# Bounding the Optimum for the Problem of Scheduling the Photographs of an Agile Earth Observing Satellite

Djamal Habet, Michel Vasquez, Yannick Vimont

# Bounding the optimum for the problem of scheduling the photographs of an Agile Earth Observing Satellite

**Djamal Habet[1] · Michel Vasquez[2] · Yannick Vimont[2]**
[1] LSIS - UMR CNRS 6168, Domaine Universitaire de Saint-Jérôme, Avenue Escadrille Normandie-Niemen, 13397 Marseille cedex 20, France
[2] École des Mines d'Alès, LGI2P Research Center, Site EERIE, Parc scientifique Georges Besse,30035 Nîmes cedex 01, France

**Abstract** The problem of managing an *Agile* Earth Observing Satellite consists of selecting and scheduling a subset of photographs among a set of candidate ones that satisfy imperative constraints and maximize a gain function. We propose a tabu search algorithm to solve this NP-hard problem. This one is formulated as a constrained optimization problem and involves stereoscopic and time window visibility constraints; and a convex evaluation function that increases its hardness. To obtain a wide-ranging and an efficient exploration of the search space, we sample it by consistent and saturated configurations. Our algorithm is also hybridized with a systematic search that uses partial enumerations. To increase the solution quality, we introduce and solve a secondary problem; the minimization of the sum of the transition durations between the acquisitions. Upper bounds are also calculated by a dynamic programming algorithm on a relaxed problem. The obtained results show the efficiency of our approach.

**Keywords** Tabu search · Consistent neighborhood · Scheduling · Agile satellite · Dynamic programming

D. Habet (✉)
e-mail: Djamal.Habet@lsis.org

M. Vasquez
e-mail: Michel.Vasquez@ema.fr

Y. Vimont
e-mail: Yannick.Vimont@ema.fr

## 1 Introduction

The mission of an Earth Observing Satellite (EOS) is to acquire photographs on the Earth's surface, in response to observation requests. The management problem of an EOS is to select and to schedule a subset of weighted images among a set of candidate ones which must satisfy imperative constraints and at the same time maximize a given profit function.

The new generation of EOS, like those studied in the French PLEIADES project, are *Agile* Earth Observing Satellites (AEOS). This means that, while the single on-board camera remains fixed on the satellite, the whole satellite is mobile along three axes (roll, pitch and yaw). This mobility potentially increases the efficiency of the whole system. Indeed, the number of ways of acquiring a given area of the Earth's surface is potentially infinite, since the azimuth and the starting time of a given image acquisition are free. However, managing an AEOS becomes significantly more difficult because the search space is considerably larger.

Both EOS and AEOS management problems are NP-hard [3, 15] and several methods have been proposed to tackle them. Four methods for solving the AEOS management problem were investigated [15]: a fast greedy algorithm, a dynamic programming algorithm, a constraint programming method, and a local search algorithm based on the insertion and removal of images in a schedule. The selection and scheduling problem for a non-agile satellite, like SPOT5, is stated in [3] and various methods have been used to solve it. The best results are obtained in [20] by using a tabu search algorithm and very good upper bounds are calculated by an original partition method [21]. A scheduling problem involving a satellite equipped with a radar instrument, very agile on the pitch axis but slow on the roll axis, is stated and solved in [12] by a partial enumeration algorithm. We will present later more works related to the subject of our interest.

In this paper, we propose a tabu search algorithm to solve the problem of selecting and scheduling the photographs of an AEOS. The search space is based on consistent and possibly saturated configurations. On the one hand, we ensure the consistency of the configurations by propagating efficaciously the problem constraints. In the other hand, the saturation characterizes the optimal solutions. It deals with the fact that it is not possible to improve the quality of a configuration by the selection of a new image and at the same time without the removal of some already selected images. The saturation will be formally defined in Sect. 4.2. During the search, a partial enumeration algorithm is used to solve a large number of decisional problems. Furthermore, we introduce a secondary problem which is the minimization of the sum of the transition durations between two image acquisitions. This problem is handled by a second tabu search algorithm based on exchanging the image orders and also on the inversion of the acquisition direction of the images. Its resolution allows to the first tabu algorithm to improve the quality of the solutions that it produces as it will be explained later. Moreover, we calculate upper bounds by means of a dynamic programming algorithm for a relaxed problem and a linearized objective function. This algorithm is inspired by the work presented in [14] on the Vehicle Routing Problem with Time Windows (VRPTW).

The paper is structured as follows. It starts with a description of the AEOS management problem in Sect. 2. We report the related works on several satellite management problems in Sect. 3. The components of our tabu search algorithm are described in Sect. 4. The secondary optimization problem is handled in Sect. 5 and the computation method of the upper bounds is explained in Sect. 6. The obtained results are presented and compared in Sect. 7 and Sect. 8 concludes the paper.

## 2 AEOS problem description

Earth observing satellites are platforms that orbit the planet and are equipped with optical instruments. Over a period of several days they perform a cycle of orbits of the Earth. Each orbit is phased out with respect to the preceding one but the trajectory is cyclic in the sense that the satellite recovers its initial position after a number of orbits. A full cycle enables the satellite to view each area of the planet. In the course of their mission, satellites take photographs of specific areas of the Earth in function of requests from a number of users including governments, research institutes, and companies. Each request generates a profit or a gain. Typically, the number of requests exceeds what can be feasibly be accomplished during a mission. The problem is to select and schedule a subset of requests yielding a maximal gain, subject to operational constraints. The present work focuses on maximizing the profit associated with a single orbit of the satellite.

A request can either be a target or a polygon. A target consists of a single strip (rectangular shape) whereas a polygon may cover a wide geographical area. Because of their size, polygons cannot usually be photographed in a single shot. For this reason, they are partitioned into strips of equal width but possibly unequal lengths and the time required to acquire a strip is proportional to its length. In the course of a single orbit, the satellite may be able to photograph several strips of the same polygon. Because of its agility, it is also able to acquire a strip using two opposite azimuths (direct and indirect) according to the satellite rotation sense. Hence, two shots (or images) per strip are possibles. The starting date of the acquisition of each shot must be in accordance with its visibility time window. Moreover, some requests are mono while others are stereo. A mono request consists of a single shot of each strip in the polygon. A stereo request consists of two shots of each strip at different angles but in the same direction. A strip from a stereo request is considered to have been acquired only if its twin strip has also been acquired. Finally, for each pair of shots, the satellite requires a minimum transition time to maneuver the camera from the end of the first strip to the start of the second one. Readers interested in a full description of the the AEOS management problem can refer to [15].

The input of an AEOS management problem is a set of candidate strips, from the current set of requests, that could be acquired. The problem to be dealt with is twofold: to select a set of strip acquisitions that maximize the total gain, and to order them in time. Formally, the problem can be described as follows [1]:

### 2.1 Data

A problem with $n$ strips involves $2n$ possible acquisitions since for each strip $i$ two shooting directions are possibles. These shots are numbered $2i - 1$ (an odd number

for a shot acquired in the direct azimuth) and $2i$ (an even number for a shot acquired in the indirect azimuth). For each strip $i \in [1,\ n]$ let:

- $tw(i)$ be the index of its stereo twin strip, 0 if $i$ is mono.
- $d(i)$ be its shooting duration.
- $su(i)$ be its corresponding surface.

For each shot (image) $j \in [1,\ 2n]$ let:

- $es(j)$ and $ls(j)$ be its earliest and latest start dates respectively.
- $ee(j)$ and $le(j)$ be its earliest and latest end dates respectively.

For each shot pair $(i,\ j), i \neq j \in [1,\ 2n]$, $t(i \rightarrow j)$ denotes the minimum transition time between the end of $i$ to the beginning of $j$. We assume that $t(i \rightarrow j) \geq es(j) - le(i)$ (otherwise it is obviously underestimated). Four variables are associated to each shot $i \in [1,\ 2n]$:

- $x_i \in \{0,\ 1\}$ equals 1 if and only if shot $i$ is selected.
- $y_{0 \rightarrow i} \in \{0,\ 1\}$ equals 1 if and only if shot $i$ is the first of the selection.
- $y_{i \rightarrow 2n+1} \in \{0,\ 1\}$ equals 1 if and only if shot $i$ is the last of the selection.
- $t_i$ is the shooting start date of $i$. The value of this variable is irrelevant when $x_i = 0$. The $t_i$ values allow to order and to schedule the shots in time.

Let $m$ be the number of polygons. The $k$th polygon is characterized by:

- $s(k)$ its total area surface.
- $g_k$ its gain when fully acquired.
- $p(k) \subset [1,\ 2n]$ the set of shots that it contains. We recall that each polygon is divided into a set of strips. Moreover, two shots are possibles per strip according to the sense of its acquisition.

Two continuous variables are associated to each polygon $k \in [1,\ m]$:

- $S_k \in [0,\ 1]$ is the percentage of the surface covered by the selected strips.
- $G_k \in [0,\ 1]$ is the corresponding percentage of the polygon gain.

Finally, one binary variable is defined for each pair of acquisitions $i \neq j \in [1\ ,\ 2n]$:

- $y_{i \rightarrow j} \in \{0,\ 1\}$ equals 1 if and only if the shot $j$ is immediately acquired after the shot $i$.

### 2.2 Constraints

The equations constraining these variables are listed below. The acquisition of the same strip in both directions is forbidden by (1). Equation (2) states the stereo constraints: simultaneous selection with identical direction. The time window for the starting time of a shot is imposed by (3). Shooting dates of consecutive images must respect minimum transition times (4). These last two equations correspond to the time constraints.

$$\forall j \in [1,\ n], \quad x_{2j-1} + x_{2j} \leq 1, \tag{1}$$

$$\forall j \in [1,\ n], \quad \text{if } tw(j) \neq 0 \text{ then } (x_{2j-1} = x_{2tw(j)-1} \text{ and } x_{2j} = x_{2tw(j)}), \tag{2}$$

$$\forall i \in [1,\ 2n], \quad \text{if } x_i = 1 \text{ then } t_i \in [es(i), ls(i)], \tag{3}$$

$$\forall i \neq j \in [1,\ 2n], \quad t_j - t_i \geq (d(i) + t(i \to j)) \cdot y_{i \to j} + (es(j) - ls(j)) \cdot (1 - y_{i \to j}). \tag{4}$$

We denote by $C4$ the set of shot pairs $(i,\ j)$ that satisfy the condition (4).

Furthermore, there is at most one first shot and one last shot (5) and each selected acquisition has exactly one predecessor and one successor (6).

$$\sum_{i \in [1,\ 2n]} y_{0 \to i} \leq 1 \quad \text{and} \quad \sum_{i \in [1,\ 2n]} y_{i \to 2n+1} \leq 1, \tag{5}$$

$$\forall i \in [1,\ 2n], \quad \sum_{\substack{j \in [0,\ 2n+1] \\ j \neq i}} y_{j \to i} = x_i = \sum_{\substack{j \in [0,\ 2n+1] \\ j \neq i}} y_{i \to j}. \tag{6}$$

In the rest of this paper, we assume that the constraints (5) and (6) are always satisfied.

### 2.3 Objective function

The criterion to maximize is a global gain $G$ defined by the sum of the gains associated to the complete or partial acquisition of each polygon $k$ and formulated by:

$$G = \sum_{k=1}^{m} g_k \times G_k \quad \text{such as:}$$

- $\forall k \in [1,\ m],\ S_k = \dfrac{1}{s(k)} \displaystyle\sum_{i \in p(k)} su(i) \cdot x_i.$
- $G_k = f(S_k)$.
- $f : [0,\ 1] \to [0,\ 1]$ is a non-linear function, piecewise linear and defined by the points $\{(0, 0),\ (0.4, 0.1),\ (0.7,\ 0.4),\ (1,\ 1)\}$.

In order to tackle quickly the problem of selecting and scheduling photographs for an AEOS, the local search methods seem highly appropriate and we chose a Tabu Search (TS) metaheuristic [10]. Our TS algorithm is inspired by a tabu resolution methodology, presented in [19], working on a consistent neighborhood. It is combined with a systematic resolution by the means of partial enumerations. Furthermore, a secondary optimization problem is introduced and solved by a second TS algorithm. Before describing the different phases of our resolution, let us in the next section give some works concerning the satellite management problems and nigher ones.

## 3 Related work

Compared to other optimization problems, the Earth Observing Satellite management problem has received a limited attention. For the AEOS problem and in the context of the ROADEF'2003 Challenge, various methods were proposed. In [17],

J.F. Cordeau and G. Laporte have proposed a tabu search algorithm which borrows from the Unified Tabu Search Algorithm [4] developed for the Vehicle Routing Problem with Time Windows (VRPTW). An important feature of their algorithm is the possibility of exploring infeasible solutions during the search by allowing the violation of the time window constraints. Thus, the value of a solution is defined by $f(s) = G(s) - \alpha w(s)$, where $w(s)$ is the total time window constraints. The parameter $\alpha$ is initially set at 1 and self-adjusts during the course of the search to allow a mixture of feasible and infeasible solutions. The moves switch between the insertion and the removal of mono and stereo shots with respect to all the constraints except the time window one which is relaxed. Moreover, the algorithm uses two diversification mechanisms. The first one is a continuous diversification scheme which introduces penalties on poor solutions. These penalties drive the search process toward the less explored regions of the search space whenever a local optimum is reached. The second diversification mechanism perturbs the solution under certain circumstances. If the best known solution has not improved after a certain number of iterations then the search stops and restarts from the best reached solution. However, this best solution is perturbed by removing a portion of randomly selected shots.

E.J. Kuipers [17] proposed a local search algorithm based on two stages. In the first one, the most promising solutions are constructed then further optimized in the second stage. These two parts use Simulated Annealing (SA) algorithms. The neighborhood is constructed in two steps. In the first one, 1, 2, 3 or 4 requests (or parts of requests) are removed from the current solution, and in the second step 1, 2, 3 or 4 requests (or parts of requests) are added to the solution resulting from the first one. The number of the added or removed requests (1, 2, 3 or 4) in both steps is randomly chosen. Moreover, a library is constructed to file the best ways of ordering a set of 5 strips in terms of transition durations. This library is updated at each new insertion and used by the two SA algorithms. Other additional schemes are added to attempt the speeding up of the optimization process of the first stage (for instance, a limitation on the number of the strips treated according to their gain). However, the second stage considers all the strips during the neighborhood construction.

In addition, a number of variants of the AEOS problem have been studied in the literature. The paper [11] presents the space mission problem which consists of selecting and scheduling a set of jobs on a single machine among a set of candidates jobs. Each candidate job has a fixed duration, a given time window and a weight. The aim is to select a feasible sequence of jobs that maximizes the sum of weights. The space mission problem is NP-hard and it is very close to the AEOS management problem with the following simplifications: null transition times, no unique strip acquisition and stereoscopic constraints, and a linear objective function. Several interesting greedy algorithms and an optimal algorithm for solving the space mission problem are proposed. 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 a lagrangian 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 concerns 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 the board of the satellite, but there are 3 instruments. Nevertheless, it is possible to pre-compute binary and ternary constraints which model the compatibilities between candidate images. Each candidate image is weighted and the problem is to find a feasible subset of the candidate images maximizing the sum of weights. This optimization problem is NP-hard and can be formulated in the general Valued Constraint Satisfaction Problem model [18]. The paper [3] fully describes the problem and proposes some instances benchmark. Some results with dedicated exact and approximate methods are given in [2, 3] and the column generation technique has been used to compute upper bounds on the benchmark instances [8]. Very good results on these instances were obtained by M. Vasquez and J.K. Hao using a dedicated Tabu Search algorithm [20]. These authors have also obtained very good upper bounds by means of an original partition method assessing the quality of their previous results [21]. Note that for some instances the optima values are still unknown.

The work presented in [9] concerns a selection and scheduling problem of a semi-agile satellite. The differences with respect to the AEOS problem are as follows: (a) the criterion to be maximized is the number of selected images (images are not weighted), (b) the satellite is slightly mobile in two axes (pitch and roll), but remains fixed during an image acquisition; so, there is only one possible azimuth for an acquisition, (c) the satellite's kinematics do not allow for a given strip to be acquired twice during the same track, and (d) there are no stereoscopic requests, hence no corresponding stereo constraints.

We now describe some related problems. In the first one which is named the Maximum Shot Sequencing Problem (MSP), the aim is to select and to 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. MSOP and MSP are also NP-hards and several algorithms are proposed and based on graph theoretic concepts. In a first approach, the time is discretized. The constraints (given by the satellite kinematics) are such that MSOP amounts to a longest path problem. 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 objective, 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 algorithm with graph-based heuristics and bounds. The approximate algorithm is a kind of greedy algorithm also based upon graph properties. Experiments are conducted on a set of randomly generated instances, allowing the proposed algorithms to be assessed against upper bounds and the impact of the discretization to be measured.

The paper [12] describes a scheduling problem concerning a satellite equipped with a radar instrument which is very agile on the pitch axis but slow on the roll axis. Only one azimuth is available for acquiring images. This problem is equivalent to the space mission problem [11] 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 [22] describes the so-called Window-Constrained Packing problem (WCP). It differs from the problems presented above in the fact that the evaluation function 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 to the observations which are well-placed inside their time window. There are no transition times between observations. This problem is similar to the space mission problem [11], with a particular gain function, depending on the starting times of the observations. The authors present some approximate algorithms for solving the WCP problem. A first one is a fast but not very accurate greedy algorithm. A second one is similar to the first, but it includes some look-ahead. It is a little better but has expensive computation times. These algorithms have been tested on randomly generated instances. Actually, the WCP problem is a simplified short-term version of the real scheduling problem. The paper investigates the associated mid-term and long-term scheduling problems and their connections, as well as different objective functions.

## 4 A Tabu Search algorithm for AEOS problem

In this section, we first review the the principles of the Tabu Search then we will describe point by point all the components of our tabu resolution.

### 4.1 Review of TS

A Tabu Search (TS) is a meta-heuristic designed to tackle hard combinatorial optimization problems. By contrast with random approaches, TS is based on the belief that an intelligent search should include more systematic forms of guidance based on adaptive memory and learning. TS can be described as a form of neighborhood search with a set of critical and complementary components. For a given optimization instance $(S, f)$ characterized by a search space $S$ and an objective function $f$, a neighborhood $\mathcal{N}$ is introduced. It associates to each configuration $s$ in $S$, a non-empty subset $\mathcal{N}(s)$ of $S$. A typical TS algorithm begins with an initial configuration $s_0 \in S$, then repeatedly visits a series of the best local configurations following the neighborhood function. At each iteration, one of the best neighbors $s' \in S$ is selected to become the current configuration, even if $s'$ does not improve the current configuration in terms of the cost function.

To avoid the problem of cycles occurring and to allow the search to go beyond local optima, a tabu list is introduced. This adds a short term memory component to the method. A tabu list maintains a selective history $H$ (short term memory), composed of previously encountered configurations or, more generally, pertinent attributes of such configurations. A simple TS strategy consists in preventing configurations of $H$ from being considered on the next $k$ iterations, called the tabu tenure. This can vary according to different attributes, and it is generally problem dependent. At each iteration, TS looks for the best neighbor from this dynamically modified neighborhood $\mathcal{N}(H, s)$, instead of $\mathcal{N}(s)$ itself. Such a strategy prevents the search

from being trapped in short term cycling and makes the process more rigorous. When attributes of configurations, instead of configurations themselves, are recorded in a tabu list, some unvisited, but nonetheless interesting configurations may be prevented from being considered. Aspiration criteria may be used to overcome this problem. A widely used aspiration criterion consists in removing a tabu status from a move when it leads to a configuration better than the best one obtained so far. Two other important ingredients of TS are intensification and diversification. On the one hand, the intensification consists in focusing the search to exploit regions of the search space, or characteristics of solutions, that the search history suggests that they are promising. On the other hand, the diversification undertakes to explore regions that differ in significant respects from regions previously visited.

Therefore, a TS is described by specifying its main elements: the configuration representation, the cost function to evaluate the configurations, the neighborhood function, the tabu list management, the aspiration criteria, and finally the intensification and diversification phases.

### 4.2 Search space definition

**Definition 1** The unconstrained search space $S$ consists of all the vectors of the pairs $(x_i, t_i)$, $i = 1, \ldots, 2n$:

$$S = \{((x_1, t_1), (x_2, t_2), \ldots, (x_{2n}, t_{2n}))/\ \forall i \in [1,\ 2n] :\ x_i \in \{0, 1\} \text{ and } t_i \in \mathbb{R}\}.$$

In this formulation, $i$ is a shot number among the $2 \times n$ possible ones issued from the $n$ strips (2 shots per strip). $t_i$ is the (unconstrained) beginning time acquisition of the shot $i$. The size of $S$ is highly huge and increases as the value of $n$ and the size of the $t_i$ intervals increase. Each element $s \in S$ is a configuration that does not satisfy necessary the problem constraints. However, a solution must satisfy the constraints defined by (1) to (4).

**Definition 2** The totally constrained search space $X$ is a subset of vectors in $S$ that satisfy all the imperative constraints (1) to (4):

$$X = \{s \in S/ \text{ all the elements of } s \text{ satisfy the constraints (1) to (4)}\}.$$

Each vector $s \in X$ is *a consistent configuration* (all the constraints are satisfied), which is evaluated by its corresponding gain $G(s) = \sum_{k=1}^{m} g_k \times f(\frac{1}{s(k)} \sum_{i \in p(k)} su(i) \cdot x_i)$ (see Sect. 2). We denote by $|s|$ the number of the shots that are selected in $s$ ($|s| = \sum_{i=1,\ldots,2n} x_i$). This constrained search space is used by our TS algorithm. However, it is equally possible for an algorithm to work with an intermediate search space where some constraints are relaxed. For this propose, we define a partially constrained search space $C$.

**Definition 3** The partially constrained search space $C$ is a subset of vectors of $S$ satisfying only the imperative constraints (3) and (4):

$$C = \{s \in S/ \text{ all the elements of } s \text{ satisfy the constraints (3) and (4)}\}.$$

This search space, where the stereo and uniqueness acquisition constraints are relaxed, will be used in order to compute upper bounds for the simplified problem.

**Definition 4** The saturated and the totally constrained search space $M$ is a subset of $X$ ($M \subseteq X$) such that:

$$M = \{s \in X \setminus \text{ for any } i \in [1,\ 2n] \text{ such as } x_i = 0 \text{ in } s, \text{ setting } x_i = 1 \text{ violates systematically some of the constraints (1) to (4)}\}.$$

In other words, each vector $s \in M$ is *a consistent and a saturated configuration* where no more images can be added without violating some imperative constraints. The principle here is that TS algorithm will explore the totally constrained search space $X$ and try to stay on the border of this saturated (promising) area. This saturation is one of the features of the optimal solution.

### 4.3 Consistent and saturated neighborhood

Now, we introduce the neighborhood function $\mathcal{N}$ over the totally constrained search space $X$. This function $\mathcal{N} : X \rightarrow (2^X - \emptyset)$ is defined as follows:

Let $s = ((x_1,\ t_1), (x_2,\ t_2), \ldots, (x_{2n},\ t_{2n}))$ be a consistent configuration, $s' = ((x'_1,\ t'_1),\ (x'_2,\ t'_2), \ldots, (x'_{2n},\ t'_{2n}))$ is a neighbor of $s$, i.e. $s' \in \mathcal{N}(s)$, if and only if the following conditions are checked:

1. $\exists! i \in [1,\ 2n] \setminus x_i = 0$ and $x'_i = 1$. Moreover, if $tw(i) \neq 0$ then $x_{tw(i)} = 0$ and $x'_{tw(i)} = 1$ (we try to insert exactly one shot in $s$, and its twin if it exists).
2. For any shot $i$ that satisfies the condition 1 and a shot $j \in [1,\ 2n]$ such as $i$ and $j$ are issued from the same strip, we have $x_j = x'_j = 0$. Moreover, if $tw(i) \neq 0$ then $x_k = x'_k = 0$ such as the shots $tw(i)$ and $k$ are also issued from the same twin strip. This condition forbids the acquisition of a strip (and its twin if it exists) in two directions and deals with the first constraint of the AEOS problem.
3. For any shot $i$ that satisfies the condition 1, we have $t_i \in [es(i),\ ls(i)]$. Moreover, if $tw(i) \neq 0$ then $t_{tw(i)} \in [es(tw(i)),\ ls(tw(i))]$. This condition deals with the time window visibility constraint.
4. For each shot $i$ that satisfies the condition 1, $\forall k \in [1,\ 2n]$ such that $(i,\ k) \notin C4$ and $x_k = 1$ we have $x'_k = 0$ and if $tw(k) \neq 0$ then $x'_{tw(k)} = 0$. Moreover, if $tw(i) \neq 0$ then $\forall l \in [1,\ 2n]$ such that$(tw(i),\ l) \notin C4$ and $x_l = 1$ then $x'_l = 0$. In addition, if $tw(l) \neq 0$ then $x'_{tw(l)} = 0$.
5. For any shot $i$ that satisfies the condition 1, $-3 \leq |s'| - |s| \leq 1$. Also, if $tw(i) \neq 0$ then $-6 \leq |s'| - |s| \leq 2$.
6. For any shot $i$ that satisfies the condition 1 and such that its insertion in $s$ requires at most the removal of two shots $j,\ k \in [1,\ 2n]$ with their twin shots if they exist ($-3 \leq |s'| - |s| \leq -1$, $x_j = x_k = 1$, $x'_j = x'_k = 0$ and $j \neq tw(k)$) then even $y_{j \rightarrow k} = 1$ or $y_{k \rightarrow j} = 1$ in $s$ ($k$ and $j$ are acquired one behind the other in $s$).

Thus, the neighborhood of $s$ is obtained by adding a free shot $i$ (not yet scheduled, $x_i = 0$) by flipping $x_i$ from 0 to 1 (condition 1), then removing some shots $k$

(by flipping $x_k$ from 1 to 0) to repair the violated constraints (condition 4). In fact, inserting a new shot may require to drop a certain number of the already fixed ones to maintain the consistency of constraints (1) to (4) described in Sect. 2.2. However, the condition 5 enforces a maximum of 2 shot removals if all the dropped shots are mono and 4 if they are stereo (in order to maintain the consistency of the stereo constraint, if a stereo shot is removed then its twin is removed too).

According to the conditions mentioned above, the best case is the insertion of a stereo shot and its twin without any removal ($|s'| - |s| = 2$). Oppositely to this case, the worst one corresponds to two insertions (stereo shot plus it twin: $+2$), such as each of these two insertions needs the removal of 2 stereo shots plus their twins: $-4$ ($|s'| - |s| = +2 - 4 - 4 = -6$). Moreover, this choice heuristic is also restricted to the removal of successive shots as described in the condition 6.

Hence, each configuration reached from $s$ and according to the condition 1 to 6 is also consistent and may be saturated (it is impossible to insert a new shot without any removal). Consequently, the elaborated neighborhood $\mathcal{N}(s)$ is *consistent and can be saturated*.

#### *4.3.1 Partial enumerations for the neighborhood evaluation*

We evaluate $\mathcal{N}(s)$ according to the gain criterion. For this purpose, consider a configuration $s = ((x_1, t_1), (x_2, t_2), \ldots, (x_{2n}, t_{2n}))$ where $|s|$ images are selected and an image $j$ such that $x_j = 0$ (i.e. $j$ is not yet selected). The shot $j$ can be inserted in $s$ through $|s| + 1$ positions: before the first shot, after the last shot, or between two successive shots on the schedule $s$. Hence, the insertion of the shot $j$ in each of the $|s| + 1$ positions is tested by allowing successive image removals (as explained above). If a position is tested positively then a neighborhood configuration $s'$ is reached by inserting $j$ at this position is $s$ (and dropping some others images, if necessary). We evaluate $s'$ by computing its corresponding gain value $G(s')$ (see Sect. 2.3). Among all the feasible insertions of $j$, the one that maximizes the gain value is selected. Consequently, at each step of the TS algorithm, we solve the decision problem of finding the best insertion position for each free shot in $s$ according to the gain function. These insertion tests correspond to *partial enumerations* under $s$.

#### *4.3.2 Incremental evaluation of the neighborhood*

TS algorithm uses an aggressive search strategy to exploit its neighborhood, i.e. at each iteration, it examines the value of $G(s')$ for each candidate neighbor $s' \in \mathcal{N}(s)$ and chooses the one with the highest gain. Those operations are very time consuming. For example, let us consider a problem without stereo strips and the possibility to remove 0, 1 or 2 shots to perform a move. Table 1 gives the computing costs of the insertion of a free shot (among the $2 \times (n - |s|)$ free ones).

Such insertion requires the test of $2|s|^2 - |s| + 2$ positions, and the total cost of an iteration is $2 \times (n - |s|) \times (2|s|^2 - |s| + 2)$, which is also the complexity (at the worst case) of the decision problem of a shot insertion. In order to overcome this complexity, incremental computing techniques are used [7]. The main idea is to use a specific data structure containing for each possible move the corresponding gain and the resulting configuration if the insertion is really performed. Each time a move

**Table 1** Computing complexity for one image insertion

| # removals | # sub_schedules | # insertion tests |
|---|---|---|
| 0 | 1 | $\|s\|+1$ |
| 1 | $\|s\|$ | $\|s\|^2$ |
| 2 | $\|s\|-1$ | $(\|s\|-1)^2$ |

**Algorithm 1:** Evaluate–$\mathcal{N}(s)$

```
begin
    L_cand ← ∅;
    best_gain ← −∞;
    D_s ← Generate-sub-schedules(s); % D_s = {s, s_1, s_2, ...} ;
    for all i, such that x_i = 0 do
        for l = 1 to |D_s| do
            for m = 0 to |s_l| do
                if Insert(s_l, i, p_m) = True then
[1]                 if (Gain(s_l, i, p_m) > g*) or (s_l is not tabu) then
                        if Gain(s_l, i, p_m) > best_gain then
                            L_cand ← ∅;
                            best_gain ← Gain(s_l, i, p_m) ;
                            L_cand ← {(s_l, i, p_m)} ;
                        else
                            if Gain(s_l, i, p_m) = best_gain then
                                L_cand ← L_cand ∪ {(s_l, i, p_m)} ;
    return (L_cand, best_gain)
end;
```

is carried out, the elements of this data structure which are affected by the move are updated accordingly.

The neighborhood is evaluated in accordance to Algorithm 1. The used notations are as follows:

- $\mathcal{L}_{cand}$ is the list of the candidate moves among the neighbors of the current configuration.
- *best_gain* is the best gain associated to the configuration obtained by the insertion of a free shot in the current configuration $s$ and contained in $\mathcal{L}_{cand}$.
- *Generate-sub-schedules*$(s)$ is the function which generates sub-schedules from $s$ by removing 0, 1 or 2 successive images (and their twins if necessary).
- The set of the sub-schedules produced by the function *Generate-sub-schedules*$(s)$ is denoted by $D_s$. The first element of $D_s$ is $s$ (no removal). We denote by $|D_s|$ the number of sub-schedules in $D_s$ and by $s_l$ the $l$th sub-schedule in $D_s$.
- $p_m$ denotes the position numbered by $m$ and situated between two shots which are scheduled respectively at the orders $m$ and $m+1$ according to their shooting start date. In particular, $p_0$ is the position before the first acquired shot and $p_{|s_l|}$ is the position after the last acquired shot in $s_l$.

*Example 1* Consider $s_l = \{(x_1 = 1,\ t_1 = 60), (x_2 = 0,\ t_2 = \infty), (x_3 = 0,\ t_3 = \infty), (x_4 = 0,\ t_4 = \infty), (x_5 = 1,\ t_5 = 40), (x_6 = 0,\ t_6 = \infty), (x_7 = 1,\ t_7 = 50), (x_8 = 0,\ t_8 = \infty)\}$. If we order the selected shots 1, 5 and 7 ($x_1 = x_5 = x_7$) in $s_l$ according to their shooting start date then we obtain $shot\ 5 < shot\ 7 < shot\ 1$ ($t_5 < t_7 < t_1$). Hence, we can express the different positions $p_m$ as follows: $\underline{p_0} - shot\ 5 - \underline{p_1} - shot\ 7 - \underline{p_2} - shot\ 1 - \underline{p_3}$.
- $p_0$ is the position before the shot 5.
- $p_1$ is the position between the shots 5 and 7.
- $p_2$ is the position between the shots 7 and 1.
- $p_3$ is the position after the shot 1.

- $g^*$ is the gain associated to the best schedule already reached $s^*$ from the beginning of the resolution, $g^* = G(s^*)$.
- $Insert(s_l, i, p_m)$ is a function which returns *True* if the insertion of the shot $i$ at position $p_m$ in a sub-schedule $s_l$ is feasible regarding to the problem constraints, *False* otherwise.
- $Gain(s_l, i, p_m)$ is the associated gain to the configuration resulted from the insertion of the shot $i$ in $s_l$ at the position $p_m$.

In Algorithm 1, we start by generating the set $D_s$ of the sub-schedules obtained by removing some shots. Then we try to insert each free shot $i$ in those sub-schedules. If an insertion is successful then we calculate its corresponding gain. Afterward, this insertion becomes candidate, the list of the best candidates is saved in $\mathcal{L}_{cand}$ and their associated gain value is *best_gain*. If a candidate move improves the current best gain then $\mathcal{L}_{cand}$ will contain only this move. The list $\mathcal{L}_{cand}$ will be used in the move heuristic as it will be explained in the next sections.

### 4.4 Tabu list management and move heuristic

We define a move by the insertion of a free shot $i$ (flipping $x_i$ from 0 to 1) followed by the removal of the conflicting shots that do not satisfy the problem constraints (for each conflicting shot $j$, we flip $x_j$ from 1 to 0). Now, we explain both the management of the tabu list and the heuristic used to select one of the move candidates.

#### *4.4.1 Tabu list management*

The role of a *tabu list* is to prevent short-term cycling. In this order, when a shot $i$ is selected and scheduled (a move is carried out), this shot is classified tabu (forbidden for any change) for a certain time called the *tabu tenure*. In our TS algorithm, this tenure is dynamically formulated by:

$$tabu(i) = iter + \alpha \times freq(i), \quad \text{where:}$$

- $iter$ is the number of the current iteration of the tabu algorithm.
- $freq(i)$ counts the number of times that the shot $i$ has been selected by the tabu algorithm (note that a shot can be inserted at a given iteration and be dropped some iterations later, then reselected after and so on).

– $\alpha$ is a variable parameter used to weight $tabu(i)$ according to $freq(i)$.

Moreover, a sequence of shots is tabu if all its shots are tabu, and not tabu if it contains at least one non-tabu shot. Likewise, we state that a sub-schedule obtained from a current schedule by removing a tabu sequence is also tabu, otherwise it is not tabu. Additionally, the direct azimuth corresponds to the natural move direction of the satellite and changing azimuth between two shootings is very costly in the terms of transition time. Consequently, the acquisitions in the direct azimuth are preferred over the indirect ones. This preference is expressed by setting $\alpha = 2 \times \beta$ for the shots acquired in a direct azimuth and $\alpha = \beta$ for the indirect ones (the value of $\beta$ is fixed empirically). Hence, if $i$ is odd then $tabu(i) = iter + 2 \times \beta \times freq(i)$ else $tabu(i) = iter + \beta \times freq(i)$. Recall that in Sect. 2.1 we use odd and even numbers to differentiate the shots regarding to their acquisition directions.

#### *4.4.2 Move heuristic*

Once the neighborhood is evaluated, the selected shot (move) is the one that maximizes the gain value and does not remove a sequence of tabu shots. However, if a move strictly improves the best gain then the *aspiration criterion* is employed to cancel the tabu status of a sub-schedule. Therefore in Algorithm 1, the condition labeled [1] corresponds to either the best gain $g^*$ is strictly improved (aspiration criterion) or the sub-schedule $s_l$ is not tabu. To summarize, we start by constructing $\mathcal{L}_{cand}$, we select randomly one candidate move $(s_l, i, p_m)$ from $\mathcal{L}_{cand}$, then we insert the shot $i$ in $s_l$ at the position $p_m$ to obtain the neighbor configuration $s'$ that replaces the current one, which completes the move.

As described above, if some moves lead to the same gain value then one of them is randomly chosen. However, this selection criterion is not the most effective one. For this reason and to tone down the random effect, we introduce a second objective function which is the minimization of the sum of the transition durations in a configuration $s$. We denote this sum by $TdT(s)$.

### 4.5 A secondary optimization problem: minimization of $TdT$

Some moves may lead to the same gain value. In this case and to make a better choice than a random one, a second objective function is defined by the minimization of the sum of transition durations that separate the acquisition of the shots. This minimization is done according to the constraints (1) to (4) of the original AEOS problem:

$$\text{Min } TDT(s) = \sum_{i=1}^{2n} \sum_{\substack{j=1 \\ j \neq i}}^{2n} x_i \cdot x_j \cdot y_{i \to j} \cdot d(i \to j)$$

subject to:

(1) For any strip $j \in [1, n]$, $x_{2j-1} + x_{2j} \leq 1$.
(2) For any stereo strip $j \in [1, n]$ and $tw(j) \neq 0$, we have ($x_{2j-1} = x_{2tw(j)-1}$ and $x_{2j} = x_{2tw(j)}$).

(3) For any shot $i \in [1,\ 2n]$, if $x_i = 1$ then $t_i \in [es(i), ls(i)]$.
(4) $\forall i \neq j \in [1,\ 2n]$ ($i$ and $j$ are shots), $t_j - t_i \geq (d(i) + t(i \rightarrow j)) \cdot y_{i \rightarrow j} + (es(j) - ls(j)) \cdot (1 - y_{i \rightarrow j})$.

*The enhanced aspiration criteria*

Before solving this second optimization problem, we improve the aspiration criterion as follows. The tabu status of a sub-schedule is canceled if one of the two conditions below is verified:

1. if a move leads to a configuration $s'$ better than the best configuration $s^*$ found so far, i.e. $G(s') > G(s^*) = g^*$,
2. if a move leads to a configuration $s'$ with the same gain value of $s^*$ but with a lower $TdT$ value than $TdT(s^*)$, i.e. $G(s') = G(s^*)$ and $TdT(s') < TdT(s^*)$.

In order to tackle the $TdT$ minimization problem, we design a second tabu algorithm based on the inversion of the acquisition direction of the strips and the exchange of the order of the shots in the considered configuration. This resolution is the subject of Sect. 5.

### 4.6 Intensification and diversification phases

The tabu mechanism may lead to a state where no move is admissible (all moves are tabu). This occurs when each possible move has been tried a large number of times without improving the best configuration already reached $s^*$. In this case, we launch an intensification phase. This one is based on a heuristic using a long-term information [10]. To this end, when the gain cannot be improved (all the shots are tabu and the aspiration criterion is not satisfied) the intensification phase attempts to overcome this situation by exploiting the best schedule $s^*$ as follows:

**First step:** The minimization of the sum of the transition durations of $s^*$ is tackled by a dedicated TS algorithm (see Sect. 5). The minimization of $TdT(s^*)$ is called *the smoothing step*. If it succeeds (i.e. the $TdT(s^*)$ value is decreased) then the tabu status of all the shots are set to 0, and the tabu exploration is restarted from the *smoothed* $s^*$. This step is very important. In fact, we have observed that during the experimental process if this first step has been achieved successfully then we may insert a free shot without any removal and consequently improve the best gain $g^*$.

**Second step:** When the smoothing step fails, we decrease the $\beta$ value by dividing it by 2, $\beta \leftarrow \beta/2$. Hence, the tabu durations are reduced (recall that $\beta$ is a variable parameter of the tabu tenure of a shot, see Sect. 4.4). Accordingly, if $\beta > 1$ then the tabu status of all the shots are set to 0 and the tabu exploration is restarted from the best solution $s^*$.

As described above, the intensification phase alternates between two exclusive steps. In fact, the execution of the fist step (respectively, the second step) inhibits the execution of the second one (respectively, the first one). Its aim is to focus the exploration around the elements of the best solution by either reordering its selected strips

and inverting their acquisition directions ($TdT(s^*)$ minimization) or by decreasing the tabu tenure of the shots. However, if the intensification phase does not improve the best gain then a diversification process is applied.

The role of diversification is to escape from the attractive zones of the search space corresponding to the local minima. For this reason, when both the tabu exploration and the intensification phase fail, we generate a new starting point (first schedule) different from the last one used at the beginning of the tabu exploration, we set the tabu status of all the shots to 0 and the $\beta$ parameter to a new value fixed empirically, then we restart the tabu search from this new point.

### 4.7 A greedy algorithm for the AEOS problem

To produce quickly a first feasible solution for the AEOS problem, we design and use a greedy algorithm. This one is based on a simple operation of inserting a shot that does not require the removal of the already selected shots. Also, each shot $i \in [1,\ 2n]$ is weighted according to the number of times that it was selected by the greedy algorithm. We denote by $wg(i)$ this weight which is initialized to 0.

When it is launched, the greedy algorithm starts by sorting the $2n$ shots in the increasing order of their weights $wg$. This treatment is accomplished by the function *SortShots*() in Algorithm 2 and the result is stored in the list $Q$ ($Q_j$ is the number of the shot that is ordered at the position $j$ in $Q$ according to its $wg(Q_j)$ value). The first element of $Q$ is the least selected shot by *greedy*() and the last one is the most selected one. Hence, acquiring the shots according to their weights aims to favor the acquisition of the less selected shots by the greedy algorithm.

In Algorithm 2, $y$ is a configuration where any shot is selected ($\forall i \in [1,\ 2n]$, $x_i = 0$). In the loop "for", we try to acquire each shot $Q_j$ by satisfying all the problem constraints. Moreover and oppositely to the move heuristic that we have defined before, we forbid removing the shots that are already fixed in $y$. In fact, if the condition at the line [1] is not checked then we simply skip the current shot and handle the following one in $Q$.

**Algorithm 2:** greedy()

```
begin
    let be y a configuration where ∀i ∈ [1, 2n] we have x_i = 0;
    % Q is the list of the 2 × n shots sorted in the increasing order of the wg values
    Q ← SortShots();
    % Q_j is the jth element of Q
    for j = 1 to 2n do
[1]     if setting x_{Q_j} = 1 in y satisfies the constraints (1) to (4) then
            x_{Q_j} ← 1;
            wg(Q_j) ← wg(Q_j) + 1
    return y
end;
```

Regarding to the weights *wg* and to their corresponding sorted shots list $Q$, two executions of Algorithm *greedy*() will return two different configurations in terms of the selected strips. This feature allows us to deal with the diversification step that needs different starting points (see last paragraph of Sect. 4.6).

### 4.8 Global tabu resolution

The TS algorithm follows a general scheme consisting of three iterative phases: exploration, intensification and diversification. All these resolution steps are given in Algorithm 3. It starts by initializing the different structures that it uses, such as the *wg*, *freq* and *tabu* values. The greedy algorithm returns a first feasible solution $s_0$ constructed by successive image insertions without any removal. $|s_0|^+$ is the number of the selected shots in $s_0$ and which are acquired in the direct azimuth. The instruction labeled [2] tunes the value of the parameter $\alpha$ according to the shooting direction of an image and as explained in Sect. 4.4: we recall that an odd (even) number indicates a shot that is acquirable on the direct (indirect) azimuth. The *Tabu–*$TdT(s^*)$ function (label [3]) corresponds to the tabu algorithm dedicated to the minimization of $TdT(s^*)$ value. Finally, the *stop criterion* is a limited execution time.

The diversification phase is launched when the intensification step fails. Hence, we execute a new iteration of the most external loop **repeat ... until** by generating a new starting point $s_0$ by the means of the greedy algorithm and executing the tabu search, ...

The different phases (exploration, intensification and diversification) use the same tabu search engine. Each one is triggered and stopped automatically by the tabu list management, i.e. whenever no more move is admissible.

## 5 Tabu algorithm for minimizing $TdT$

As we have seen previously, the $TdT$ minimization is used in the intensification process. We recall that for a given configuration $s$, we have $TdT(s) = \sum_{i=1}^{2n} \sum_{j=1,\, j\neq i}^{2n} x_i \cdot x_j \cdot y_{i\to j} \cdot d(i \to j)$. The aim is to obtain a shortest schedule in terms of the sum of the transition durations between the strip shootings. Indeed, the reduction of this sum can generate visibility time windows sufficiently broad and usable by the satellite to acquire new shots and without removing those that are already selected. the minimization of $TdT$ is based on two operations: the exchange of the order of the shots and the inversion of the acquisition direction of the strips. The order exchange is inspired by the *2–opt* operation [16] which is widely used in solving the Traveling Salesman Problem (TSP).

As in the last section, we detail the various elements of the TS algorithm dedicated to solve the $TdT$ minimization problem.

### 5.1 Neighborhood evaluation

As for the first TS algorithm, the explored search space is totally constrained. Hence, each visited configuration is consistent and may be saturated. Accordingly, we define the neighborhood function $\mathcal{N}_t$ over $X$ as follows:

**Algorithm 3:** *Tabu–AEOS*()

```
begin
    for i = 1 to 2n do wg(i) ← 0; % initialization of the wg weights
    repeat
        % initialization of the tabu and the frequency values of each shot
        for i = 1 to 2n do
            x_i ← 0; freq(i) ← 0; tabu(i) ← 0;
[1]     s ← s_0 ← greedy();          % the initial configuration
        s* ← s;                      % initialization of the best configuration
        g* ← G(s*);                  % initialization of the best gain
        β ← |s_0|^+;                 % initialization of the β factor
        iter ← 0;                    % initialization of the iteration number
        repeat
            (L_cand, best_gain) ← Evaluate-N(s);
            if L_cand ≠ ∅ then
                %% Tabu exploration
                (s_l, i, p_m) ← randSelect(L_cand);
                s ← propagate_move(s_l, i, p_m); % Inserts i in s_l at p_m and propagates all the
                constraints
                freq(i) ← freq(i) + 1;
                iter ← iter + 1;
[2]             if (i modulo 2) = 1 then
                    α ← 2 × β;
                else α ← β;
                tabu(i) ← iter + α × freq(i);
                if (best_gain > g*) or (best_gain = g* and TdT(s) < TdT(s*)) then
                    s* ← s; g* ← best_gain;
            else
                % intensification: step 1
[3]             (z, TdT_z) ← Tabu–TdT(s*);
                if TdT_z < TdT(s*) then s* ← z;
                else
                    % intensification: step 2
                    β ← β/2;
                for i = 1 to 2n do freq(i) ← 0; tabu(i) ← 0;
                s ← s*;
        until (β < 1 or |s*| = n);
        %% If (β < 1) then Diversification phase
    until (stop criterion or |s*| = n);
    return (s*, g*);
end;
```

Let $s = ((x_1, t_1), (x_2, t_2), \ldots, (x_{2n}, t_{2n}))$ be a current configuration, $s' = ((x'_1, t'_1), (x'_2, t'_2), \ldots, (x'_{2n}, t'_{2n}))$ is a neighbor of $s$, i.e. $s' \in \mathcal{N}_t(s)$, if and only if the following conditions are verified:

1. $\exists! i \in [1,\ 2n] / \ x_i = 0$ and $x'_i = 1$. Moreover, if $tw(i) \neq 0$ then $x_{tw(i)} = 0$ and $x'_{tw(i)} = 1$.
2. For any shot $i$ that satisfies the condition 1 and a shot $j \in [1,\ 2n]$ such as $i$ and $j$ are issued from the same strip, we have $x_j = 1$ and $x'_j = 0$. Moreover, if $tw(i) \neq 0$ then $x_k = 1$ and $x'_k = 0$ such that $tw(i)$ and $k$ are also issued from the same strip. The strip, that corresponds to the shots $i$ and $j$, is acquired according to the direction of $j$ in $s$ ($x_j = 1$ in $s$). Setting $x'_j$ to 0 and $x'_i$ to 1 in $s'$ means that we change the shooting direction of this strip.
3. The constraints (3) and (4) are satisfied in both $s$ and $s'$.
4. $|s'| = |s|$. In other words, the same number of strips is selected in $s$ and $s'$.

In both the configurations $s$ and $s'$, the same strips are acquired (no strip is removed from $s$) as stated by the condition4. However, there is only one strip in $s'$ (condition 1) plus, if necessary, its twin for which the acquisition direction is inverted (condition 2). Moreover, to deal with time constraints (3) and (4), the order of the corresponding strip can be exchanged in the configuration $s'$ (in other words, its starting time changes).

A neighbor configuration $s'$ of $s$ is evaluated according to the sum of the transition durations between the shots that it contains. Hence, for each $s' \in \mathcal{N}_t(s)$, the value of $TdT(s')$ is calculated and the best neighbors are selected through Algorithm 4 which is given below. We use the following notations:

- $\mathcal{L}^t_{cand}$ is the list of the candidate moves.
- $best\_TdT$ is the best $TdT$ value obtained at the end of the evaluation of $\mathcal{N}_t(s)$

**Algorithm 4:** Evaluate–$\mathcal{N}_t(s)$

**begin**
  $\mathcal{L}^t_{cand} \leftarrow \emptyset$; $best\_TdT \leftarrow +\infty$ ;
  **for** $i \in [1,\ 2n]$, *such that* $x_i = 1$ *in* $s$ **do**
    $s^{-i} \leftarrow remove(s, i)$;
    **for** $m = 0$ ***to*** $|s^{-i}|$ **do**
      **if** $Insert(s^{-i}, j, p_m) = True$ **then**
        **if** $(TdT(s^{-i}, j, p_m) < TdT^*)$ *or* $((j, p_m)$ *is not tabu*$)$ **then**
          **if** $TdT(s^{-i}, j, p_m) < best\_TdT$ **then**
            $\mathcal{L}^t_{cand} \leftarrow \emptyset$;
            $best\_TdT \leftarrow TdT(s^{-i}, j, p_m)$ ;
            $\mathcal{L}^t_{cand} \leftarrow \{(s^{-i}, j, p_m)\}$ ;
          **else**
            **if** $TdT(s^{-i}, j, p_m) = best\_TdT$ **then**
              $\mathcal{L}^t_{cand} \leftarrow \mathcal{L}^t_{cand} \cup \{(s^{-i}, j, p_m)\}$ ;
  **return** $(\mathcal{L}^t_{cand}, best\_TdT)$;
**end**;

- $s^{-i}$ is the configuration issued from $s$ by removing the shot $i$ (and eventually its twin) which is already selected ($x_i = 1$ in $s$ and $x_i = 0$ in $s^{-i}$). $|s^{-i}|$ is the number of the selected shots in $s^{-i}$.
- $remove(s, i)$ is the function which removes the shot $i$ from $s$. The resulted configuration is $s^{-i}$.
- $j$ is the index of the shot issued from the same strip as $i$ but taken in the opposite direction.
- $TdT^*$ is the best $TdT$ value obtained for the whole second TS algorithm.
- $TdT(s^{-i}, j, p_m)$ is the $TdT$ value of the schedule obtained by inserting the shot $j$ in the position $p_m$ in $s^{-i}$.

At each iteration of the $\mathcal{N}_t(s)$ evaluation, each selected shot $i$ in $s$ is removed (the resulting schedule is $s^{-i}$). Then, we try to insert the shot $j$ in each position $p_m$ in $s^{-i}$ without any shot removal and by satisfying all the constraints (1) to (4). If an insertion is possible then we calculate the $TdT$ value of the new configuration obtained by adding the shot $j$ to $s^{-i}$. The best candidate moves with the lowest $TdT$ value ($best\_TdT$) are stored in $\mathcal{L}^t_{cand}$ which will be used in the phase of the move selection.

### 5.2 Tabu list management and move heuristic

For the $TdT$ minimization, the tabu tenure is defined for the pairs ($shot$, $position$). Indeed, each time a shot $j$ is inserted at position $p_m$ then the tabu tenure of $(j, p_m)$ is dynamically updated by the formula:

$$tabu(j, p_m) = iter' + \lambda \times freq'(j, p_m), \quad \text{where:}$$

- $iter'$ is the current iteration of the second tabu process.
- $freq'(j, p_m)$ is the number of the times that the shot $j$ was inserted at at the position $p_m$ in $s$.
- $\lambda$ is a variable parameter which is fixed empirically.

After the evaluation of the neighborhood $\mathcal{N}_t$, a shot $j$ (which corresponds to the changing of the acquisition direction of an already scheduled shot $i$) is inserted at the position $p_m$ if the resulting configuration has the minimum $TdT$ value and the pair $(j, p_m)$ is not tabu. However, an aspiration criterion is employed to cancel the tabu status when $TdT(s^*)$ (the best value already reached) is decreased. Hence, among all the candidate moves, one is randomly selected from $\mathcal{L}^t_{cand}$, the insertion of $j$ in $s^{-i}$ is performed and the constraints are propagated.

### 5.3 The general tabu algorithm for the $TdT$ minimization

Combining all the points described above leads us to Algorithm 5 for the $TdT$ minimization. This algorithm terminates when no more candidate moves are available (i.e. $\mathcal{L}^t_{cand} = \emptyset$) and it returns the best configuration found $s^*$ and its corresponding $TdT$ value, $TdT^*$. The value of $\lambda$ parameter is equal to the number of the acquired shots in $s$. The *propagate_move* function achieves the move and propagates all the

**Algorithm 5:** *Tabu–TdT*$(s)$

**begin**
  **for** *any shot* $i$ *and position* $p_l$ **do**
    $freq'(i,\ p_l) \leftarrow 0$; $tabu(i,\ p_l) \leftarrow 0$;
  $s^* \leftarrow s$; $TdT^* \leftarrow TdT(s^*)$;
  $iter' \leftarrow 0$;
  $\lambda \leftarrow |s|$;
  **repeat**
    $(\mathcal{L}^t_{cand}, best\_TdT) \leftarrow$ Evaluate–$\mathcal{N}_t(s)$ ;
    **if** $\mathcal{L}^t_{cand} \neq \emptyset$ **then**
      $(y, j, p_m) \leftarrow randSelect(\mathcal{L}^t_{cand})$ ;
      $s \leftarrow propagate\_move(y, j, p_m)$ ;
      $freq'(j, p_m) \leftarrow freq'(j, p_m) + 1$;
      $iter' \leftarrow iter' + 1$ ;
      $tabu(j, p_m) \leftarrow iter' + \lambda \times freq'(j, p_m)$ ;
      **if** $best\_TdT < TdT^*$ **then**
        $s^* \leftarrow s$;
        $TdT^* \leftarrow best\_TdT$;
  **until** $\mathcal{L}^t_{cand} = \emptyset$;
  **return** $(s^*,\ TdT^*)$;
**end**;

constraints of the problem, namely, setting $x_j = 1$ and $x_i = 0$ ($i$ and $j$ are issued from the same strip) to deal with the first problem constraint, updating the visibility time windows and updating the starting acquisition date. Note that we have not introduce in this resolution a diversification and an intensification phases with the aim to have short execution times of the *Tabu–TdT*$(s)$ algorithm.

Designing an exact algorithm to solve the AEOS management problem remains a difficult task even if a column generation approach was proposed but the obtained results were not so promising [17]. In Sect. 7 we will compare the performances of our resolution to the best known ones. Moreover, we try to approximate the bounding of the optima values by calculating upper bounds on a simplified problem as described in the next section.

## 6 Upper bounds for the AEOS problem

To calculate upper bounds in a polynomial time, some simplifications on the original AEOS problem are mandatory. As described in Sect. 2, the AEOS management problem is a Shortest Path Problem with Time Windows (SPPTW) (see for example [6]) with several additional properties increasing its hardness:

– Two scanning directions are possibles for each strip acquisition and mutually exclusive.
– The stereo constraints: some pairs of strips describe two acquisitions of the same geographical area under different angles. Selecting one without the other is forbidden and scanning directions must be identical.
– Requests are grouped into polygons whose gain is a convex piecewise linear function of the surface covered by the selected strips.

Consider the simplified AEOS management problem where the uniqueness and stereoscopic acquisition constraints are relaxed (the redundant strip acquisition and violation of the stereo constraint are allowed) and the evaluation function for polygons is linearized ($f(x) = x$). Accordingly, each instance of the AEOS management problem can be represented as a directed graph, in which each vertex $i$ is represented by the pair $(pe_i, te_i)$, where $pe_i$ is the geometric point of the end of a strip and $te_i$ is the end time of a strip acquisition. The directed edges represent the possible transition between vertices (the visibility time windows and the transition constraints are satisfied). A gain $Ge_i$ is associated with each vertex $i$ corresponding to the accumulated gain on the path reaching $i$ from a starting vertex. To keep the graph finite, the time $te_i$ must take values only from a discrete set, such as the natural numbers.

An optimal solution of the simplified AEOS management problem is just the longest path in this graph. Finding such a path deals with SPPTW which is solvable in a polynomial time. Hence, an adaptation of a dedicated algorithm to SPPTW can be successfully applied. For this reason, we elaborate a Dynamic Programming (DP) algorithm (as done in [14] also inspired by [6]) to deal with the simplified problem.

The constructed graph is circuit-free. However, a vertex might still be visited more than once, i.e. a strip may be acquired more than once on a path (redundancy). In [13], a method for eliminating 2 cycles (i.e. double acquisition of a single strip) is suggested but cannot easily be generalized to cycles of higher orders. The implemented DP algorithm incorporates a 2-cycle elimination but it is forced to accept cycles of higher order.

Now, we outline the main elements of the DP algorithm: the local optimal value of the quality criterion $Ge_i^*$ at a vertex $(pe_i, te_i)$ can be expressed as:

$$Ge_i^* = \underset{j}{\text{Max}}\{Ge_j^* + gs_i, \text{ such that constraints (3) and (4) are satisfied}\}$$

where $gs_i$ is the gain of the strip from which the vertex $(pe_i, te_i)$ is issued. Satisfying constraints (3) and (4) means the existence of an edge from vertex $(pe_j, te_j)$ to $(pe_i, te_i)$ in the constructed graph.

Accordingly, we define a dominance rule, for an efficient evaluation of each edge in the DP algorithm as follows: given two vertices $i$ and $j$ represented by $(pe_i, te_i)$ and $(pe_j, te_j)$ and characterized by the accumulated gains $Ge_i$ and $Ge_j$ respectively then:

$$(i \text{ dominates } j) \text{ if and only if } (pe_i = pe_j \text{ and } te_i \leq te_j \text{ and } Ge_i \geq Ge_j).$$

Several strategies to decrease the computation time of the algorithm have been proposed. In [5], the authors suggested a method to decrease the width of the time win-

dow and also proposed using a pulling algorithm instead of the more straightforward reaching algorithm usually used in DP-algorithms for shortest path problems.

After presenting both algorithms dedicated to the resolution of the AEOS management problem and the calculation of the upper bounds, we give and discuss the results obtained on the provided benchmarks.

## 7 Computational results

The AEOS management problem was the subject of the 3rd international challenge organized by the French Society of Operations Research and Decision Analysis (ROADEF'2003), and proposed by the CNES[1] and ONERA[2] French space agencies.

### 7.1 Test instances and experimental settings

The provided instances are artificially generated, with the number of requests ($m$ value) ranging from 2 to 375, and the number of strips ($n$ value) varying from 2 to 534 with a maximum of 113 stereo strips ($n_{stereo}$ value). Table 2 gives the properties of each of the 20 used instances.[3]

The TS and DP algorithms are implemented in C/C++ language and compiled using Visual C++. The experiments were carried on a PIV 1.9 Ghz PC with 512 MB of RAM. The execution time, corresponding to the stop criterion of Algorithm 3, is fixed to 1800 seconds in order to evaluate the behavior of the algorithm over a relatively long computation time. Table 3 gives the obtained results. The TS algorithm was run 10 times per instance with different random seeds and the following information are collected:

**Table 2** Test instance characteristics

| Instance | $m$ | $n$ | $n_{Stereo}$ | Instance | $m$ | $n$ | $n_{Stereo}$ |
|---|---|---|---|---|---|---|---|
| 2 9 36 | 2 | 2 | 0 | 2 21 140 | 284 | 420 | 58 |
| 2 9 66 | 4 | 7 | 0 | 2 21 155 | 311 | 472 | 55 |
| 2 13 111 | 68 | 106 | 12 | 2 21 170 | 294 | 450 | 71 |
| 2 15 170 | 218 | 295 | 39 | 2 21 22 | 306 | 455 | 54 |
| 2 26 96 | 336 | 483 | 63 | 2 21 37 | 315 | 477 | 62 |
| 2 27 22 | 375 | 534 | 67 | 2 21 7 | 289 | 410 | 49 |
| 2 9 170 | 12 | 25 | 4 | 2 21 81 | 297 | 436 | 59 |
| 3 25 22 | 150 | 342 | 113 | 2 21 96 | 291 | 437 | 49 |
| 3 8 155 | 12 | 28 | 10 | 3 21 155 | 135 | 295 | 105 |
| 4 17 186 | 77 | 147 | 48 | 3 21 81 | 135 | 283 | 88 |

[1] Acronym of "Centre National d'Etudes Spatiales"

[2] Acronym of "Office National d'Etudes et de Recherches Aérospatiales"

[3] Available from the WEB site of the challenge: www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2003

**Table 3** The results obtained after 1800 seconds running time

| Instance | Best | | | Worst | Average | | Best known |
|---|---|---|---|---|---|---|---|
| | Gain | $N_s$ | Time | Gain | Gain | Time | Gain |
| 2 9 36 | 10,423,440 | 2 | <1 | 10,423,440 | 10,423,440 | <1 | 10,423,440 |
| 2 9 66 | 115,710,959 | 7 | <1 | 115,710,959 | 115,710,959 | <1 | 115,710,959 |
| 2 13 111 | 563,597,071 | 54 | 90 | 563,597,071 | 563,597,071 | 90 | 563,597,071 |
| 2 15 170 | 719,417,220 | 40 | 221 | 719,417,220 | 719,417,220 | 561 | 719,417,220 |
| 2 26 96 | 1,005,301,900 | 35 | 1725 | 985,763,300 | 989,671,020 | 443 | 1,005,301,900 |
| 2 27 22 | 966,643,460 | 30 | 57 | 966,643,460 | 966,643,460 | 57 | 967,910,750 |
| 2 9 170 | 191,358,231 | 17 | <1 | 191,358,231 | 191,358,231 | <1 | 191,358,231 |
| 3 25 22 | 425,983,220 | 28 | 95 | 425,983,220 | 425,983,220 | 95 | 425,983,220 |
| 3 8 155 | 121,680,360 | 12 | <1 | 121,680,360 | 121,680,360 | <1 | 121,680,360 |
| 4 17 186 | 185,406,200 | 37 | <1 | 185,406,200 | 185,406,200 | <1 | 185,406,200 |
| 2 21 140 | 1,030,060,860 | 32 | 191 | 1,027,543,540 | 1,029,288,814 | 467 | 1,029,892,360 |
| 2 21 155 | 1,150,632,847 | 35 | 1074 | 1,129,245,020 | 1,132,568,329 | 762 | 1,150,632,847 |
| 2 21 170 | 914,978,310 | 40 | 573 | 906,992,592 | 912,183,616 | 1151 | 891,060,370 |
| 2 21 22 | 1,160,594,340 | 32 | 115 | 1,160,366,840 | 1,160,571,590 | 831 | 1,160,366,840 |
| 2 21 37 | 954,965,580 | 37 | 190 | 954,605,760 | 954,821,652 | 602 | 954,965,580 |
| 2 21 7 | 842,378,700 | 33 | 115 | 842,378,700 | 842,378,700 | 116 | 842,378,700 |
| 2 21 81 | 986,679,410 | 30 | 897 | 98,424,5930 | 985,894,172 | 778 | 986,679,410 |
| 2 21 96 | 113,4461,030 | 38 | 843 | 1,125,880,120 | 1,130,197,543 | 932 | 1,133,044,250 |
| 3 21 155 | 460,196,570 | 36 | 3 | 460,196,570 | 460,196,570 | 3 | 4,6019,6570 |
| 3 21 81 | 37,3553,350 | 28 | 16 | 373,553,350 | 373,553,350 | 16 | 373,553,350 |

– Over the 10 runs, the best gain value, the time needed to reach this gain and the number of the selected strips in the corresponding solution (columns 2, 3 and 4 of Table 3).
– Over the 10 runs, the worst gain (column 5), and the average gain and time (columns 6 and 7).
– The best known gains (column 8) issued from the published results during the ROADEF' 2003 challenge, in a booklet of abstracts [17] and also available on the WEB site of the challenge[3], but using different test parameters: Sun-Blade-1000 750 MHz/512 MB workstation and 300 seconds running time.

### 7.2 Result discussion

For the ten first instances (from 2 9 36 to 4 17 186, which we call A instances), all the best known gains are reached except for instance 2 27 22. These values are obtained after a maximum of 221 sec, except for instance 2 26 96, which required 1725 sec. For the ten instances 2 21 140 to 3 21 81 (B instances), all the best values are reached after less than 200 sec computing time except for the instances 2 21 155 and 2 21 81, which require 1074 and 897 sec respectively. Furthermore, the gains which are

**Table 4** Upper bound values

| Instance | Best | UB | Gap % |
|---|---|---|---|
| 2 9 36 | 10,423,440 | 10,423,440 | 0.00 |
| 2 9 66 | 115,710,959 | 115,710,959 | 0.00 |
| 2 13 111 | 563,597,071 | 750,675,448 | 24.92 |
| 2 15 170 | 719,417,220 | 1,007,583,810 | 28.60 |
| 2 26 96 | 1,005,301,900 | 1,262,199,140 | 20.35 |
| 2 27 22 | 966,643,460 | 1,225,871,727 | 21.15 |
| 2 9 170 | 191,358,231 | 191,358,231 | 0.00 |
| 3 25 22 | 425,983,220 | 595,723,590 | 28.49 |
| 3 8 155 | 121,680,360 | 121,680,360 | 0.00 |
| 4 17 186 | 185,406,200 | 233,270,070 | 20.52 |
| 2 21 140 | 1,030,060,860 | 1,268,577,920 | 18.80 |
| 2 21 155 | 1,150,632,847 | 1,445,129,777 | 20.38 |
| 2 21 170 | 914,978,310 | 1,177,206,242 | 22.27 |
| 2 21 22 | 1,160,594,340 | 1,478,755,136 | 21.51 |
| 2 21 37 | 954,965,580 | 1,281,931,945 | 25.51 |
| 2 21 7 | 842,378,700 | 1,060,675,390 | 20.58 |
| 2 21 81 | 986,679,410 | 1,315,194,307 | 24.97 |
| 2 21 96 | 1,133,044,250 | 1,366,459,404 | 17.08 |
| 3 21 155 | 460,196,570 | 604,049,451 | 23.81 |
| 3 21 81 | 373,553,350 | 492,952,136 | 24.22 |

obtained for the instances 2 21 140, 2 21 170, 2 21 22 and 2 21 96 are improved after a maximum of 843 sec.

Several comments can be made about these results. First, our TS algorithm is efficient and robust. In fact, 15 best gains are reached and 4 are improved, even if the best known gains were obtained on different experimentation conditions and the optimal gains are unknown. About the robustness of the algorithm, the gap value between the best gain found and the average gain, calculated using the formula $100 \times (\frac{Best}{Average} - 1)$, is equal to 0% for 12 instances (the best known value is always reached), and less then 1.60% for the rest of the benchmark, which is a poor value. Secondly, these instances seems highly constrained since only a small number of candidate strips are selected in the best solutions.

In Table 4, the gaps with the upper bounds are calculated by the formula $100 \times (1 - \frac{Best}{UB})$, where the column *Best* corresponds to the best gain values obtained by our TS algorithm and the column *UB* to the gains obtained by the dynamic programming algorithm on the simplified problem as explained in Sect. 6. For the instances 2 9 36, 2 9 66, 2 9 170 and 3 8 155, our TS algorithm reaches the optimality. For the other instances the gaps are between 17.08% and 28.60%. Even if these gaps are not so tight, the results obtained on the simplified problem (by the relaxation of some constraints and the linearization of the objective function) at least prove the optimality of four instances.

## 8 Conclusion

In this paper, we have proposed a tabu search algorithm to solve the problem of selecting and scheduling photographs of an Agile Earth Observing Satellite (AEOS). This problem is formulated as a constrained optimization problem. The involved constraints are both unary and binary and the evaluation function, to be maximized, is convex which increases the hardness of the problem. The TS algorithm explores a search space based on consistent and possibly saturated configurations. Consistency is ensured by an effective constraint propagation for each new move carried out during the search process. The resulting neighborhood is quickly evaluated by the use of incremental techniques. Moreover, the TS algorithm is bridged with systematic search by means of partial enumerations in order to solve a large number of decisional problems. In addition and to improve the efficiency of the resolution, we introduced a secondary problem which is the minimization of the sum of the transition durations in a schedule. A second tabu search algorithm was developed to tackle it where the neighborhood construction is based on exchanging the order of the acquired strips in the treated configuration and inverting their acquisition directions.

Furthermore and as an exact algorithm is hard to design for the AEOS management problem, we calculated upper bounds for a simplified problem by relaxing the uniqueness and stereo acquisition constraints, and the linearization of the objective function. Then, a dynamic programming algorithm was launched on this less constrained problem.

Our resolution was applied on benchmarks provided by the French space agencies for the ROADEF'2003 challenge. The results obtained showed the efficiency of the TS algorithm. Indeed, all the best known results were reached (except for one instance) and optimality was proved for 4 instances. Note that those instances seem highly constrained because of the low number of the selected strips in the obtained solutions.

Concerning future work, the stereoscopic constraint needs to be handled in a more efficient way. This constraint, which consumes a large amount of computing time when it is handled, increases the difficulty of the problem, and unfortunately the proportion of the stereo shots in the solutions is generally very small. In addition, we will enhance the efficiency of the second optimization problem (the minimization of the sum of transition durations in a schedule) by using more powerful techniques taken from the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem with Time Windows (VRPTW). Finally, the optimal values of the AEOS managing problem remain unknown. To overcome this weakness, we will study a way to improve the tightness of the upper bounds. A first possibility is to use the techniques based on the Lagrangian relaxation.

## References


1. Benoist, T., Rottembourg, B.: Upper bounds of the maximal revenue of an Earth observation satellite. 4OR: Q. J. Oper. Res. **2**(3), 235 (2004)
2. Bensana, E., Agnèse, G.V.J., Bataille, N., Blumstein, D.: Exact and approximate methods for the daily management of an Earth observation satellite. In: Proceeding of the 4th International Symposium od Space Mission Operations and Ground Data Systems (spaceOps–96) (1996)

3. Bensana, E., Lemaître, M., Verfaillie, G.: Earth observation satellite management. Constraints: Int. J. **4**(3), 293–299 (1999)
4. Cordeau, J., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. J. Oper. Res. Soc. **59**, 928–936 (2001)
5. Desrochers, M., Desrosiers, J., Solomon, M.: A new optimization algorithm for the vehicle routing problem with time window. Oper. Res. **40**, 342–354 (1992)
6. Desrochers, M., Soumis, F.: A generalized permanent labelling algorithm or the shortest path problem with time windows. INFOR **26**, 191–212 (1988)
7. Fleurent, C., Ferland, J.A.: Genetic and hybrid algorithms for graph coloring. Ann. Oper. Res. **63**, 437–461 (1996)
8. Gabrel, V.: Improved linear programming bounds via column generation for daily scheduling of Earth observation satellite. Technical report, LIPN (1999)
9. Gabrel, V., Moulet, A., Murat, C., Paschos, V.T.: A new model and derived algorithms for the satellite shot planning problem using graph theory concepts. Ann. Oper. Res. **69**, 115–134 (1997)
10. Glover, F., Laguna, M.: Tabu Search. Kluwer, Amsterdam (1997)
11. Hall, N., Magazine, M.: Maximizing the value of a space mission. Eur. J. Oper. Res. **78**, 224–241 (1994)
12. Harrison, S.A., Price, M.E.: Task scheduling for satellite based imagery. In Proceedings of the 18th Workshop of UK Planning and Scheduling Special Interest Group, pp. 64–78 (1999)
13. Houck, D., Picard, J., Queyranne, M., Vemuganti, R.: The travelling salesman problem as a constrained shortest path problem: theory and computational experience. Oper. Res. **17**(2–3), 93–109 (1980)
14. Kohl, N., Madsen, O.B.G.: An optimization algorithm for the vehicle routing problem with time windows based on Lagrangean relaxation. Oper. Res. **45**, 395–406 (1997)
15. Lemaître, M., Verfaillie, G., Jouhaud, F., Lachiver, J.M., Bataille, N.: Selecting and scheduling observations of agile satellites. Aerosp. Sci. Technol. **6**(5), 367–381 (2002)
16. Lin, S.: Computer solutions of the traveling salesman problem. Bell Syst. Comput. J. **44**, 2245–2269 (1965)
17. ROADEF'2003 Challenge: Booklet of Abstracts. ROADEF society, France (2003)
18. Schiex, T., Fargier, F., Verfaillie, G.: Valued constrained satisfaction problems: hard and easy problems. In Proceedings of IJCAI'95, 14th International Joint Conference on Artificial Intelligence pp. 631–639 (1995)
19. Vasquez, M., Habet, D., Dupont, A.: Neighborhood design by consistency checking. In the Proceedings of the First International Workshop on Heuristics (IWH'02), vol. 4, pp. 19–27 (2002)
20. Vasquez, M., Hao, J.K.: 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), 137–157 (2001)
21. Vasquez, M., Hao, J.K.: Upper bounds for the SPOT5 daily photograph scheduling problem. J. Comb. Optim. **7**, 87–103 (2003)
22. Wolf, W., Sorensen, S.: Three scheduling algorithms applied to the Earth observing systems domain. Manag. Sci. **46**(1), 146–168 (2000)