

Selecting and scheduling observations for agile satellites: some lessons from the constraint reasoning community point of view

G rard Verfaillie and Michel Lema tre

ONERA, Center of Toulouse
2 avenue  douard Belin, BP 4025, 31055 Toulouse Cedex 4, France
{Gerard.Verfaillie, Michel.Lemaitre}@cert.fr
<http://www.cert.fr/dcsd/cd/THEMES/oc.html>

Abstract. This paper presents some lessons that can be drawn, from the point of view of the *constraint reasoning* and *constraint programming* community, from trying to model and to solve as best as possible the *mission management* problem for the new generation of *agile* Earth observation satellites, that is the *selection* and the *scheduling* of observations performed by the satellite.

1 Introduction

The mission management problem for the current generation of Earth observation satellites, like those of the French *Spot* family, has been already presented [3]. Various methods, able to solve it either optimally or approximately, have been proposed and compared [4, 3, 14].

This paper is devoted to the mission management problem for the new generation of *agile* Earth observation satellites, like the already operational American *Ikonos* satellite and those of the future French *Pl iades* family.

The main difference between both these generations of satellites lies on the *degrees of freedom* that are available for image acquisition. Whereas the non-agile *Spot* satellites have only *one* degree of freedom, along the roll axis, provided by a mobile mirror in front of each instrument, the agile *Pl iades* satellites will have *three* degrees of freedom, along the roll, pitch, and yaw axes, provided by the attitude control system of the whole satellite. Whereas there is, with the *Spot* satellites, only *one* way of acquiring an image of a given area on the Earth surface from a satellite revolution, there will be, with the *Pl iades* satellites, an *infinite* number of ways of acquiring it from a satellite revolution, because the *starting time* and the *azimuth* of image acquisition will be free.

The first consequence of this greater freedom is an expected better *efficiency* of the whole imaging system. The second one is a far larger *space* (in fact infinite) of imaging opportunities, and consequently a far greater *complexity* of the management problem.

In this paper, we describe the *mission management* problem for agile Earth observation satellites, as it has been stated by the *CNES*¹ managers of the *Pléiades* project (Section 2). Then, we describe the *simplifications* we had to do in order to get a manageable problem (Section 3). We show how this simplified problem can be *mathematically* stated (Section 4) and describe the four *algorithms* or *approaches* we designed, implemented and experimented for solving it (Section 5). We show and discuss the *experimental* results that have been obtained on training instances provided by the *CNES* (Section 6). We conclude with some *lessons* that we drew from this study and that deserve, in our opinion, discussion in the *constraint reasoning* and *constraint programming* community (Section 7).

2 Problem description

Satellite orbit Earth observation satellites use specific orbits that are:

- *quasi polar*: the satellite orbital plane passes nearly through the Earth north and south poles; the conjunction of a quasi polar orbit with the natural Earth movement around its polar axis allows the whole Earth surface to be flown by the satellite each day (see Figure 1);
- *circular*; this implies a constant image acquisition altitude;
- *heliosynchronous*: the angle between the satellite orbital plane and the Earth-Sun axis remains constant during the whole year; this implies constant Earth illumination conditions for image acquisition; note that the satellite can only acquire images during the illuminated part of each revolution;
- *phased*: after a given number of revolutions, the satellite goes back exactly at the same position with respect to Earth.

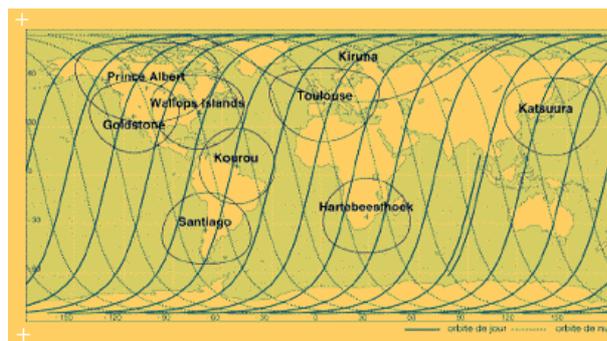


Fig. 1. The track of the satellite on the Earth surface during one day.

¹ French space agency: <http://www.cnes.fr>.

Image acquisition degrees of freedom The satellite is compactly built around one *optical instrument*. At any time, it is moving on its orbit and can simultaneously move around its roll, pitch and yaw axes, thanks to its *attitude control* system.

The core of the instrument is made up of a set of aligned photo-diodes that allow at any time a *segment* on the Earth surface to be acquired as a set of aligned pixels. The combined translation and rotation movements of the satellite allow then an *image* to be acquired as a set of contiguous segments (see Figure 2).

To simplify their processing, these images are constrained to be rectangular *strips*. Although the *width* of these strips actually depends on the acquisition angle, we consider that it is fixed and equal to its minimum value (obtained exactly under the satellite orbit). Their *length* and their *direction* (from 0 to 180 degrees) are however free.

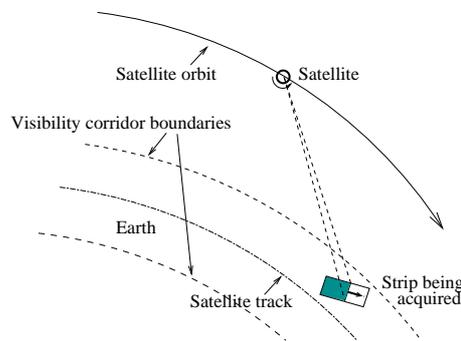


Fig. 2. Acquisition of a rectangular strip.

User requests Observation requests can be submitted by users at any time. Each of these requests is defined by:

- a *target area*, which can be, either a *spot* (a small circular area), or a *polygon* (a large polygonal area);
- a *validity period*, outside of which its acquisition has no utility (usually specified in days);
- a set of *acquisition angular constraints* (minimum and maximum roll and pitch angles);
- a *type*, which can be either *mono* or *stereo*; in case of a *stereoscopic* request, an associated selected strip must be acquired twice during the same illuminated half-revolution, by satisfying specified acquisition angular constraints and by using the same azimuth;
- a *weight*, which expresses its importance.

From requests to images A *spot* can be covered by one strip of any direction. It is not the case with *polygons* that generally need several strips to be covered. The strips associated with a polygon can be acquired from several successive illuminated half-revolutions.

Note that any strip can be acquired using either of the two associated opposite *azimuths* (azimuths range from 0 to 360 degrees). We call an *image* the association between a strip and an acquisition azimuth. Two potential images are thus associated with any strip.

Acquisition and transition constraints For each illuminated half-revolution h and for each candidate image i , the acquisition angular constraints allows us to determine whether or not i can be acquired from h and, in case of positive answer, the *earliest* and *latest* acquisition *starting time* of i . As the acquisition *speed* is constant, the acquisition *duration* of any image is proportional to its length.

For each illuminated half-revolution h and for each pair of candidate images i and j , a minimum *transition time* between the end of the acquisition of i and the beginning of the acquisition of j can be computed, taking into account the movement of the satellite on its orbit and its attitude manoeuvring capabilities. Note that this transition time depends on the time at which the transition begins, that is on the time at which the acquisition of i begins. Note also that the computation of this transition time implies itself to solve a complex *continuous* constrained optimization problem, that has no analytical solution and may be very time consuming, since the best algorithms in terms of solution quality may need a half hour of computing.

Energy consumption As satellite attitude manoeuvres are *energy* consuming and as this energy is limited on board, this limitation must be taken into account. Note that, because solar panels are firmly attached to the satellite, in order to limit vibrations and to increase agility, energy production and image acquisition may be conflicting tasks (the attitude positions needed for image acquisition may imply that the solar panels are no more well oriented towards the sun).

Data recording and downloading Images must be not only acquired. The resulting data must be *recorded* on board and *downloaded* to any appropriate station on the ground. Consequently, the limitation of the *on board recorders*, the *visibility windows* between the satellite and the stations on the ground, and the limitation of the *data flow* between the satellite and the ground must be taken into account too. Note also a possible conflict between data downloading and image acquisition.

Acquisition uncertainties Because of the optical nature of the instrument, the presence of *clouds* can decrease the quality of an acquired image and even invalidate it. As an absence of clouds over a given area cannot be guaranteed a long time in advance, it is never sure that a planned image acquisition will be successful.

Optimization criterion Although other criteria could be meaningful, the chosen criterion is the *sum* (or the *expected sum* to take into account uncertainties) of the *gains* associated with the *satisfied requests*, that is an *utilitarianist* criterion. In

a first time, it can be considered that the gain associated with a satisfied request equals its *weight*. But, whereas *spot* acquisition requests are either satisfied or not, *polygon* acquisition requests may be only partially satisfied. Consequently, two criteria have been considered:

- a first, called *linear*, where the gain associated with a completely satisfied request equals its weight and where the one associated with a partially satisfied request is proportional to the useful acquired surface;
- a second, called *non linear*, where the gain associated with a completely satisfied request is the same, but the one associated with a partially satisfied request is the result of the application of a *convex* function to the useful acquired surface.

The advantage of the non linear criterion is to favour the *termination* of already partially acquired polygons.

Mission management organization It is assumed that the selection and the scheduling of the images that will be acquired by the satellite is done *each day* for the following day, taking into account the current *set of user requests*, the current *state of the satellite*, and the currently available *meteorological forecasts*.

As each illuminated half-revolution defines a nearly independent subproblem, we consider that the basic problem to solve is a selection and scheduling problem on *one illuminated half-revolution*.

Selection and scheduling are done *on the ground*, under the supervision of human operators. When an acquisition plan has been built, the associated set of commands is uploaded to the satellite. When this plan has been executed, the associated data are analyzed by human operators and the strips associated with validated images are withdrawn from the set of user requests.

This kind of organization can be characterized as a regular *off-line on the ground* mission management organization. Others *on-line*, eventually *on board*, more *reactive* organizations could be considered, but are out of the scope of this paper.

3 Problem simplifications

In order to get a manageable problem, we must simplify substantially the previously described problem. The successive simplifications we did are the following.

Image acquisition degrees of freedom In addition to the assumption of a fixed strip *width*, we made the assumption of a fixed *direction*. Such an assumption may seem strange in the context of an agile satellite, because it removes in fact one of the three degrees of freedom. It is however justified by the results of simulations which showed that the satellite attitude movements around the yaw axis, required to vary the acquisition direction, are very costly in terms of transition time and are not compensated by a greater freedom of acquisition of either spots or polygons. This fixed direction can be however freely chosen.

From requests to images As a consequence of the previous assumption of a fixed acquisition direction, all the *spots* are acquired using this direction and all the *polygons* are cut up along the same direction. For each polygon, this cutting up is performed once and for all before selection and scheduling and an *offset* is chosen such that the useless acquired surface is minimized.

Acquisition and transition constraints We assume that the transition time between two image acquisitions does not depend on the time at which the transition begins. Moreover, in order to bypass the complexity of the computing of this minimum transition time, we pre-compute a *table* of minimum transition times using a reasonable discretization of the parameter space, that we exploit using simple *linear interpolations*.

Energy consumption, data recording and downloading For the moment, we do not consider the constraints related to the energy, memory, visibility, and data flow limitations.

Acquisition uncertainties In order to take into account the acquisition uncertainties, as well as the remaining acquisition opportunities from other satellite revolutions, we use an approach inspired from [15], which defines a rational way of modifying the weight that is associated with each request and used by the selection and scheduling process. Roughly speaking, this modification favours the requests the acquisition certainty of which is high from this revolution and the number of remaining acquisition opportunities from other revolutions is low.

Optimization criterion For the *non linear* criterion, we use a *piecewise linear* convex function.

4 Problem mathematical statement

The problem resulting from these simplifications, we call *SRSS* for *Satellite Revolution Selection and Scheduling*, can be mathematically stated as follows.

Data Let R be the set of *requests* that can be acquired, at least partially, from the considered illuminated half-revolution. For each $r \in R$, let W_r be its weight and A_r be its surface (multiplied by two in case of a stereoscopic request).

Let I be the set of potential *images*, associated with R . For each $i \in I$, let r_i be its associated request, E_i be its earliest starting time, L_i be its latest starting time, D_i be its duration, A_i be its useful surface, and $W_i = W_{r_i} \cdot \frac{A_i}{A_{r_i}}$ be its weight.

For each pair of images $(i, j) \in I \times I$, let M_{ij} be the minimum transition time between i and j . Let $B \subseteq I \times I$ be the set of *pairs of images* (i, j) , such that i and j are images of the same strip, using opposite azimuths. Let $S \subseteq I \times I$ be the set of *pairs of images* (i, j) , such that i and j are the two elements of a *stereoscopic* image of the same strip, using thus the same azimuth.

Decision variables We need three sets of *decision variables*: the first for the *selection*, the second for the *scheduling* of the selected images, and the third for the acquisition *starting times* of the selected images.

For each $i \in I$, let x_i be equal to 1 if the image i is selected, and to 0 otherwise. For each pair $(i, j) \in I \times I$, let f_{ij} be equal to 1 if the image i is followed by the image j in the chosen sequence, and to 0 otherwise. For each $i \in I$, let t_i be the starting time of the the image i , if it is selected.

Constraints Let o be a fictitious image, used to begin and end the chosen sequence, and $I^+ = I \cup \{o\}$.

The constraints that define the feasible selections and sequences are the following:

$$\forall i \in I : (x_i = 1) \Rightarrow (E_i \leq t_i \leq L_i) \quad (1)$$

$$\forall (i, j) \in I \times I : (f_{ij} = 1) \Rightarrow (t_i + D_i + M_{ij} \leq t_j) \quad (2)$$

$$\forall (i, j) \in B : x_i + x_j \leq 1 \quad (3)$$

$$\forall (i, j) \in S : x_i = x_j \quad (4)$$

$$x_o = 1 \quad (5)$$

$$\forall i \in I^+ : \sum_{j \in I^+} f_{ij} = \sum_{j \in I^+} f_{ji} = x_i \quad (6)$$

The constraints 1 and 2 are *temporal constraints*, associated with the acquisition angular constraints and the minimum transition time constraints. The constraints 3 state that only one image per strip is needed. The constraints 4 state that the two elements of a stereoscopic image are needed. The constraints 6 state that the variables x_i and f_{ij} actually define a sequence of selected images.

Criterion Whereas the *linear* criterion Q_l can be defined as follows:

$$Q_l = \sum_{i \in I} W_i \cdot x_i = \sum_{i \in I} W_{r_i} \cdot \frac{A_i}{A_{r_i}} \cdot x_i \quad (7)$$

the *non linear* criterion Q_{nl} can be defined as follows:

$$Q_{nl} = \sum_{r \in R} W_r \cdot P\left(\sum_{i \in I | r_i = r} \frac{A_i}{A_r} \cdot x_i\right) \quad (8)$$

where P is a *piecewise linear* convex function, defined on $[0, 1]$ and such that $P(0) = 0$ and $P(1) = 1$. Note that both criteria are equivalent when $\forall x \in [0, 1], P(x) = x$.

Problem analysis Apart from the constraints 3 and 4, and the non linear criterion, *SRSS* has the classic form of a *selection* and *scheduling* problem. In fact, it is close to well known problems like:

- the *Traveling Salesman* problem [6, 10], at which temporal constraints would be added and where the goal would be no more to visit all the cities by minimizing the travel distance, but to maximize the sum of the weights of the visited cities;

- the *Job Shop* and *Open Shop Scheduling* problems [6], where the goal would be no more to complete all the jobs in a minimum time, but to maximize the sum of the weights of the completed jobs;
- the *Knapsack* problem [6], where the usual linear capacity constraints would be replaced by temporal constraints.

It can be established that, like these problems, *SRSS* is *NP-hard*. This implies in practice that any algorithm able to solve it to optimality may need in the worst case a computation time that grows exponentially with the size of the instance to be solved.

It may be interesting to look at it as the combination of three subproblems: *selection*, *scheduling*, and *temporal assignment*. Indeed, whereas the *selection* and *scheduling* subproblems are hard, the *temporal assignment* subproblem, that is the problem of deciding if a specified sequence of images can be achieved or not, is *polynomial* and can be solved by a simple *propagation* on the earliest and latest starting times associated with each image (in fact, by enforcing *arc consistency*). This observation will be used by the *local search* algorithm (see Section 5.4). Note also that the optimization criterion only depends on the *selection* choices, and does not depend on the *scheduling* and *temporal assignment* choices.

It can be also noted that, provided that the *time* has been *discretized*, a *weighted acyclic directed graph* can be associated with any instance of *SRSS*. In this graph, a *vertex* is associated with any pair (i, t) , where $i \in I$ is a candidate image and t a possible acquisition starting time for i ($E_i \leq t \leq L_i$, equations 1). A *directed edge* exists between two vertices (i, t) and (j, t') iff $i \neq j$ and the acquisition of i starting at time t can be followed by the acquisition of j starting at time t' ($t + D_i + M_{ij} \leq t'$, equations 2). This temporal constraint prevents the presence of cycles. The *weight* associated with each directed edge is the weight W_i of the image i associated with its origin. Assuming a *linear* optimization criterion (equation 7), looking for an optimal solution for an *SRSS* instance is equivalent to looking for a *longest path* in the associated graph, that does not involve two vertices associated with the same image (equations 3), and that involves the two vertices associated with the two elements of a stereoscopic image each time it involves one of them (equations 4). This observation will be used by the *dynamic programming* algorithm (see Section 5.2).

5 Four solving algorithms

First, it can be observed that, in case of a *linear* optimization criterion, the problem mathematical statement presented in the previous section defines a *mixed integer programming* problem, that suggests the use of dedicated tools. Unfortunately the use of *CPLEX*, one of the most powerful *Integer Programming* tools, provided us with poor results : only very small instances (no more than twenty candidate images) could be deal with. So, this way has been given up. Four other ways have been then explored :

- a *greedy* algorithm (*GA*);

- a *dynamic programming* algorithm (*DPA*);
- a *constraint programming* approach (*CPA*);
- a *local search* algorithm (*LSA*).

The first two (*GA* and *DPA*) are limited to a *linear* optimization criterion (equation 7) and do not take into account the *stereoscopic* acquisition constraints (equations 4). The last two (*CPA* and *LSA*) are not limited and take into account the whole set of constraints.

5.1 A greedy algorithm

The *greedy* algorithm we considered imitates the behavior of an on-line mission management system that, in parallel with image acquisition, would be wondering what next image to acquire. It starts with an empty sequence of images. At each step, it chooses an image to be added at the end of the current sequence and repeats this until no image can be added.

At each step, the chosen image is one of the images that is not present in the current sequence yet, can follow the last image of the current sequence, and maximizes a criterion that is an *approximation* of the gain that is possible to get by making this choice. When the chosen image is added at the end of the current sequence, the temporal constraints are propagated.

If \hat{G} is an approximation of the problem optimum, $E = \min_{i \in I} E_i$ and $L = \max_{i \in I} L_i$, and T_i is the earliest ending time of i , if it would be added at the end of the current sequence, the chosen criterion is:

$$W_i + \hat{G} \cdot \frac{L - T_i}{L - E} \quad (9)$$

The first part of the criterion measures the immediate gain resulting of the choice of i . The second part is approximation of the gain that it would be possible to obtain later. This later gain is assumed to be proportional to the remaining time. As the problem optimum is not known, it is possible to start with any approximation, to run the greedy algorithm, to use its result as a better approximation, and so on.

A non linear optimization criterion (equation 8), as well as stereoscopic acquisition constraints (equations 4), which both link images that are set anywhere in the sequence, cannot be easily taken into account by such a sequential decision process.

5.2 A dynamic programming algorithm

The *dynamic programming* algorithm uses the observation, done in Section 4, of the possible transformation of *SRSS* into a *longest path* problem in a *weighted acyclic directed graph*, obtained thanks to a *time discretization* and under the assumption of a *linear* optimization criterion. But, to obtain a purely longest path problem, *polynomially* solvable, it is necessary to remove the constraints 3 and 4.

Assuming that the stereoscopic acquisition constraints 4 have been anyway removed, a way of removing the constraints 3 consists in *ordering* the set of candidate images and in imposing that the chosen sequence respects this order, which comes down to deciding about the *scheduling* subproblem. Indeed, if we remove now from the graph all the edges the destination vertex of which precedes its origin vertex in the chosen order, the constraints 3 will be necessarily met by any path.

In the general case, it may be difficult to find a pertinent order. But, *natural* orders may be exhibited in our specific problem: either a *temporal* order according to the middle of the temporal window associated with any image, or a *geographical* order according to the latitude of the middle of the strip associated with any image. In both cases, the idea is to prevent the satellite to turn its instrument backwards thanks to its attitude control system while going forwards on its orbit, because this kind of movement may be considered to be generally inefficient.

The *dynamic programming* algorithm we designed is only an efficient way of looking for such a longest path. It explores the images in the inverse order of the chosen order, and the possible starting times in the inverse order of the natural order. For each pair (i, t) , it computes the maximum gain $G^*(i, t)$ that it is possible to obtain by acquiring i and starting this acquisition at time t . For that, it uses the following equation [2]:

$$G^*(i, t) = \max_{(j, t') | c(i, t, j, t')} [W_i + G^*(j, t')] \quad (10)$$

where $c(i, t, j, t')$ holds iff there is an edge between (i, t) and (j, t') , that is iff $i \neq j$ and the acquisition of i starting at time t can be followed by the acquisition of j starting at time t' ($t + D_i + M_{ij} \leq t'$, equations 2). Doing that, it records the pair (j, t') (in fact, one of these) associated with $G^*(i, t)$. Moreover, it takes advantage of the following monotonicity property:

$$\forall i, t, t' : t < t' \Rightarrow G^*(i, t') \leq G^*(i, t) \quad (11)$$

which states that starting later cannot improve the gain.

As the *greedy* algorithm and for the same reasons, this *dynamic programming* algorithm can easily take into account, neither the non linear optimization criterion (equation 8), nor the stereoscopic acquisition constraints (equations 4). For example, taking into account stereoscopic acquisition constraints would induce time and memory requirements growing exponentially with the number of stereoscopic images.

5.3 A constraint programming approach

Constraint programming is neither an algorithm nor a family of algorithms. It is first a *modelling framework*, which uses the basic notions of *variables* and *constraints* and to which many various generic algorithms can be applied.

For solving our problem, we could have used any basic *constraint reasoning* or *constraint programming* tool, provided by software companies, research centers,

or academic teams, like our own tools². We decided to use *OPL* [8], firstly because it is a nice high level *modelling* tool, and secondly because it can call and combine *constraint programming* and *integer linear programming*.

OPL allowed us to build various models of *SRSS*, all of them more compact than the linear one described in Section 4. The model we finally chose deals with a restriction of *SRSS*, which consists in finding a feasible optimal sequence of images of a *fixed length*. We start with a length equal to 2 and increase this length at each step, until no feasible sequence can be found. The largest optimal found sequence is an optimal solution of *SRSS*.

Unfortunately, even with this approach, the first results, obtained within a limited time, were very poor in terms of quality. Neither the use of pertinent heuristics for the variable and value orderings, nor the use of non standard search strategies like *Limited Discrepancy* [7], improve them significantly.

The only way we found to obtain better results with this approach was to add *constraints* that are not redundant, and thus may decrease the problem optimum, but are chosen in such a way that we can hope that the loss in terms of quality will not be too high. The constraints we added are the following :

- images the *weight* of which is too low are removed from the set of candidate images;
- each image is constrained to appear only in a specified *sub-sequence* of the whole sequence; for example, an image the associated strip of which is located near the equator will not appear at the beginning of the sequence;
- although the considered sequences can follow an *order* that is different from the natural temporal or geographical order (discussed in Section 5.2 and used by the *dynamic programming* algorithm), the amplitude of a *backtrack* with respect to this order is limited;
- at each step of the algorithm, the considered sequences are constrained to involve all the images that are involved in the sequence that has been chosen at the previous step (not necessarily in the same order).

Adding these constraints allows us to obtain reasonable quality results on all the instances whatever their size.

5.4 A local search algorithm

Local search algorithms, like *hill-climbing search*, *simulated annealing*, *tabu search*, or *genetic algorithms* [1], are known to be applicable each time one wants to find within a limited time reasonable quality solutions to large constrained optimization problems.

Rather than using generic algorithms, we designed a simple *specific* algorithm dedicated to our problem. This algorithm defines a local search through the set of the *feasible* sequences of images. It starts with an empty sequence and stops when a specified time limit is reached.

² See <ftp://ftp.cert.fr/pub/lemaitre/LVCSP/>.

At each step, it chooses one action among two possible ones: either to *add* an image to the current sequence, or to *remove* an image from it. The choice between both these actions is random and made according to a dynamically evolving probability. The result of an image adding may be either a success or a failure. In case of success (resp. failure), the adding probability is increased (resp. decreased). On the other hand, an image removal is always successful and does not modify the adding probability.

In both cases (adding or removal), an image is chosen to be added or removed. This choice is random, with a probability to be added (resp. removed) that is proportional (resp. inversely proportional) to its *weight*. In case of image adding, the choice of the position in the sequence is random, with a uniform probability among all the alternatives.

To determine if adding an image at a specified position is possible or not, and to update the time windows associated with each image in the current sequence when adding or removing an image, temporal constraint propagation mechanisms are used (see Section 4).

6 Experimental results

We compared the performances of these four approaches, by running the associated algorithms on six instances we chose among a set of training instances provided by the *CNES*, as being representative of this set.

For each instance and each algorithm, the computation time was limited to two minutes, except for *LSA* that was running one hundred times, two minutes each time, because of its stochastic behavior. Within this time, *GA* and *DPA* terminated, *CPA* was stopped before termination, and *LSA*, which cannot terminate naturally, simply stopped after two minutes. Results were compared in terms of quality (quality of the best solution found after two minutes).

A first experiment, involving the four algorithms (*GA*, *DPA*, *CPA*, and *LSA*), was carried out. In this experiment, the optimization criterion was linear and the stereoscopic requests dealt with as if they were unrelated (the stereoscopic constraints 4 were ignored). Results are presented in Table 1. Despite of its restriction to a predefined image sequencing, *DPA* produces systematically the best results.

Unfortunately, the best two algorithms from this first experiment (*DPA* and *GA*) cannot deal with a non linear optimization criterion and with stereoscopic constraints.

A second experiment, involving only the two other algorithms (*CPA* and *LSA*), was carried out. In this experiment, the optimization criterion was non linear and the stereoscopic requests correctly dealt with. Results are presented in Table 2. *LSA* produces systematically the best results.

In both tables, a row is associated with each instance. The instance number appears in the first column, the number of involved strips in the second column, the results, in terms of quality, provided by *GA*, *DPA*, *CPA*, and *LSA*, in the last

four columns. For *LSA*, average and maximum results over the hundred trials are provided. For each instance, the best results are displayed in bold.

instance id	# strips	GA	DPA	CPA	LSA av. (max.)
2:13_111	106	532	603	442	574 (587)
2:15_170	295	707	843	527	723 (779)
2:26_96	483	831	1022	782	826 (877)
2:27_22	534	895	1028	777	800 (861)
3:25_22	342	436	482	253	345 (375)
4:17_186	147	188	204	177	192 (196)

Table 1. First experiment: linear optimization criterion, stereoscopic constraints ignored.

instance id	# strips	CPA	LSA av. (max.)
2:13_111	106	241	414 (490)
2:15_170	295	350	446 (490)
2:26_96	483	439	516 (592)
2:27_22	534	410	455 (561)
3:25_22	342	149	255 (298)
4:17_186	147	125	145 (156)

Table 2. Second experiment: non linear optimization criterion, stereoscopic constraints dealt with.

7 Lessons

We conclude with some lessons we drew from this study and choose to present along the four considered algorithms. It is however important to stress that, because many mistakes may be done while modelling a problem, designing and implementing an algorithm, using a tool, carrying out experiments . . . these lessons cannot be considered as being universal and definitive truths. They are presented here to stimulate discussions in the *constraint reasoning* and *constraint programming* community.

Greedy algorithm It is confirmed that *greedy* algorithms are always the first available solution when facing a large complex constrained optimization problem. They are easy to implement, generally require little computation time, and produce reasonable quality solutions. The one we considered can be seen as a degraded version of the *dynamic programming* algorithm. But other greedy algorithms could have been considered, based on other *variable* and *value heuristics*.

Dynamic programming algorithm When applicable, that is when the number p of subproblems to consider is not too high, *dynamic programming* is clearly the best solution. It is easy to implement, requires a computation time and a memory that are proportional to p , and produces optimal solutions. As shown in [5], its applicability depends widely on the structure of a graph associated with

each problem instance (the *induced width* of the *macro-structure* graph in the *CSP* framework). It is however important to note that, if p grows exponentially with the problem size, both computation time and memory requirements of *dynamic programming* grow the same way. To bypass this difficulty, hybridizations between *dynamic programming* and *tree search*, as it is for example proposed in [9], deserve certainly more attention in the *constraint reasoning* community.

Constraint programming approach *Constraint programming* offers clearly very nice modelling frameworks: various types of constraints can be expressed in an elegant way, various models of a problem can be explored by adding or removing constraints. Difficulties arise with the solving methods, that are currently limited to *constraint propagation* and *tree search*.

For our problem, *local constraint propagation* mechanisms are clearly not powerful enough. We think that there are at least two reasons for that: firstly, although powerful specific propagation rules are available for *scheduling* problems, these rules are not applicable as long as *selection* decisions have not been made; the same phenomenon occurs when one goes from the *CSP* framework to the *Max-CSP* framework: basic *arc consistency* algorithms do not work anymore [12]; secondly, even when these *selection* decisions have been made, the time windows associated with each image are too large with regard to the duration of each image to allow propagation mechanisms to deduce any *scheduling* constraint. As it is well known, *depth-first tree search* mechanisms do not succeed to improve quickly the first *greedy* solution and exhibit a poor *anytime* behavior.

On the other hand, adding constraints to the problem statement, allowed us to obtain reasonable quality results. It is known in the constraint community, that adding *redundant* constraints, that is constraints that are satisfied in all the problem solutions, helps the search (this is what is done by the *constraint propagation* mechanisms). In constrained optimization problems, an interesting way of helping the search consists in adding *non redundant* constraints, that is constraints that are not satisfied in all the problem solutions, but in all the optimal solutions, or at least in some of them, and thus do not decrease the problem optimum, or decrease it at least as possible. It is in fact what has been done with success with the *dynamic programming* algorithm: discretizing the time and adding sequencing constraints result in that case in a *polynomial* problem, solvable by a *dynamic programming* approach.

Local search algorithm *Local search* mechanisms are widely applicable, because they only require the ability to evaluate any complete solution. The results we obtained with a very simple search strategy confirm the significance of a search through the set of the feasible solutions and of a combination between heuristic and random movements. The stochastic behavior of the resulting algorithms is always irritating and the numerous parameters difficult to tune. Hybridizations between *local search*, *limited tree search* and *constraint propagation*, as it has been for example proposed in [13, 11], is certainly one of the currently most promising ways of research.

These lessons may seem to be negative for the constraint programming approach, because the basic constraint programming tools we used did not provide us with actually satisfactory results. It is true if constraint programming is seen as limited to the combination between constraint propagation and tree search. But it is not true if it is seen as a powerful modelling framework, as well as a modular software architecture, to which many either specialized or generic algorithms, coming from the *Constraint Reasoning*, *Interval Analysis*, *Graph Theory*, *Artificial Intelligence*, or *Operations Research* communities, can be connected.

Acknowledgments We thank Jean-Michel Lachiver and Nicolas Bataille from CNES for their confidence, and Frank Jouhaud, Roger Mampey, Jérôme Guyon, and Mathieu Derrey from ONERA for their participation in this study.

References

1. E. Aarts and J. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.
2. R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
3. E. Bensana, M. Lemaître, and G. Verfaillie. Earth Observation Satellite Management. *Constraints : An International Journal*, 4(3):293–299, 1999.
4. E. Bensana, G. Verfaillie, J.C. Agnèse, N. Bataille, and D. Blumstein. Exact and Approximate Methods for the Daily Management of an Earth Observation Satellite. In *Proc. of SpaceOps-96*, München, Germany, 1996.
5. R. Dechter. Bucket Elimination: a Unifying Framework for Reasoning. *Artificial Intelligence*, 113:41–85, 1999.
6. M. Garey and D. Johnson. *Computers and Intractability : A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.
7. W. Harvey and M. Ginsberg. Limited Discrepancy Search. In *Proc. of IJCAI-95*, pages 607–613, Montréal, Canada, 1995.
8. P. Van Hentenryck. *The OPL Optimization Programming Language*. MIT Press, 1999.
9. J. Larrosa. Boosting Search with Variable Elimination. In *Proc. of CP-00*, Singapore, 2000.
10. E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, editors. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.
11. L. Lobjois, M. Lemaître, and G. Verfaillie. Large Neighbourhood Search using Constraint Propagation and Greedy Reconstruction for Valued CSP Resolution. In *Proc. of the ECAI-00 Workshop on "Modelling and Solving with Constraints"*, Berlin, Germany, 2000.
12. T. Schiex. Arc Consistency for Soft Constraints. In *Proc. of CP-00*, Singapore, 2000.
13. P. Shaw. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In *Proc. of CP-98*, pages 417–431, Pisa, Italia, 1998.
14. M. Vasquez and J.K. Hao. A Logic-constrained Knapsack Formulation and a Tabu Algorithm for the Daily Photograph Scheduling of an Earth Observation Satellite. *To appear in the Journal of Computational Optimization and Applications*, 2001.
15. G. Verfaillie, E. Bensana, C. Michelon-Edery, and N. Bataille. Dealing with Uncertainty when Managing an Earth Observation Satellite. In *Proc. of i-SAIRAS-99*, pages 205–207, Noordwijk, The Netherlands, 1999.