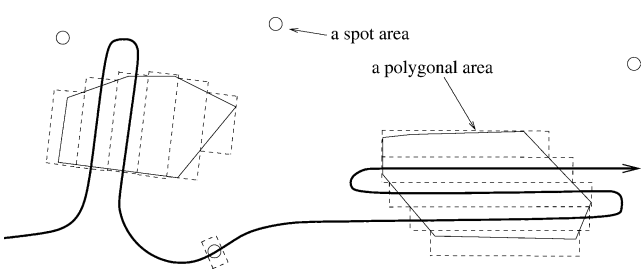

Fig. 3. A sequence of image acquisitions. Areas to be imaged (spots or polygons) are in solid thin lines. Strips (contour of images) are displayed dashed. The trace of the instrument aim over the ground is displayed as a solid thick curve.

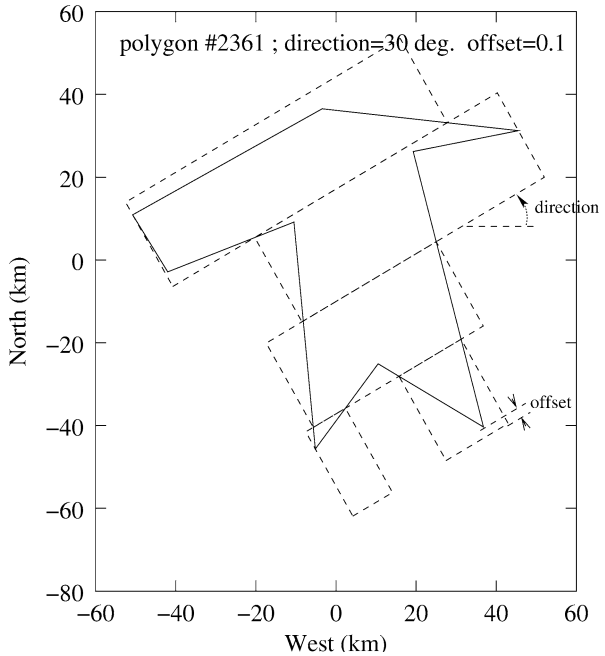

Fig. 4. The cutting up of a polygonal area into contiguous parallel strips of constant width is controlled by two parameters: the direction and the offset. The polygon is represented by a solid line, and the resulting strips by dashed lines.

acceptable simplification, it will be considered constant for all strips. On the other hand, the length of a strip is not fixed and may vary, up to a given limit. The acquisition speed is made constant over the strip, hence the duration of an image is directly proportional to the length of the strip.

2.1.4. From requests to images

An observation request concerns some specified area on the Earth surface. An area can have one out of two shapes:

- a spot (a small circular area, the radius of which is less than 10 km)
- a large polygonal area; the length of a polygonal area ranges from 20 to 100 km.

Each observation request must be analyzed and transformed into a set of strips covering the request. A spot can be covered by a single strip, while a polygonal area may require several contiguous strips to be completely imaged. Actually, strips are not required to be parallel, but this feature simplifies the pre and post-processing of images (see Fig. 4). The process of cutting-up polygonal areas into contiguous strips is controlled by two parameters: a *direction* (angle in the range 0 to 180 degrees between the strip axis and the South) and an *offset* (distance between a chosen point of the polygon and the lateral side of the last strip). Although the agility of the satellite allows any azimuth for the strip axis, in this study, a constant direction is chosen for all strips of all areas. This will be explained and justified in Section 2.2.1. The offset is such that the non productive strip surface is minimized.

It should be noticed that once a direction is chosen, a given strip can be imaged in either two opposite azimuths (an azimuth ranges from 0 to 360 degrees). So, two potential images are associated with each strip, and only one of them has to be acquired in order to acquire the strip.

2.1.5. Customers requests

The input data of the general AEOS selection and scheduling problem is the current set of observation requests from the customers. This set evolves continuously, with the addition of new coming requests, and with the withdraw of requests which have been (possibly partially) satisfied or which are out of date. Each request is associated with the following data:

- a *geometrical description* of the area to be imaged on the Earth surface. This description includes the location and the shape of the area (spot or polygon)
- a *validity period*, outside of which the request has no utility, usually given in days (for example: from 10.03.2002 to 30.06.2002)
- a set of *angular constraints*, (range of the possible angular aims) for imaging the request (for example a request may require an angular aim not greater than 30° in pitch and 45° in roll from the vertical axis)
- whether or not the request is *stereoscopic*. If this is the case, selected strips will have to be imaged twice during the same over-flight, with specified angular constraints, and using the same azimuth. This amounts to duplicate strips issued from stereoscopic requests
- a *weight* reflecting the importance of the request. This weight can be seen as the price that will be paid by the customer if the request is completely satisfied.

2.1.6. Time constraints and transition manoeuvres

Orbital parameters of the satellite allow us to compute for each image, from its validity period and angular constraints, a set of available time windows for acquiring the image.

As previously stated, an image acquisition requires a duration directly proportional to the length of the corresponding strip. On the other hand, the attitude change between two consecutive images is not instantaneous and requires a transition manoeuvre. Satellite kinematics allows us to compute for each pair of images a minimum duration of a transition manoeuvre. This minimum duration does not only depend on the location and the azimuth of the images themselves, but depends also on the precise instant time of the beginning of the transition manoeuvre (which is also the end of the first image acquisition).

Computing this minimum duration can be expressed as finding the first instant that gives a zero or positive value to a function depending only on the second image beginning time. The angular constraints can be converted in an allowed time interval. However, physical reasons lead to a large discontinuity of the minimum duration function, difficult to handle numerically in several meaningful cases.

This numerical problem leads to a time consuming algorithm which does not even cover actually all the discontinuity configurations. For solving selection and scheduling problem instances, where computation time is critical, an approximated tabulation technique was used (see Section 2.2.1).

2.1.7. Memory and energy constraints

Acquired images are stored on board until they can be downloaded towards a ground station. The available memory on board is limited and this can have an impact on the selection and scheduling process. Energy on board is a limited resource as well, but this limitation would be rarely reached. Memory and energy constraints have not been taken into account in this study. It is only planned to check that produced sequences satisfy these constraints, and if needed, to locally modify these sequences in order to meet them.

2.1.8. Meteorological uncertainties

An acquired image is not necessarily a satisfactory one, due to possible bad climatic conditions. Clouds preclude the good quality of images. Thus, a request (more precisely a set of images covering the request) may have to be selected several times, until it becomes acquired with satisfying climatic conditions. The selection and scheduling process can benefit from short term weather forecasts, in the following way: suppose that two equally weighted areas A1 and A2 can be acquired during the same time window, but not both of them; if we know that A1 will probably benefit from better weather conditions than A2, of course the acquisition of A1 must be favoured. Long term weather forecasts can also be used, for example to compute the expected number of times an area must be imaged before having a complete quality satisfaction. Section 2.2.1 points to a simplified but rational way of taking into account this uncertainty.

2.1.9. The quality criterion to be maximized

Although other criteria could be meaningful, the chosen quality criterion to be maximized is based on the sum of the weights of the satisfied requests (standard utility criterion). However, a difficulty arises from the fact that a request may be only partially covered by satisfactory images. What will be the reward for such partially covered areas? A simple policy is to choose, for each request, a reward proportional to the acquired surface. The corresponding criterion is named the *linear* quality criterion. This policy has the drawback to easily let many requests only partially covered (because algorithms tend to choose the longest, most rewarding images, and to neglect others). The Programming Center would like to favour the ending of acquisition of the almost acquired requests, before beginning new ones. This preference constraint can be taken into account by a modified quality criterion: we maximize the sum of *partial rewards* obtained on each request. The partial reward obtained on a request is a convex function of the acquired surface, instead of a linear one (see Fig. 6). This criterion is named the *non-linear* quality criterion. Both criteria are mathematically stated in Section 2.2.2.

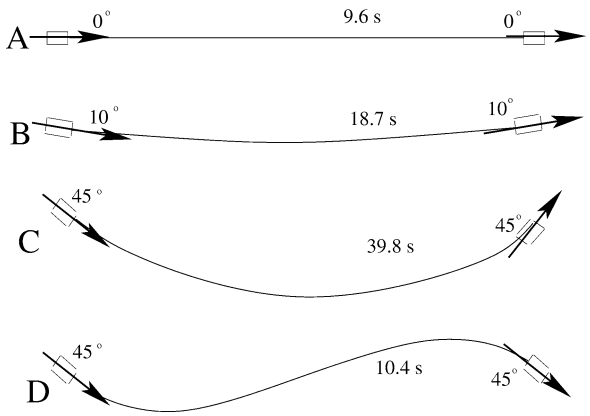

Fig. 5. Influence of azimuth changes on the duration of transition manoeuvres, here for transitions of 100 km length. Time transitions are very sensitive to the azimuth change between images. A deviation of 20 degrees results in a doubling of the transition time (from A to B), whereas this time is multiplied by four for the maximum possible deviation (90 degrees, from A to C). The transition time remains almost constant when the axes of the images are not aligned provided they are parallel (from A to D).

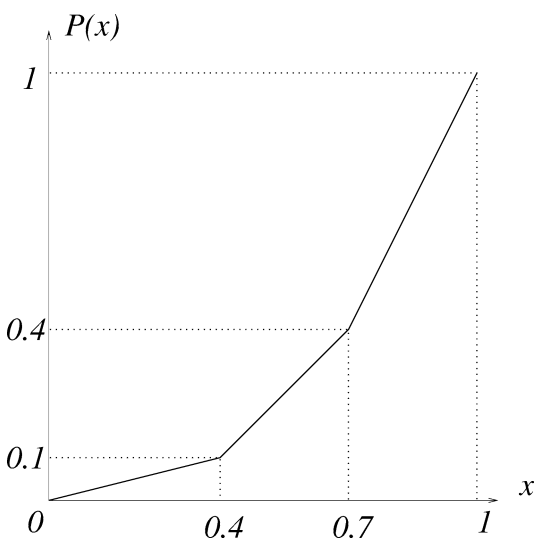

Fig. 6. Example of a partial reward function $P(x)$. The partial reward resulting from the acquisition of a fraction x of the total surface of a request of weight W is $W \times P(x)$.

As said before, the selection and scheduling problem consists of selecting daily the sequence of images that will be taken by the satellite over the next day. Let us call this short period of time (a day) the *scheduling horizon*. However, the selection process should maximize the quality criterion over a long period of time (for example one month or one year). Let us name this longer period of time the *optimization horizon*. Therefore, the sequence of computed schedules over the successive (short) scheduling horizons must maximize the sum of the *expected* partial rewards obtained over the (long) optimization horizon.

2.2. The Track Selection and Scheduling Problem: a simplification of the full problem

The previous section showed that the overall problem is both difficult to model (that is, to cast into a mathematically stated problem) and to solve. Modeling is a necessary step before solving, and it is common to say that modeling is simplifying. But simplifications are also needed to overcome the combinatorial nature of the problem, combined with the difficulties due to the sources of uncertainties. This section details which simplifications have been done to the full problem in order to be able to actually solve it.

2.2.1. Simplifying the full problem

Tracks as scheduling and optimization horizons. For the purpose of this work, we can define a *track* as an enlightened half-revolution of the satellite. The main simplification of the full problem is to process track by track, therefore taking the track as the scheduling horizon as well as the optimization horizon. Hence the name of AEOS Track Selection and Scheduling Problem for the simplified problem.

This simplification leads to sub-optimal solutions, because it does not take into account the fact that most of the images have several other opportunities after the current track, depending on the validity period associated with each request. However, the negative effects of this short term optimization horizon, as well as the effects of the meteorological uncertainties can be partially corrected by the modification of the weights of requests. Images having the smallest number of remaining feasible opportunities and the highest realization likelihood (good meteorological forecast for the next track) must be favoured in some way. A rational way of combining three criteria (the original weight, the number of remaining opportunities, the weather forecast) into a working weight is given in [20]. It is based on a mathematical approach drawn from the *Markov Decision Process* framework.

Geometry and transition manoeuvres. The cutting up of areas in strips is pre-computed, using a single direction for all requests of the same track. Fig. 4 displays the cutting up of a polygonal area, resulting in a set of contiguous parallel strips.

The choice of a constant direction is motivated by a main reason: changing the azimuth between two images is very costly in term of transition times, as showed by the results displayed in Fig. 5. Optimizing the cutting up of each polygon separately (so as to obtain the less possible number of strips for each one, or a minimum sum of the strips lengths) is likely to increase the duration of transition manoeuvres, and so to decrease the available productive time for proper imaging.

Choosing a single and constant direction for the cutting up of areas in strips appears to be an important restriction to the satellite capabilities, since the satellite agility is then partially unused. Actually, the satellite agility is still used (thus making a difference with non-agile satellites):

- the cutting up direction is maintained constant for a given track, but it can be freely chosen
- it can be changed between tracks, or even between two groups of images of the same track separated by a large gap, since in this case transition times are not a constraint
- once the direction is fixed (that is, an angle in the range 0 to 180 degrees), two opposite azimuths can still be used for imaging.

As said before (Section 2.1), transition times actually depend on the starting time of the transition. They may vary by a factor of 10% depending on the starting time, for a typical forward transition of 100 km. We decided to simplify and approximate these transition times by a constant mean duration time depending only on the two images to be joined (and not on the starting time of the transition). The model is therefore much simpler: with this simplification, checking that a given sequence of images satisfy the time constraints (visibility and manoeuvre) can be done in linear time. However, each produced solution schedule is checked afterward, using the true transition times. If the produced schedule does not pass the check, approximated transition times are majored and the schedule is computed again.

The choice of a constant cutting up direction has an interesting side effect: we said before that minimum transition times are very costly to compute. Choosing a constant direction allows for a pre-computation of a table of transition times of a reasonable size. This table is exploited later by interpolation, before each track processing, in order to obtain an approximation of the minimum transition times between each pair of candidate images of the processed track.

2.2.2. Mathematical statement of the AEOS Track Selection and Scheduling Problem

The input of a Track Selection and Scheduling Problem instance will be a set of candidate strips, from the current set of requests, that could be acquired on the track to be processed (request validity period and image visibility constraints satisfied).

As we have already noticed, a strip can be acquired using two opposite azimuths, as a result of the agility of the satellite. So, with each candidate strip are associated two possible candidate images. Only one image out of both has to be taken to acquire the corresponding strip.

The problem is to select, from the set of candidate strips, a feasible sequence of images, maximizing the quality criterion. Here, “feasible” means that the sequence must respect time constraints (visibility and angular constraints, duration of transition manoeuvres). In order to state these constraints more precisely, we need some precise notations.

Input data. Let R be the set of requests which could be at least partially scheduled on the track. For each $r \in R$, let W_r be its weight and A_r its surface area.

Let I be the set of candidate images of the track to be processed, obtained from R by the geometric cutting up pre-processing mentioned above. For each candidate image $i \in I$, we can compute from the given track parameters and visibility constraints:

- E_i: its earliest starting time;
- L_i: its latest starting time;
- D_i: its duration, strictly positive;
- A_i: its productive surface area.

We can also compute, for each possible pair of images (i, j), a quantity M_{ij}, strictly positive, which is a minimum of the transition duration from the end of i to the beginning of j. If a transition from i to j is not possible, a default very large value will be given.

Let B be the set of pairs of images (i, j) such that i and j are images from the same strip (opposite azimuths). Notice that $|I| = 2 \times |B|$.

Finally, let S be the set of pairs of corresponding stereoscopic images. The proportion of stereoscopic images is then $2 \times |S|/|I| = |S|/|B|$.

Decision variables. In this statement, three sets of decision variables are used: one for the selected set of images, one other to give the order of the selected images in the resulting sequence, and the last one for the starting times of image acquisitions.

Let $x_i,\ i \in I$, be 1 if the image i is selected, and 0 otherwise. Let $f_{ij},\ i, j \in I$, be 1 if i is followed by j in the selected sequence, 0 otherwise. Let t_i be the starting time of the acquisition of the image i, if selected.

Constraints. Here are the constraints which define feasible sequences:

$$\text{path}(x, f) \tag{1}$$

$$\forall i \in I\colon\ (x_i = 1) \Rightarrow (E_i \leqslant t_i \leqslant L_i) \tag{2}$$

$$\forall i, j \in I\colon\ (f_{ij} = 1) \Rightarrow (t_i + D_i + M_{ij} \leqslant t_j) \tag{3}$$

$$\forall (i, j) \in B\colon\ x_i + x_j \leqslant 1 \tag{4}$$

$$\forall (i, j) \in S\colon\ x_i = x_j. \tag{5}$$

The constraints (2) and (3) are time constraints (respect of time windows and transition times, respectively). The constraint (4) states that we need only one image per strip, and (5) enforces constraints on stereoscopic images. The constraint (1) expresses that the values of the decision variables x_i and $f_{ij},\ i, j \in I$, must represent an (oriented) simple path of selected images. It could be replaced by the following constraints:

$$\forall i \in I^+\colon\ \sum_{j \in I^+} f_{ij} = \sum_{j \in I^+} f_{ji} = x_i \tag{6}$$

$$x_0 = 1 \tag{7}$$

with $I^+ = I \cup \{o\}$, where o is a fictive image which begins and ends the sequence of selected images (with no time and no visibility constraints). The constraint (6) expresses that if an image is selected, there is an unique selected image preceding it, and an unique selected image following it. If an image is not selected, it has no preceding and no following selected image.[2]

Quality criteria. Two distinct quality criteria (to be maximized) will be considered. The linear criterion is

$$Q = \sum_{i \in I} W_i x_i$$

with $W_i = W_r \frac{A_i}{A_r}$, where r is the request the image i belongs to. The non-linear criterion, allowing us to favour the termination of polygons, is

$$Q = \sum_{r \in R} W_r P(x_r)$$

where P is the partial reward function introduced in Section 2.1 (see Fig. 6) and x_r is the acquired fraction of the surface of the request r, defined as

$$x_r = \sum_{i \in r} \frac{A_i}{A_r} x_i.$$

It can be easily checked that both criteria are equivalent when the P function is linear ($P(x) = x$).

Complexity of the problem. It can be shown that, despite the simplifications, the AEOS Track Selection and Scheduling Problem is NP-hard, which means that in practice, any algorithm solving this problem to optimality may need a computation time growing exponentially with the size of the instance to be solved [7]. In other words, this problem is highly combinatorial. As a matter of fact, it amounts to search for a longuest path in a directed graph with additional contraints. The vertices of the graph are the candidate images. There is a directed edge from an image to another if and only if a transition is possible from one image to the other. Notice that this graph can have cycles. Edges are labelled by two quantities: first, the corresponding transition time; second, a length which is the weight of the image corresponding to the origin of the edge. A time window and a duration are associated to each node. Additional constraints are just time constraints (2) in the above model, plus band and stereo constraints (4) and (5).

Numerical data. To better understand the nature of this combinatorial problem, we give in this paragraph some numerical data. The period of the satellite is 5850 seconds

[2] Notice that the classical subtour elimination constraints (see for example [13, p. 25]) are not required here because time constraints already prevent subtours.

(1 hour, 37 minutes and 30 seconds), so the duration of a track is about 49 minutes (half a period). The speed of the satellite on its track is 7531 m/s, and the speed of the vertical projection of the satellite over the ground is 6850 m/s. When the satellite is acquiring an image, the speed of the sighting point on the ground is maintained at 3420 m/s. The average difference between the earliest starting time and latest starting time of an image is 70 seconds. The typical duration of an image acquisition is 6 seconds, and the minimum transition time between the acquisition of two successive 100 km distant images is about 10 seconds. The number of requests per track may vary from 1 up to about 400, and the number of candidate strips per track ranges from 1 to about 600 (notice that there are two possible images for each strip). The proportion of stereoscopic requests is highly variable (see the data for some typical instances in Section 3.5).

2.3. Related work

The article [8] presents a problem, named the Space Mission problem (SM), which amounts to select and schedule a set of jobs on a single machine, among a set of candidate jobs. Each candidate job is associated with a fixed duration, a given time window and a weight. The aim is to select a feasible sequence of jobs maximizing the sum of weights. The SM problem is NP-hard. It is very close to our *restricted* AEOS Track Selection and Scheduling Problem. Actually, it is equivalent to the statement given in Section 2.2.2 with the following simplifications:

- null transitions times;
- no strip constraints (4);
- no stereoscopic constraints (5);
- a linear quality criterion.

Several interesting greedy algorithms and an optimal algorithm for solving the SM problem are proposed. They all suppose a discretization of time. The optimal algorithm is based on a Dynamic Programming scheme. Upper bound formulations are presented, based on a preemptive relaxation (a job can be fragmented) and on a lagrangean relaxation. The proposed algorithms are tested on 30 randomly generated instances with up to 200 candidate jobs.

The selection and scheduling problem for the SPOT5 satellite was partially stated in the introduction. It is a non-agile satellite with only one moving axis (rolling). A consequence of this lack of manoeuvrability is that the starting time of each candidate image is fixed. This feature would result in a very simple (polynomial) problem if there were only one imaging instrument on board, but there are 3 instruments. Nevertheless, it is possible to pre-compute binary and ternary hard constraints which model the compatibilities between candidate images. Each candidate picture is given a weight, and the problem is to find a feasible subset of the candidate images maximising the sum of weights. This optimisation problem is NP-hard, and can be formulated in the general Valued Constraint Satisfaction Problem model [16]. The article [2] fully describes the problem, and proposes some instances, to serve as a benchmark. Some results with dedicated exact and approximate methods are given in [3] and [22]. The column generation technique [5] has been used to compute upper bounds on the benchmark instances. Best and very good results on these instances have been obtained by Vasquez and Hao [18] using a dedicated Tabu Search algorithm. These last authors recently obtained very good upper bounds for these instances, by way of an original partition method [19], assessing the quality of their previous results. Notice that for some instances the optimum value of the quality criterion is still not known.

A new problem arises when the satellite is exploited by several entities. In this case, the objective is not only to give a maximal satisfaction to each entity, but also to enforce a kind of fairness constraint on selections. This share problem has been stated and studied in [14].

The work presented in [6] concerns a problem of selection and scheduling for a kind of semi-agile satellite, the agility of which is halfway between SPOT5 and PLEIADES. The differences with our AEOS problem are the following:

- the criterion to be maximized is the number of selected images (images are not weighted);
- the satellite is weakly mobile on two axes (pitch and roll), but remains fixed during an image acquisition; so, there is only one possible azimuth for an acquisition (parallel to the satellite motion);
- the satellite kinematics do not allow for a given strip to be acquired twice during the same track (contrary to agile satellites);
- there are no stereoscopic requests, hence no corresponding constraints.

Two related problems are stated. In the first one, named the Maximum Shot Sequencing Problem (MSP), the aim is to select and schedule a set of images over several consecutive tracks (a given image can be acquired from two or more tracks), so several possible disjoint time windows are given for each image. In the second problem, named the Maximum Shot Orbit Sequencing Problem (MSOP), only one track is processed, so a single time window is associated to each candidate image (as in our problem). MSOP and MSP also are NP-hard. Several algorithms are proposed for solving these problems, based on graph-theoretic concepts. In a first approach, time is discretized. Constraints given by the satellite kinematics are such that MSOP amounts to a longuest path problem (in a way similar to our DP approach). With time discretization, MSP amounts to find a maximal independent set in an incompatibility graph, and can be solved by an approximate algorithm based on a near-optimal partition into cliques. Due to the nature of the criterion, an interesting upper bound is available. Exact and approximate

algorithms are also presented for the continuous time model, to solve both MSP and MSOP. The exact algorithm is a branch-and-bound with graph-based heuristics and bounds. The approximate one is a kind of greedy algorithm also based upon graph properties. Experiments are conducted on a set of randomly generated instances, allowing to assess the proposed algorithms against upper bounds, and to measure the impact of the discretization. The results show the good quality of the proposed approximate algorithms.

The article [9] describes a scheduling problem involving a satellite equipped with a radar instrument, very agile on the pitch axis (thanks to an electronic scan) but slow on the roll axis. Only one azimuth is available for acquiring images. This problem is equivalent to the SM problem [8] with transition times. The authors describe a partial enumeration algorithm for this problem and give preliminary results for randomly generated instances.

The work reported in [23], in the context of the NASA's Earth Observing System domain, describes the so-called Window-Constrained Packing problem (WPC). It differs from the previously presented ones by the fact that the evaluation function (the return for selecting an observation) is a priority-weighted sum of observation durations under "suitability functions". In other words, observations have non fixed durations (ranging between a maximum and a minimum), and preference is given to higher priority observations, with longer durations, and best placed inside their time-window. There are no transition times between observations. This problem is similar to the SM problem [8], with a particular quality function, depending on starting times of observations. The authors present three approximate algorithms for solving the WPC problem. The first one is a fast but not very accurate greedy algorithm. The second one, similar to the first, but including some look ahead, is a little slower but much more accurate. The last one, a genetic algorithm, outperforms others but has expensive computation times. These algorithms have been tested on randomly generated instances. Actually the WPC problem is a short term simplified version of the real scheduling problem. The article investigates the mid-term and long term associated scheduling problems and their connections, as well as different objective functions.

3. Solving the AEOS Track Selection and Scheduling Problem

First, it should be noticed that the mathematical statement given above defines a Mixed Integer Programming problem, and as such is amenable to solving with specialized tools. Unfortunately, this way does not seem to give interesting results in practice. A reason could be the poor quality of cutting-plane bounds provided by this 0-1 variable model.[3] Another more general reason is the difficulty to infer orders between candidate images, due to the large time windows for the starting times. This probably also explain the disappointing behavior of complete methods on this problem. Four other directions have been investigated for solving the AEOS Track Selection and Scheduling Problem. Two of them, a Greedy Algorithm (GA) and Dynamic Programming Algorithm (DPA), are restricted to the linear quality criterion and do not take into account the stereoscopic constraint (numbered 5 in the above statement). The last two, a Constraint Programming Approach (CPA) and a Local Search Algorithm (LSA), have no restrictions and solve the problem with all constraints.

3.1. A Greedy Algorithm (GA)

This was our first designed algorithm. The aim was to obtain a fast but reasonably clever algorithm, stating a milestone for future algorithms.

The idea of the algorithm is to build progressively a sequence of images from the beginning of the track until its end. At each step, the selected image i is such that

- i can immediately follow the last selected image (respecting visibility and transition constraints), and the corresponding strip has not already been imaged in the beginning of the sequence;
- i maximizes the criterion $W_i + E(i)$, where $E(i)$ is an estimate of the value (sum of the weights) of the best possible sequence following i, if i is selected. Let $\widehat{Q}$ be an estimate of the optimal value on the whole track. $E(i)$ is defined as

$$E(i) = \widehat{Q} \times \rho(T_i), \quad \text{with } \rho(t) = \frac{t_{\max} - t}{t_{\max} - t_{\min}}$$

where T_i is the earliest possible ending time of image i considering the images already selected. $t_{\min}$ is the first possible time for beginning images on the track, and $t_{\max}$ the last possible one.

The justification for the expression of $E(i)$ is the following: it assumes that the value of the best possible sequence following i is proportional to the remaining time from the end of i to the end of the track. Of course, this assumption is only a rough approximation. Notice the similarity of this algorithm with a dynamic programming scheme.

In practice, $\widehat{Q}$ is initially set to the sum of the weigths of the candidate images, and the algorithm is run once, giving a first estimate of the optimal value on the whole track. Then, this estimate is used as a new value for $\widehat{Q}$, and the algorithm is run again. This process is itered 10 times, keeping the best produced schedule.

This sequential version of the Greedy Algorithm is not well adapted to instances involving stereoscopic requests, because they induce coupling constraints between pairs of non-consecutive images, that are difficult to take into

[3] We thank Michel Vasquez for this possible explanation.

account efficiently by a purely sequential process. Let us try to be more precise. A straightforward way of introducing a pair of stereoscopic images into the schedule is the following: if the first image is selected, then the second one is introduced too, as an immediate successor. However, as both images must be sufficiently separated (in order to meet stereoscopic conditions) the time interval between the acquisition of these images is rather large (about 60 seconds) and this would allow for at least one intermediate image to be included in between. It could be the first image of another steroscopic pair. So, if we decide to include possible intermediate images, then the algorithm must be different and it will depart strongly from its simple original form. If we decide not to include intermediate images, then the algorithm will be clearly sub-optimal. This is why this simple sequence-based greedy algorithm is not adapted to instances involving stereoscopic requests.

3.2. A Dynamic Programming Algorithm (DPA)

This algorithm originates from the following question: is there a modification of the problem which

- makes the problem solvable by a polynomial algorithm;
- has a low impact on the quality criterion?

The answer to this question is positive, at least if we consider only the linear criterion and if we ignore the constraint (5) on stereoscopic requests. We will make these assumptions in this section.

Each instance of the AEOS Track Selection and Scheduling Problem can be represented by a directed graph, in which the vertices are the pairs (i,t) where i is a candidate image, and t its possible starting times, and the directed edges represent the possible transitions between vertices (visibility and transition constraints satisfied). With each edge, we associate the weight W_i of the image corresponding to the origin of the edge. Time must be discretized in order to keep a finite graph. It should be noticed that this time-labelled graph is circuit-free. An optimal solution is just a longest path in this graph, provided that this path does not include two images of the same strip. This last constraint can be enforced by ordering the set of candidate images, allowing only edges in the graph respecting this order. The problem of finding a longest path in a circuit-free graph is solvable by a polynomial algorithm, namely an adapted standard shortest-path algorithm.

But which ordering of images can we choose? Fortunately, "natural" orders exist for the candidate images. For example, images can be ordered by their geographical positions in the track, or by the midpoint of their respective time-windows. These orders are well-configured in the sense that good schedules, obtained by hand or by the greedy algorithm (without any order on images), tend to respect them.

It should be emphasized that, due to this restriction on the possible transitions between images, the polynomial optimal algorithm solving the restricted problem (i.e., with an order on images) is no more optimal on the unrestricted problem (i.e., without ordering). Nevertheless, we accept this sub-optimality, hoping to obtain better results with the polynomial algorithm than with a practically sub-optimal resolution of the NP-hard non-restricted problem.

The polynomial algorithm evoked above can be attractively designed according to the Dynamic Programming paradigm (see for example [4, Chapter 16]). Let i be a candidate image and t a possible starting time for it. The local optimal value of the quality criterion $v^\star(i,t)$ (that is the highest possible sum of weights in any path issued from the image i, starting at time t) can be expressed as

$$v^\star(i,t) = \max\bigl\{v^\star(i',t') + W_i \text{ such that } C(i,t,i',t')\bigr\}$$

where $C(i,t,i',t')$ holds if and only if there is an edge in the graph from (i,t) to (i',t'), meaning that the visibility and transition constraints holding on i,t,i',t' are satisfied.

The Dynamic Programming algorithm is just an optimized way to compute the function $v^\star$ for all possible (i,t). In particular, the algorithm takes advantage of the following monotonicity property:

$$\forall i,t,t'\colon\ t < t' \Rightarrow v^\star(i,t) \geqslant v^\star(i,t')$$

which states that starting later cannot improve the value of the criterion.

Fig. 7 gives the pseudo-code of the Dynamic Programming algorithm. The notations and data structures used by this algorithm are as follows (refer to the Section 2.2.2 for already defined notations).

- an image i is denoted by an integer ranging from 0 to $|I|-1 = 2\times|B|-1$;
- the integer variable d denotes a possible starting time for an image, relative to the beginning of the temporal window for the image: if t is an absolute starting time for i, then $t = E_i + d$;
- $T_i = L_i + 1 - E_i$ is the number of possible starting times for the image i. This number of course depends on the time discretization;
- $g[i,d]$ is the local optimal value of the quality criterion for a partial schedule beginning by image i if i is started at relative time d. At the end of the calculation, we have $g[i,d] = v^\star(i, E_i + d)$ with the above notations;
- *inext*$[i,d]$ is the image which follows the image i in a local optimal schedule beginning by image i if i is started at relative time d;
- *dnext*$[i,d]$ is the relative starting time of *inext*$[i,d]$. This array is used at the end of the computation to recover an optimal sequence of images;
- $g^\star$ is the optimal value of the quality criterion.

The constraint (4) of the model (Section 2.2.2) stating that only one image per strip is required, is implicitly satisfied by the way the images are enumerated inside the nested loops of the algorithm.

```
DPA()
  Sort the set of strips B by position order
  for b ← |B| − 1 downto 0 // b is the strip index
    for k ← 0 to 1 // opposite azimuths for the strip b
      i ← 2b + k // i is the image index
      for d ← T_i − 1 downto 0
        if d = T_i − 1 then
          g[i, d] ← W_i
          inext[i, d] ← −1 // marks the last image of a schedule
        else
          g[i, d] ← g[i, d + 1]
          inext[i, d] ← inext[i, d + 1]
          dnext[i, d] ← dnext[i, d + 1]
        for b' ← b + 1 to |B| − 1
          for k' ← 0 to 1
            i' ← 2b' + k'
            UPDATE(i, d, i')
  // Recovering an optimal schedule
  g* ← 0
  for i ← 0 to 2|B| − 1
    if T_i ≠ 0 and g* < g[i, 0] then
      g* ← g[i, 0]
      i* ← i // first image of an optimal schedule

UPDATE(i, d, i') =
  t' ← E_i + d + D_i + M_ii'
  if t' > L_i' then return // too late for image i'
  d' ← t' − E_i'
  if d' < 0 then
    if I_i' = 0 then return // no visibility for image i'
    d' ← 0 // first possible relative starting time for i'
  if g[i', d'] + W_i > g[i, d] then
    // it is better to follow i by i'
    g[i, d] ← g[i', d'] + W_i
    inext[i, d] ← i'
    dnext[i, d] ← d'
```

Fig. 7. Pseudo-code of the Dynamic Programming algorithm.

The algorithmic complexity of this algorithm is clearly $O(|B|^2 \cdot \max_i T_i)$, in which $\max_i T_i$ is proportional to the inverse of the time discretization interval.

In the actual implementation, the expression $t' \leftarrow E_i + d + D_i + M_{ii'}$ in UPDATE is replaced by $t' \leftarrow d + EDM_{ii'}$. The quantities $EDM_{ii'}$ are pre-computed and discretized as a whole in order to minimize discretization errors.

We discuss now briefly the impact of the non-linear criterion and the stereoscopic requests on this Dynamic Programming algorithm. These are difficult to take into account because they need to include into each state (i, t) a part of the history (the path in the graph by which each state is reached). A straightforward adaptation of the previous algorithm would require memory resources growing exponentially with the number of stereoscopic requests, resulting in an impracticable algorithm. A more clever adaptation is needed.

3.3. A Constraint Programming Approach (CPA)

Constraint Programming is a generic technique for modeling and solving combinatorial problems. See for example [11,12,15]. It is now reaching a mature state. We have tried to use this existing framework and its associated algorithms to solve the Track Selection and Scheduling Problem.

We chose the OPL language and framework [17], because of its elegance and simplicity. With OPL (*Optimization Programming Language*), one describes a *model* of a problem: data, decision variables, constraints and optimization criterion. A solver is associated with the language. It searches through the search space described by the model, trying to find optimal solutions, in an ordered and systematic way. The underlying algorithms are said to be *complete*, which means that they offer the guarantee to find an optimal solution, provided they are given enough resolution time. The constraint programming approach is very flexible. It allowed us to easily take into account all operational constraints (non-linear quality criterion as well as stereoscopic requests).

Nevertheless, some care must be taken when building the model, so that it can be solved in practice. In our problem, in order to keep a practicable size for the search space, a key idea is to solve the problem using several successive resolutions: in each resolution, we search for an optimal sequence of a *fixed* length. Several resolutions are launched with increasing lengths, until inconsistency, in order to find an overall optimal sequence.

The solver must be guided in the search space, in order to find good solutions as quickly as possible. OPL provides some ways to do it. In particular, we can state which decision variables have to be instantiated first. This greatly helps the solver.

The above mentioned techniques are not sufficient to solve models in a reasonable amount of time. A way to control the combinatorial explosion is to cut down the search space by adding constraints. These constraints may be redundant, that is induced by others. In this case we keep the possibility to find an optimal solution. If added constraints are not redundant, then the optimality is no more guaranteed, but yet can be approached when the additional constraints are justified by some heuristics. In our case, the following heuristics have been used to add constraints to the model:

- for large instances, the candidate images are limited to those having a chance to be part of a good solution: we eliminate images having very small weights;
- in a good solution, the order in which images appear in the sequence, more or less obeys the order in which images are flown over by the satellite; this means that we can constrain each candidate image to appear only in a given sub-sequence of the whole sequence (for example, an image which belongs to the middle of the track is constrained to belong to the middle of the sequence if it is selected);
- following the same idea, an optimal sequence will seldom include a “backtrack”, that is a transition between two images going in the opposite movement of the satellite, even if the agility of the satellite permits it; so we add a constraint, not forbidding backtracks, but limiting them in range;

- taking advantage of the multi-pass resolution process explained above, successive resolutions are conducted in increasing order of the sequence length, and we force the optimal sequence computed in a resolution to be a subset of the solution sequence of the next resolution.

The use of these additional constraints of course removes the guarantee of finding optimal solutions. But associated with a clever way to limit the computation time at each step, the enriched model enables us to solve most instances of the problem, in a reasonable amount of time, with acceptable quality results.

3.4. A Local Search Algorithm (LSA)

Local search (see for example [1]) is a well-known general technique for solving highly combinatorial problems, including EOS selection and scheduling problems [18,23]. When faced with a very large search space, as this can be found in our application, local search is often a valuable alternative. It includes methods like *Descent Search*, *Tabu Search*, *Simulated Annealing*, *Genetic Algorithms* ... These methods are in essence incomplete, meaning that they cannot give any proof of optimality of the solutions generated, and no information about the quality of the obtained solutions.

We have adapted the principles of local search to the AEOS Track Selection and Scheduling Problem in the following ways. A current feasible solution sequence of images is maintained. A better solution is searched for in the neighbourhood. A neighbouring solution is obtained either by

- inserting a new image in the current sequence, accepting the insertion if the new sequence is feasible;
- or by removing an image from the current sequence.

The key points in local search algorithms are:

- the use of heuristics in order to guide the search towards good regions of the search space;
- a well tuned level of randomness: too much randomness prevents the heuristics to be efficiently followed, and not enough randomness tends to restrict the search to a deterministic walk.

The following tunings seem to give good results in our application:

- insertion or removal is decided randomly but nonuniformly, according to a probability which evolves during the search: a successful insertion increases the probability of future insertions, and conversely, a failure in an insertion decreases this probability;
- a new image to be inserted is randomly but non uniformly selected, among those currently not selected, with a probability proportional to the weight of the image;
- the place where to insert a new image is uniformly randomly chosen.

Figs. 8–10 give the detailed pseudo-code of the Local Search Algorithm. The notations used are the following.

3.4.1. Parameters of the local search

- n_t: number of tries;
- n_m: number of movements (insertion or removal of an image) per try;
- $\alpha \in [0, 1]$: evolution factor for the probability of insertion p_a in case of success of insertion;
- $\beta \in [0, 1]$: evolution factor for the probability of insertion p_a in case of failure of insertion.

All these parameters are constant during a search. They determine the global behavior of the algorithm.

3.4.2. Global variables

- *SI*: the current sequence of selected images;
- *NSB*: the current set of non selected strips;
- p_a: probability that the next movement is an insertion;
- *currentValue*: value of the quality criterion for *SI*;
- *bestSI*: best sequence *SI* found so far;
- *bestValue*: value of the quality criterion for *bestSI*.

With each image i are associated two variables:

- e_i: the earliest starting time for i within *SI*;
- l_i: the latest starting time for i within *SI*.

Notice that these times can be different from E_i and L_i because of the propagation of temporal constraints inside the sequence *SI*.

The main procedure LSA (Fig. 8) is merely a loop performing n_t tries. The ONETRY procedure starts from an empty sequence *SI*. The choice between an insertion movement (INSERTIONMOVE) or a removal movement (REMOVALMOVE) is decided randomly, depending on the probability p_a. INSERTIONMOVE returns **True** if the insertion succeeds, **False** otherwise. As said above, the probability p_a of a future insertion evolves according to the success of the insertion.

INSERTIONMOVE first selects randomly a strip b among the set of non selected strips *NSB*. The probability of selecting a given strip is proportional to the weight of the corresponding images. Then an azimuth is randomly chosen among the two possible opposite azimuths, resulting in an image i, candidate for insertion into *SI*. A set of possible positions P for this insertion is returned by POSSIBLEPOSITIONS. The insertion position p is chosen randomly in P. Then, if the image i has a corresponding stereoscopic image, the insertion of the latter is tried. If it fails, previous state of *SI* and *NSB* is restored.

```
LSA()
  bestValue ← 0;   n ← 1
  while n ≤ n_t
    OneTry()
    n ← n + 1

OneTry()
  NSB ← Copy(B); SI ← ∅;   p_a ← 1;   j ← 1
  while (j ≤ n_m) and (NSB ≠ ∅)
    if Random(0,1) ≤ p_a then
      if InsertionMove() then  // success : p_a is increased
        p_a ← min(1, p_a/α)
        if currentValue > bestValue then
          bestValue ← currentValue; bestSI ← SI
      else  // failure : p_a is decreased
        p_a ← p_a · β
    else
      RemovalMove()
    j ← j + 1

InsertionMove()
  b ← RandomStrip()
  i ← RandomAzimuth(b)
  P ← PossiblePositions(i)
  if P = ∅ then return False
  p ← RandomPosition(P)
  AddToCurrentSequence(i, p)
  if i has a corresponding stereo image then
    i' ← stereo image corresponding to i
    P' ← PossiblePositions(i')
    if P' = ∅ then
      RemoveFromCurrentSequence(i)
      return False
    else
      p' ← RandomPosition(P')
      AddToCurrentSequence(i', p')
  update currentValue
  return True

RemovalMove()
  if SI = ∅ then return
  i ← RandomImage()
  RemoveFromCurrentSequence(i)
  if i has a corresponding stereo image then
    i' ← stereo image corresponding to i
    RemoveFromCurrentSequence(i')
```

Fig. 8. Local Search Algorithm: main procedures.

```
PossiblePositions(i)
  s ← |SI|
  if s = 0 then return {0}
  P ← ∅
  // is it possible to insert i at the beginning of SI ?
  j ← SI[0] // first image of SI
  if E_i + D_i + M_ij ≤ l_j then add 0 to P
  // is it possible to insert i inside SI ?
  for p ← 1 to s − 1
    j ← SI[p]
    k ← SI[p − 1]
    e_i ← max(E_i, e_k + D_k + M_ki) // earliest starting time for i
    l_i ← min(L_i, l_j − D_i − M_ij) // latest starting time for i
    // is it possible to insert i between k and j ?
    if e_i ≤ l_i then add p to P
  // is it possible to insert i at the end of SI ?
  j ← SI[s − 1] // last image of SI
  if e_j + D_j + M_ji ≤ L_i then add s to P
  return P
```

Fig. 9. Local Search Algorithm: search for possible insertion positions.

```
AddToCurrentSequence(i, p)
  insert i in SI at position p
  update NSB : remove the strip of i from NSB
  Propagation()

RemoveFromCurrentSequence(i)
  remove i from SI
  add the strip of i to NSB
  Propagation()

Propagation()
  s ← |SI|
  if s = 0 then return
  // compute earliest starting times
  j ← SI[0]; e_j ← E_j // first image of SI
  for p ← 1 to s − 1
    i ← SI[p − 1]
    j ← SI[p]
    e_j ← max(E_j, e_i + D_i + M_ij)
  // compute latest starting times
  j ← SI[s − 1]; l_j ← L_j // last image of SI
  for p ← s − 2 downto 0
    j ← SI[p]
    k ← SI[p + 1]
    l_j ← min(L_j, l_k − D_j − M_jk)
  // check consistency of starting times
  foreach i ∈ SI
    if e_i > l_i then error
```

Fig. 10. Local Search Algorithm: adding and removing a given image to/from the current sequence, and full propagation of temporal constraints.

REMOVALMOVE selects randomly an image among the sequence *SI* for removal. The probablity of selection of an image is inversely proportional to its weight. If needed, the corresponding stereographic image is removed too.

POSSIBLEPOSITIONS, Fig. 9, computes the set of possible positions for the insertion of the image i in *SI*. Possible insertion positions are numbered from 0 to s, where s is the length of the sequence *SI*.

ADDTOCURRENTSEQUENCE, Fig. 10, calls PROPAGATION for a complete propagation of temporal constraints. PROPAGATION first computes earliest starting times e_i for each image in the sequence using an upward loop, then latest starting times l_i using a downward loop. The next loop checks the consistency of these temporary starting intervals. It is worth noticing that it is just a verification, since PROPAGATION is always called either for an insertion in a position previously checked by POSSIBLEPOSITIONS, or for a removal. This procedure has room for possible improvements[4] but we let it in this form for simplicity.

An irritating drawback of local search methods is their non-determinism: each execution results in a different solution. To get an idea of the performance of a local search algorithm, it is a common practice to compute a *quality pro-*

[4] For example, propagation loops need only to start from the insertion or removal position, as all e_i and l_i do not have to be recomputed.

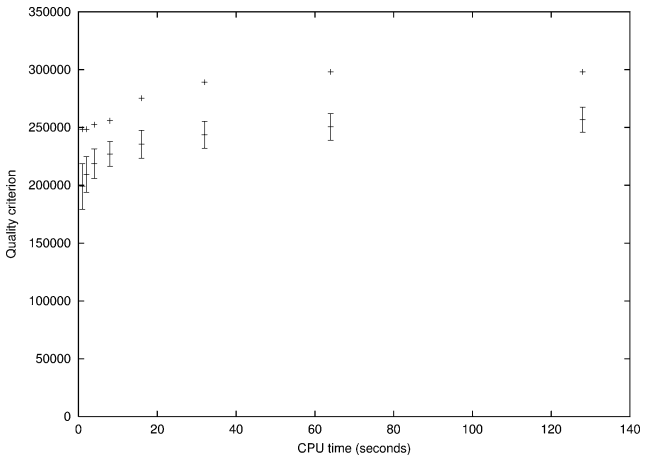

Fig. 11. A quality profile obtained with the local search algorithm for the resolution of a large instance of the AEOS Track Scheduling Problem (instance id = 3:25_22, 342 candidate strips). Average with standard deviation and maximum value of the quality criterion, obtained after a given CPU elapsed time, over 100 different executions of the local search algorithm.

file, such as the one presented in Fig. 11. It gives the average, standard deviation and maximum value of the quality criterion obtained after a given CPU elapsed time, over 100 different resolutions of a large instance. In our case, it can be seen that a steady average and maximum quality is obtained after 1 minute of CPU time. For this instance, average values are within 15% of the maximum, and standard deviations are rather small.

3.5. Comparisons of methods and algorithms

3.5.1. Performance comparisons

We compared the performances of the four presented methods, by running them on six representative instances of the AEOS Track Selection and Scheduling Problem. These instances are extracted from a set of artificial instances, built by the organism which will be the interface with clients, foreseeing the requests to such a system over the 2005 horizon. A first experiment was conducted, involving the four presented methods, but restricted to the linear quality criterion, and processing stereoscopic requests as if they were unrelated (that is, by removing the constraint labeled (5) in Section 2.2.2). Results are presented in Table 1, middle part. DPA always shows the best ratings, despite the restriction to a predefined order of images (which other algorithms have not). Moreover, the algorithm is rather fast. GA is also very fast, but is not so efficient. CPA gives disappointing results.

Unfortunately, the two best algorithms of the previous experiment are not able to solve the problem with all operational constraints. A second experiment was conducted only with methods which are able to take into account all operational constraints (non-linear quality criterion and stereoscopic constraint). The results, obtained in the same conditions as before, are displayed in Table 1, lower part. It shows that LSA always gives the best results.

Table 1
Compared performances of algorithms on selected instances. GA: Greedy Algorithm; DPA: Dynamic Programming Algorithm; CPA: Constraint Programming Approach; LSA: Local Search Algorithm. The upper table gives some statistics on the instances: $|R|$ is the number of requests of the instance; the triplet (*ster circ poly*) gives respectively *ster*: the number of stereoscopic requests, *circ*: the number of requests which are circular small areas (covered by one image), *poly*: the number of requests which are large polygonal areas (covered by several images). $|I|$ is the number of candidate images, P_s is the proportion of stereoscopic images among the candidate images, and D is the average duration (in seconds) of an image acquisition for the considered instance. The column labelled $|V|$ gives the number of vertices of the generated graph for the Dynamic Programming algorithm (the chosen time discretization for these experiments is 1 second). The middle table gives the results obtained on the restricted problem (not all operational constraints enforced, see Section 3.5). The lower table gives the result obtained on the full problem (with only two methods available). The middle and lower tables display the value of the quality criterion obtained with each method (columns) and each instance (rows), with a CPU time limited to 2 minutes per instance for each method, except for LSA which was run 100 times (2 minutes each run) in order to obtain a meaningful average. For LSA, values of parameters are $n_m = 20000, \alpha = 0.5, \beta = 0.996$, and n_t is automatically ajusted to fit the runtime limit. Best performances are displayed bold-face

Instance id	$\|R\|$	*ster*	*circ*	*poly*	$\|I\|$	P_s	D	$\|V\|$
2_13_111	68	12	64	4	212	0.23	5.6	14523
2_15_170	218	39	206	12	590	0.26	5.0	36271
2_26_96	336	55	308	28	966	0.26	5.6	55776
2_27_22	375	63	346	29	1068	0.26	5.7	67582
3_25_22	150	87	17	133	684	0.69	7.7	12326
4_17_186	77	40	23	54	294	0.71	6.9	6352

Instance id	GA	DPA	CPA	LSA av. (max.)
2_13_111	532	**603**	442	574 (587)
2_15_170	707	**843**	527	723 (779)
2_26_96	831	**1022**	782	826 (877)
2_27_22	895	**1028**	777	800 (861)
3_25_22	436	**482**	253	345 (375)
4_17_186	188	**204**	177	192 (196)

Instance id	CPA	LSA av. (max.)
2_13_111	241	**414** (490)
2_15_170	350	**446** (490)
2_26_96	439	**516** (592)
2_27_22	410	**455** (561)
3_25_22	149	**255** (298)
4_17_186	125	**145** (156)

3.5.2. Other comparisons

Apart from performances, methods have also to be compared on other grounds, namely simplicity and ease of understanding, flexibility (i.e., ease of change), quickness to bring into play.

The Greedy Algorithm and the Dynamic Programming Algorithm belongs to the same family of algorithms (GA could be considered as a weakened Dynamic Programming).

They give solutions of good quality in a short computation time, but as they are, they stumble over the problem of taking into account constraints that connect distant states (non-linear criterion, stereoscopic requests).

Constraint Programming is the easiest method to bring into play. A great flexibility is permitted in the way models can be built. Potentially any kind of constraints can be expressed. Robustness is also a quality of the approach, allowing the model to evolve easily by local modifications. These qualities are particularly useful in real-word applications. On the other hand, mastering the combinatorial explosion can be a challenging task. Models must be necessarily tuned for an efficient search. Some support tools are provided for help, but they can be difficult to use.

Local search is easy to understand, and basic local search algorithms are simple enough to implement. The method is also able to take into account potentially any constraint. However, less flexibility is available since constraints have to be wired into the code. Last, exploiting local search can be rather frustrating, due to the inherent non-determinism of the method, and to the difficulty of tuning the numerous search parameters.

4. Conclusions

In this article, we described the general problem of managing the new generation of agile Earth Observing Satellites, and a way to cast it in a simpler problem: the AEOS Track Selection and Scheduling Problem. Even this simplified problem remains computationally hard. Four quite different methods have been investigated in order to solve it: a fast Greedy Algorithm, a Dynamic Programming Algorithm, a method based on the existing Constraint Programming framework, and a Local Search Algorithm, based on insertion and removal of images in a sequence. The last two methods were able to take into account all specific operational constraints of our real-world problem, in particular stereoscopic requests. Each one has its own advantages. To summarize, the Constraint Programming Approach is very flexible, while the Local Search Algorithm gives better performance. However, adaptations of the Greedy Algorithm and of the Dynamic Programming Algorithm to the resolution of the full problem should be also investigated, as well as weakened versions of Branch-and-Bound algorithms, such as *Limited Discrepancy Search* [10].

Other schemes for local search should also be investigated, as there is probably room for improvements in this direction too.

The future of this work concerns the extension of this kind of methods towards the better resolution of the general selection and scheduling problem, taking into account the prevision of arrival of future requests and weather forecast more accurately, as well as a multi-track version of the problem.

Other interesting investigations include generalizations of the described problem

- taking into account capacity constraints (memory and energy consumptions) and data dump opportunities;
- involving several satellites devoted to the same observing mission;
- using selection processes operating jointly at different horizons of time (mid-term and long-term scheduling);
- aiming to replace the daily computed selection and scheduling, by a probably more efficient reactive and continuous planning [21].

Acknowledgements

The authors would like to thank Roger Mampey for his help with the geometric aspects of this study, especially for his contribution to the polygon cutting-up algorithm. The anonymous reviewers also must be thank for their careful reading and useful comments of a first version of this article.

References

[1] E. Aarts, J.K. Lenstra, Local Search in Combinatorial Optimization, Wiley, 1997.
[2] E. Bensana, M. Lemaître, G. Verfaillie, Earth observation satellite management, Constraints: An Internat. J. 4 (3) (1999) 293–299.
[3] E. Bensana, G. Verfaillie, J.C. Agnèse, N. Bataille, D. Blumstein, Exact and approximate methods for the daily management of an Earth observation satellite, in: Proc. 4th International Symposium on Space Mission Operations and Ground Data Systems (SpaceOps-96), Münich, Germany, 1996.
[4] T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, The MIT Press, Cambridge, MA, 1990.
[5] V. Gabrel, Improved linear programming bounds via column generation for daily scheduling of earth observation satellite, Technical Report, LIPN, Université 13 Paris Nord, January 1999.
[6] V. Gabrel, A. Moulet, C. Murat, V.T. Paschos, A new single model and derived algorithms for the satellite shot planning problem using graph theory concepts, Ann. Oper. Res. 69 (1997) 115–134.
[7] M.R. Garey, D.S. Johnson, Computers and Intractability, a Guide to the Theory of NP-Completeness, Freeman, 1979.
[8] N.G. Hall, M.J. Magazine, Maximizing the value of a space mission, European J. Oper. Res. 78 (1994) 224–241.
[9] S.A. Harrison, M.E. Price, Task scheduling for satellite based imagery, in: Proc. Eighteenth Workshop of the UK Planning and Scheduling Special Interest Group, University of Salford, UK, 1999, pp. 64–78.
[10] W.D. Harvey, M.L. Ginsberg, Limited discrepancy search, in: Proc. IJCAI-95, 1995, pp. 607–613.
[11] P. Van Hentenryck, V. Saraswat, Strategic directions in constraint programming, ACM Comput. Surveys 28 (4) (1996).
[12] J. Jaffar, M. Maher, Constraint logic programming: A survey, J. Logic Programming 19/20 (1994) 503–582.
[13] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, D.B. Shmoys, Traveling Salesman Problem, Wiley, 1985.
[14] M. Lemaître, G. Verfaillie, N. Bataille, Exploiting a common property resource under a fairness constraint: A case study, in: Proc. 18th IJCAI-99, Stockholm, Sweden, 1999, pp. 206–211.
[15] A.K. Mackworth, E.C. Freuder, The complexity of some polynomial network consistency algorithms for constraint satisfaction problems, Artificial Intelligence 25 (1985) 65–74.

[16] T. Schiex, H. Fargier, G. Verfaillie, Valued constraint satisfaction problems: hard and easy problems, in: Proc. IJCAI-95, Montréal, Québec, 1995, pp. 631–639.

[17] P. Van Hentenryck, ILOG OPL 3.0, Optimization Programming Language, Reference Manual, ILOG, January 2000.

[18] M. Vasquez, J.-K. Hao, A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite, J. Comput. Optim. Appl. 20 (2) (2001) 137–157.

[19] M. Vasquez, J.-K. Hao, Upper bounds for the SPOT5 daily photograph scheduling problem, J. Combinatorial Optim, 2002, to appear.

[20] G. Verfaillie, E. Bensana, C. Michelon-Edery, N. Bataille, Dealing with uncertainty when managing an Earth observation satellite, in: Proc. 5th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-99), Noordwijk, The Netherlands, 1999, pp. 205–207.

[21] G. Verfaillie, E. Bornschlegl, Designing and evaluating an on-line on-board autonomous Earth observation satellite scheduling system, in: Proc. Second NASA International Workshop on Planning and Scheduling for Space, San Francisco, CA, 2000, pp. 122–127.

[22] G. Verfaillie, M. Lemaître, T. Schiex, Russian doll search for solving constraint optimization problems, in: Proc. AAAI-96, Portland, OR, 1996, pp. 181–187.

[23] W.J. Wolfe, S.E. Sorensen, Three scheduling algorithms applied to the Earth observing systems domain, Management Sci. 46 (1) (2000) 148–168.